6.6 NETCDF-4: SOFTWARE IMPLEMENTING AN ENHANCED DATA MODEL FOR THE GEOSCIENCES

Russ Rew*, Ed Hartnett, and John Caron
Unidata Program Center, Boulder, Colorado

1. INTRODUCTION

Developed by the Unidata Program of the University Corporation for Atmospheric Research (UCAR), netCDF is widely used in earth, ocean, and atmospheric sciences because of its simple data model, ease of use, portability, and strong user support infrastructure. Use of the netCDF data model, data access libraries, and machine independent format for the creation, access, and sharing of data in the geosciences continues to grow.

HDF5 software, originally developed at the National Center for Supercomputing Applications (NCSA) and now developed, maintained, supported, and distributed by The HDF Group, Inc., implements another popular data model, data access libraries, and format for scientific data. The use of HDF5 is also increasing.

Over the last two years, the groups who develop and maintain the associated netCDF and HDF5 software have been collaborating in creating software that uses enhancements to the HDF5 data model and format to implement a richer netCDF data model. The result is intended to combine some of the desirable characteristics of netCDF and HDF5, while taking advantage of their separate strengths. The NetCDF-4 library provides compatibility with existing netCDF programs and data, additional data modeling abstractions, and features for use in high performance computing, such as parallel I/O.

After providing some background, we describe additions to the netCDF data model and make recommendations for data providers and developers who may be considering the use of netCDF-4 for future archives or applications.

2. BACKGROUND

Although netCDF (Unidata, 2005) and HDF5 (NCSA, 2005) are typically referred to as file formats for portable, self-describing data, each is also a *data model* that organizes a collection of associated abstractions into a high-level view of how to access data, above the level of input-output facilities provided in particular programming languages. Both data models are implemented as freely available libraries supporting application programming interfaces (APIs) in multiple programming languages.

Use of a data model provides a better level of abstraction to describe the higher-level data objects that netCDF and HDF offer, well above the level of bits, bytes, and disk blocks. A data model appropriate for scientific data access provides advantages similar to the relational model for accessing highly structured tables in enterprise databases: a logical view of data independent from low-level details of storage and from the particular language interface used to access the data.

While relational databases are adequate for many kinds of highly structured scientific data, traditional relational database systems lack adequate support for access to data in multidimensional arrays, good tools for analysis and visualization, the ability to handle large data volumes efficiently using access patterns common in the sciences, and simple programming language interfaces for such data access patterns, according to Gray (2005). As a result, alternate data models such as netCDF and HDF5 have evolved to support useful abstractions for scientific data access.

For example, each data model has the notion of a named multidimensional array of data elements of the same abstract type: a v*ariable* in netCDF parlance and a d*ataset* in HDF5. Both models use the term a*ttribute* to describe metadata that can be attached to other data objects to provide ancillary information, such as the units of measure. Both data models provide independence from the physical representation of the data, insulating applications from locating desired data by disk offsets, or dealing with access changes necessitated by the addition of new variables or attributes to existing datasets.

The HDF5 data model provides more types, abstractions, and mechanisms for extensibility than netCDF, which makes it more powerful for modeling complex data and relationships, but somewhat more difficult to master. It represents data within groups (providing name scopes like directories in a filesystem) as collections of multidimensional arrays of structures, with links providing names for groups and structures. Named attributes can be attached to each dataset or group. The shape of datasets may be dynamic, permitting new data to be added along multiple dimensions. Support is provided for user-defined types and for reference types that are analogous to pointers. With its support for parallel I/O, *chunking (*described below), and data compression, HDF5 is especially appropriate for use in high-performance computing contexts.

The netCDF *classic* data model, used for all versions of netCDF before netCDF-4, has fewer primitive data types and abstractions, representing data as sets of

**Corresponding author address:* Russ Rew, Unidata/UCAR, PO Box 3000, Boulder, CO 80307, email: russ@unidata.ucar.edu. The National Science Foundation is Unidata's primary sponsor.

multidimensional arrays of primitive types with named variables, dimensions, and attributes (see Figure 1). Shared dimensions (an abstraction not previously supported by the HDF5 data model) explicitly represent variables defined on a common grid. Variables, dimensions, and attributes are global, but attributes may also be local to a variable. One dimension may be unlimited (dynamic), and data may be appended efficiently to all the variables that use this dimension.

HDF5 is used for many of NASA's Earth Observing System data products and in DOE's Advanced Simulation and Computing program. HDF5 has also been chosen for distributing and archiving NPP and NPOESS data products. NetCDF's simpler data model has proved adequate for representing gridded output from climate and forecast models (for example the IPCC Fourth Assessment model results) as well as archives for many kinds of observational data in the earth sciences. Recently, a netCDF-3 interface was added to ESRI's suite of GIS applications, enabling direct access to much atmospheric and oceanographic data within a widely used GIS context.

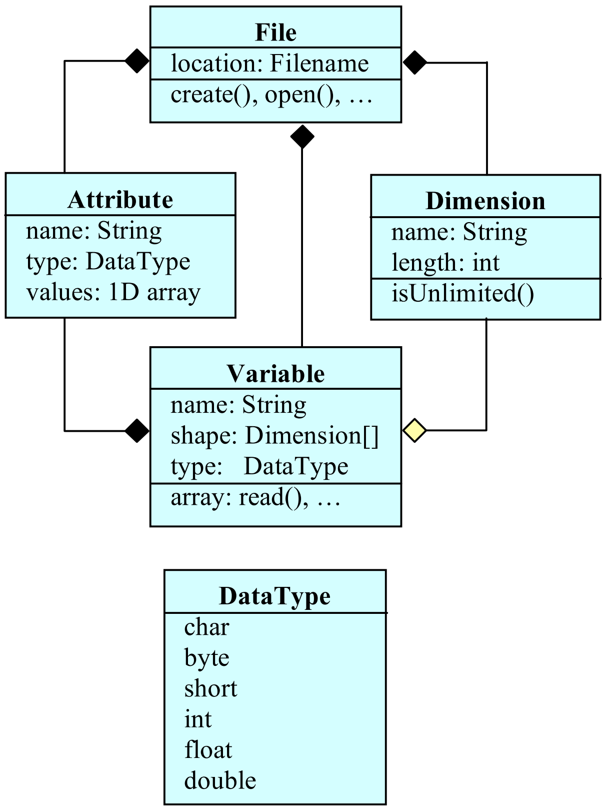

Figure 1. NetCDF *classic* data model: a file may have one or more variables, dimensions, and attributes. A variable may also have one or more attributes. Variables may share dimensions, indicating a common grid. One dimension may be dynamic (of unlimited length).

3. THE NEW NETCDF-4 DATA MODEL

The netCDF-4 data model adds support for multiple unlimited dimensions, new primitive types, user-defined types (compound, variable-length, enum, and opaque) and groups (see Figure 2). The new data model is, by intention, a restricted subset of the HDF5 data model. As described in Caron (2006), NetCDF, HDF5, and OPeNDAP developers have begun to discuss formalizing this intermediate Common Data Model, providing useful mappings among the three data models, and evolving the data models to mitigate differences and to make OPeNDAP the remote access protocol for netCDF-4 and netCDF-4 the persistence format for OPeNDAP. Agreement on such a Common Data Model could enhance interoperability for scientific data and applications, allowing data providers to structure their data in a way that would simplify access using any of HDF5, netCDF-4, or OPeNDAP.

3.1 Multiple Unlimited Dimensions

An important feature of the netCDF classic data model is the ability to efficiently append new data to variables in a netCDF file. This is implemented by specifying an *unlimited* dimension along which variables can grow. Time is often used for the unlimited dimension, allowing new time steps to be added for time-dependent variables.

The restriction to only a single unlimited dimension per file facilitates efficient access, but is also a significant limitation in the netCDF classic data model, because there is sometimes a need to allow data to grow along multiple dimensions. For example a data provider might want to add observational data for both new times and new observing locations, but if time and observing station are dimensions, then either the number of times or number of observing locations must be fixed in advance. Workarounds for this limitation have included specifying a maximum for all dimensions but one (which wastes space) or associating an artificial dimension with a tuple of desired dynamic dimensions (which obscures the natural multidimensional structure of the data). In netCDF-4, multiple unlimited dimensions are supported, so such workarounds are unnecessary.

3.2 New Primitive Data Types

In the netCDF classic data model, numeric data must be represented with only five primitive types, corresponding to the types for numeric data that could be represented portably using the XDR standard for external data representation: byte (8 bits), signed short (16 bits), signed int (32 bits), float (32 bits), or double (64 bits). The model also supports text strings as arrays of 8-bit characters.

The netCDF-4 data model adds support for 64-bit integers, unsigned integer types, and strings that need not be treated as just arrays of characters.

3.3 User-defined Types

Four kinds of user-defined data types available in the netCDF-4 data model are compound types, variable-length types, enumerations, and opaque types.

Compound types: User-defined structures in C make it easy to build up more complex and useful types from primitives, possibly of different types. But C structures cannot be written and read portably, because the padding and alignment of structure members of different types may vary from platform to platform. NetCDF-4 exploits HDF5 capabilities to support portable I/O for user-defined *compound types*, corresponding to C structures. (Although the concept is the same, we adopt the HDF5 terminology of "compound type" rather than "struct" to lessen the divergence of terminology between netCDF and HDF5.)

Compound types are useful for representing multiple parameter values at each grid point or at each time and space location for ungridded data. When a compound type is used, accessing all the information at a point requires reading only one variable, rather than reading multiple parameter values from multiple variables.

As in C, compound types may be elements of arrays, my include array members, and may be nested..

In HDF5, attributes may only be assigned to a whole compound type, not individual member variables. To accomplish assigning an attribute such as “units” to each member variable, create an attribute named “units” of compound type that has the same member variable names, and assign the appropriate units string to each member variable of the resulting units attribute. Such a mechanism may be generalized to assign an attribute to a subset of member variables, using identity between names of the member variables in the compound type and names of member variables in the associated attribute.

Variable-length types: The netCDF-4 data model supports variable-length vectors of any type. This permits "ragged arrays" where the length of each row varies. An example where this might be useful is soundings, where the data for each sounding is of variable length. This eliminates the need to declare a maximum number of observations per sounding.

A variable-length vector differs from a variable that uses an unlimited dimension, because the variable-length does not correspond to a named dimension that can be shared with other variables. A variable-length variable has a base type that may be of primitive type or of another user-defined type, such as a compound type. Using variable-length data types in languages that lack automatic memory management requires special memory allocation and deallocation procedures to prevent memory leaks.

Enumerations and Opaque Types: Other new user-defined types include enumerations and opaque types. Enumerations support the definition and use of named integer constants as values. Opaque types permit storing bland data as a named, fixed-size sequence of bytes.

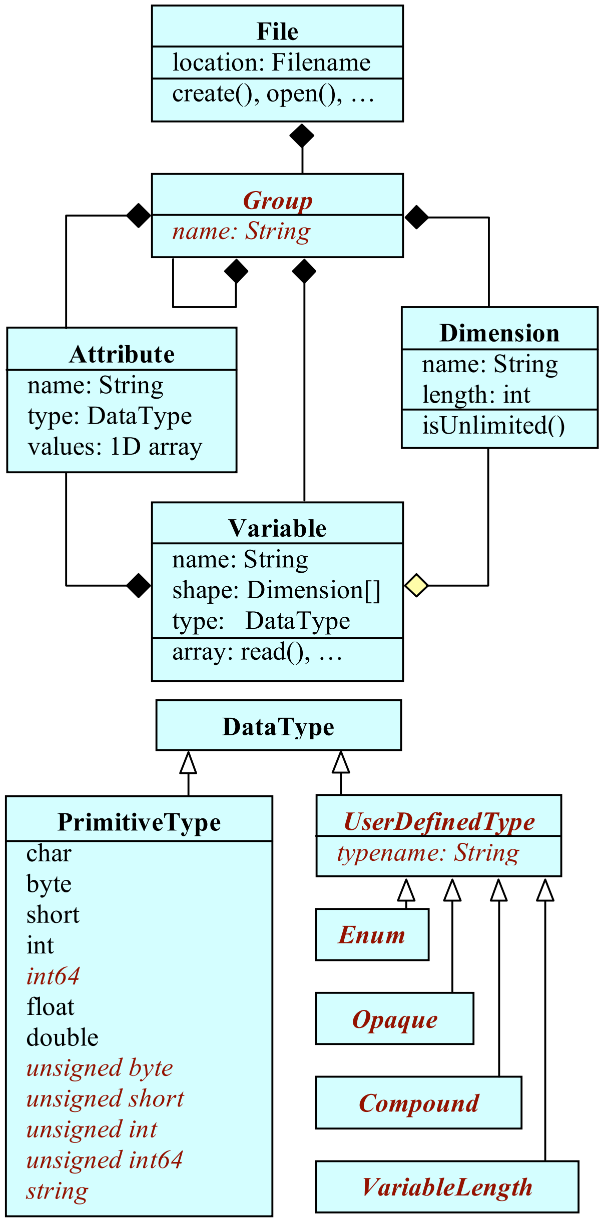

Figure 2. NetCDF-4 data model, added features marked in *italics*: a file has a top-level unnamed group. Each group may contain one or more named subgroups, variables, dimensions, and attributes. A variable may have one or more attributes. Variables may share dimensions, indicating a common grid. One or more dimensions may be dynamic (of unlimited length). User-defined compound types are available.

3.5 Groups

A flat file system with no directories may be adequate for hundreds of files, but it doesn't scale well for representing thousands of files that are more naturally grouped in nested directories. Such a hierarchical file system supports name scopes so that each directory may have its own "index.html" file, for example, without confusion or name clashes.

Analogously, name spaces and grouping can be useful for scalability in modeling complex simulations with multiple ensembles of outputs or large collections of observational data made possible with modern instruments and sensor networks.

Multiple attributes may share the same name (for example, "units"), since each variable establishes a scope for names of attributes attached to that variable. Providing scopes for variable names makes practical the use of mulitiple variables with the same name. An example where this might be useful is storing ensembles or model outputs run on different grids within the same file. In netCDF-4, a group is analogous to a directory in a modern file system, in that it serves as a container for other groups as well as for variables, dimensions, and attributes. Each group establishes a naming scope for the objects it contains.

The addition of groups to the netCDF classic model has been accomplished in a way that preserves simplicity and backward compatibility for files and programs that do not use or need groups. Every netCDF-4 file has a single unnamed top-level group that corresponds exactly to the single flat name space in netCDF-3 files. A netCDF-3 program that knows nothing about groups will function properly after recompiling and linking with the netCDF-4 library, and netCDF-3 data that does not use groups will still be accessible through the new interface. Thus use of groups is completely optional and transparent in situations where they are not needed.

Groups are one of the primary types of HDF5 objects, but netCDF-4 groups are intentionally restricted from the full generality of HDF5 groups: in netCDF-4, unlike HDF5, they form a strict hierarchy (tree), so that each group has a unique parent and a unique name.

There are various other uses for groups in the netCDF-4 data model:

- Groups can be useful to "factor out" shared information that should be stored only once in a file. For example, information about a grid common to ensemble model runs could be stored in a single parent group.

- Groups can directly represent common data, for example storing observations within each country in a group with the country's name.

- Information of limited interest can be stored out of the way in a group. For example, instrument calibration coefficients might be stored in a separate group for observed data, out of the way of users who don't care about calibration details.

- Groups may be useful for directly modeling some recursive data structures, such as nested meshes.

4. THE NETCDF-4 DATA FORMAT

The netCDF-4 data format includes support for previous netCDF format variants for compatibility with existing data. New features supported by the format include dynamic schema changes, chunking, "reader makes right" numeric conversions, and use of the Universal Character Set in names.

4.1 Format Variants

Before 2005, there was only one netCDF file format used by all versions of netCDF. Release 3.6 provided support for large files by allowing 64-bit offsets in the format where previously only 32-bit offsets had been permitted. This necessitated distinguishing between the formats: the original format is now referred to as the netCDF *classic* format, and the second variant (supported in netCDF version 3.6 and later) as the netCDF *64-bit offset* format.

The netCDF library detects which variant of the format is used for each file when opening it for reading or writing, so it is not necessary to know which variant is used. Of course, versions of the library earlier than 3.6 cannot access data in the 64-bit offset format, so conservative data providers will preserve interoperability by avoiding use of the 64-bit format variant until all the applications used to access it have been relinked with an upgraded version of the library.

With netCDF-4 and later, there is a third format variant based on HDF5. This variant is referred to as the *netCDF-4 format*, referring to an HDF5 file created through the netCDF-4 library interface. Again, the library automatically detects which variant of the format is used for each file when it is opened for reading or writing, so it is not necessary for users to know which variant of the format is used. However, new features of the enhanced netCDF-4 data model, such as groups and compound types, cannot be added to netCDF-3 files. If you open an existing netCDF-3 file and try to make use of any feature specific to netCDF-4, such as creating a group, an error will be returned and the file left unchanged, since such operations are not supported for netCDF-3 files.

For convenience, we have introduced a fourth format variant: *netCDF-4 classic*. This refers to a file that uses the HDF5 storage format, but no features specific to netCDF-4 such as groups or compound types. Such files can be accessed, manipulated, and visualized by netCDF-3 applications that are merely relinked to the netCDF-4 library. These files are a kind of hybrid that can be explicitly created and manipulated with the netCDF-3 library interfaces and applications, but that are HDF5 files underneath. This format is preserved by the interface, because any attempt to add a netCDF-4-specific feature to such a file will result in an error. As described below, there are potential performance implications in just using the netCDF-3 interface with the HDF5 storage format.

The version of the format will always be detected on reading and preserved by the library on writing, so it is not possible to open a file and convert it to a different format variant just by writing into it. To change the format of a file will require use of a utility that copies the file (for example the NCO ncks utility). Future versions of netCDF software will continue to support reading and writing files using the classic (32-bit offset) format, the 64-bit offset format, and both netCDF-4 format variants. There is no need to convert existing archives from netCDF-3 to netCDF-4 formats.

4.2 Dynamic schema changes

In the netCDF-3 classic or 64-bit offset formats, the schema information (metadata) about names, types, and shapes of variables, dimensions, attributes, and attribute values is stored in a header region at the beginning of the file. This imposes constraints on schema changes, making it necessary to explicitly enter "define mode" to add new variables, dimensions, or attributes. If you neglect to reserve extra space in the header when a file is created, additions to the schema may result in copying all the data.

In netCDF-4 using HDF5 storage, information about data objects is no longer stored in a single contiguous header region, so these constraints are no longer necessary. New variables, dimensions, and attributes can be added dynamically, with no performance penalty. Functions for entering and leaving define mode are still supported, but will not result in expensive copying of data. This is one of the potential performance benefits with netCDF-4 classic files that use none of the netcdf-4-specific data model features.

4.3 Chunking

Variables that make use of an unlimited (dynamic) dimension or that use compression are stored using *chunking*. This refers to multidimensional tiling of values so that there need not be a favored order of access to values. This form of storage is especially appropriate when it can be anticipated that a variable's data will be accessed along more than one dimension, for example by row and by column. On the other hand, chunking can result in a performance penalty compared to use of netCDF-3 storage if data is always read in the same order in which it was written. Tuning the default chunking parameters provides new opportunities for optimizing I/O performance for different patterns of access.

4.4 Reader-makes-right numeric conversions

NetCDF-3 stores data in a portable canonical form using the XDR standard. On some platforms, this means data conversions in the form of byte swapping occur both on writing and reading data, leading to noticeable inefficiencies in copying large volumes of data from one file to another.

NetCDF-4 takes advantage of HDF5's support for native as well as XDR representations to support a "reader makes right" approach that avoids unnecessary conversions or byte swapping in most cases. Data is written using the native representation of the writing platform and converted on-the fly to the native representation of the reading platform only when the data is accessed for reading.

4.5 Universal Character Set in names

An HDF5 enhancement added for support of netCDF-4 is the ability to specify a sequence of characters from the Universal Character Set (Unicode / ISO 10646) for names of variables, dimensions, attributes, and groups. NetCDF-4 uses UTF-8 encoding, so that names that use US-ASCII characters are unchanged.

5. OTHER FEATURES OF NETCDF-4

Other features in the initial netCDF-4 reference release include use of autotools and refactored documentation.

5.1 Autotools

With a transition to use of GNU autotools (autoconf, automake, libtool), building netCDF from source on new platforms should be somewhat easier than with netCDF-3. Another advantage of autotools is the ability to create dynamic libraries instead of static libraries.

5.2 Refactored documentation

We have created a new language-neutral netCDF user's guide and four language-specific user's guides for C, Fortran-77, Fortran-90, and C++ interfaces. All the documentation is available from the Unidata netCDF web site.

5.3 Programming Language Interfaces

In the initial release, only the C interface will support all the features of the netCDF-4 data model. Implementation and testing of Java and Fortran interfaces are underway and it is anticipated that these will be available soon after the 4.0 release of netCDF. The Fortran and C++ interfaces included in the release will support access via the netCDF-3 interfaces to netCDF classic, 64-bit offset, or netCDF-4 classic files.

The Java interface implements higher layers of the Common Data Model that incorporate generalized coordinate systems and prototype support for higher-level scientific data types such as georeferenced grids and trajectory data. Work is underway to add support for these higher-level abstractions to the netCDF-4 C interface. The Java netCDF library also provides client access to data on HTTP and OPeNDAP servers and an I/O framework providing netCDF interfaces to data in other formats including GRIB, HDF5, GINI, and NEXRAD, by reading data in these other formats into the Common Data Model. Another advanced feature of the Java interface is support for access through virtual datasets represented in NcML, an XML dialect for netCDF.

Providing access to netCDF-4-specific features from other programming languages including C++, Perl, Python, and Ruby will depend on efforts in the netCDF community from developers who have generously created and maintained these language interfaces in the past.

6. RECOMMENDATIONS

Because it is currently only available in test form as a beta release, there is little experience with netCDF-4 on which to base recommendations for data providers, application developers, and users. Nevertheless, we can provide some guidance for those considering using netCDF-4, based on intended use of the new features and some limited testing.

6.1 Classic chicken and egg situation

Adapting an existing application to correctly handle netCDF-4 features such as nested groups and nested compound types will require some effort. If there are no data providers making use of these features yet (because very few applications exist that handle them), then there will be no incentive to expend the resources to adapt applications. Ultimately, data and applications will only be adapted to netCDF-4 if a critical mass of data and useful applications exist. Some data providers may decide that netCDF-4 is not enough simpler than HDF5 to justify using netCDF-4 rather than HDF5. Similarly, application developers may decide that if they need to modify their applications to fully support netCDF-4, they might consider expending the extra effort required to provide full support for HDF5 as well.

Data in netCDF-4-classic files provide some of the benefits of netCDF-4, such as efficient dynamic schema changes made possible with HDF5 storage, but few of the costs of supporting the enhanced data model.

6.2 Applications

It will take some time for analysis and visualization applications to be able to deal with new netCDF-4 features such as groups and user-defined compound types. Our recommendation to developers of such applications is to first try just relinking the existing application with the new netCDF-4 library. This should then support access to classic, 64-bit offset, and netCDF-4 classic files with no changes needed to the application. Adapting to nested structures and groups may be as simple as flattening the name space by using "." and "/" characters in names.

The generic utility applications, ncdump and ncgen, that are part of the standard netCDF distribution are still (as of this writing) being adapted to netcdf-4-specific features. Other generic software such as the NCO utilities are also being adapted to netCDF-4 and may eventually show significant performance gains over the corresponding netCDF-3 utilities, especially on parallel platforms.

6.3 Data Providers

Before making use of netCDF-4-specific features, ensure this will not significantly restrict data access by users of applications that cannot yet handle such features. If you supply the applications, this may not be a problem, but if off-the-shelf applications such as MATLAB, IDL, or ArcGIS are used to access the data, stick with netCDF-3 or netCDF-4-classic until the relevant applications have been adapted and users have been given sufficient time to update their versions of the applications.

6.4 Distributed Access

Those who make data available for remote access through protocols such as OPeNDAP should also consider carefully when to begin using netCDF-4-specific features, because such features may require support in both the server and client. OPeNDAP already has a head start on providing such support, because its protocol was designed to serve data represented in various formats, including netCDF and HDF5. The THREDDS Data Server includes an OPeNDAP server that is already adapted to the Common Data Model implemented in netCDF-Java, and clients built on netCDF-Java can also handle this data model. Future efforts to provide Common Data Model support in other OPeNDAP servers and in the OPeNDAP protocol provide an opportunity to further enhance interoperability in building distributed data services.

6.5 Interoperability with HDF5

It is possible to create data using the HDF5 interfaces that cannot be accessed through netCDF-4 interfaces, because the netCDF-4 model is intentionally simpler. For example, with HDF5 it is possible to create a group containment cycle in which each of two groups contain the other. As another example, HDF5 has a reference data type, used for indirection much like pointers in C, but HDF5 references are not supported in netCDF-4.

Nevertheless, it is practical to create HDF5 files that are accessible through the netCDF-4 interface, if the files are restricted to use only features supported by netCDF-4 and if the new HDF5 “dimension scales” feature is used to model shared dimensions. Following this advice is equivalent to using the Common Data Model, which is intended to be a more interoperable subset of data types and structures that will map easily to several interfaces. The netCDF-4 documentation accompanying the first release will contain more guidance about use of the Common Data Model.

6.6 Conventions and Recommended Practices

Authors and maintainers of netCDF conventions, such as the widely adopted CF conventions, are advised to be very conservative in making use of new netCDF-4 features. The benefits of capturing more meaning in the data by using compound types, for example, may be outweighed by the initial paucity of analysis and visualization applications that can handle these data types. Support for general coordinate systems is part of the Common Data Model that will eventually be supported by a layer above netCDF-4, but that layer will not be part of the initial 4.0 release. When the coordinate support is part of the software, it will no longer be necessary to encode coordinates in conventions, but such coordinate conventions must still be supported for existing data archives. Hence, existing conventions should continue to support netCDF-3 data.

6.6 Parallel I/O Users

If you need to make use of parallel netCDF I/O for performance reasons, you currently have a choice:

- Use pnetCDF (an implementation developed by a group of researchers from Argonne and Northwestern using MPI-IO directly but not HDF5), supporting classic and 64-bit format variants from either C or Fortran-90.
- Use the parallel I/O interfaces that are part of netCDF-4, which may be used with HDF5-supported features, but are currently only available from C.

It is premature to offer much additional guidance at this early point, since we lack experience in adapting and tuning models to use either of these parallel I/O facilities. Both will benefit from user feedback as a result of additional use. Ideally, the same interface should be available for either package, but that is not yet the case. Developers should realize that use of any netCDF-4 specific features may prevent use of pnetCDF, so benchmarking these two approaches might be advised before committing to netCDF-4 specific features.

8. BACKWARD COMPATIBILITY COMMITMENT

For scientific data important to future generations, preserving backward compatibility of data access is sacrosanct. Unidata has a strong commitment to ensure that access to all earlier forms of netCDF data will be supported by current and future versions of the software. As far as practical, we are also committed to maintaining the core APIs (C, Java, Fortran), so that programs that currently work with netCDF data will continue to work with existing data when recompiled and relinked with new versions of the libraries.

NetCDF and HDF5 already share with XML a kind of "forward compatibility," in which data objects that are not recognized are ignored. This is why adding a new variable to netCDF files typically has no effect on programs that previously handled files lacking that variable. However, it is unfortunately not possible to ensure that existing applications and utilities, recompiled and relinked but not otherwise modified, can access new data that makes use of added features of the data model, such as groups and compound types.

9. STATUS

The new netCDF-4 software is currently available only in alpha release form, because it depends on HDF5 version 1.8 enhancements that have not yet been officially released. Java and Fortran interfaces for netCDF-4-specific features are still under development. The ncdump utility handles netCDF-4 features and can produce output in NcML, but the ncgen utility that performs the reverse transformation is still under development for netCDF-4-specific features.

The release of HDF5 version 1.8 is expected in the first half of 2006, and the first release of netCDF-4 is anticipated soon after that.

ACKNOWLEDGMENTS

This work was supported by the NASA Earth Science Technology Office under NASA award AIST-02-0071. We would also like to acknowledge collaboration and development efforts of Mike Folk, Quincey Koziol, Robert E. McGrath, Elena Pourmal, and Muqun (Kent) Yang at NCSA and Greg Sjaardema at Sandia laboratories.

REFERENCES:

Caron, J., E. Davis, Y. Ho, R. Kambic, 2006: Unidata’s THREDDS Data Server, 22nd International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology, Atlanta, Georgia, Am. Meteor. Soc.

Gray, J., D. T. Liu, M. A. Nieto-Santisteban, Al. S. Szalay, G. Heber, D. DeWitt, 2005: Scientific Data Management in the Coming Decade. *Cyberinfrastructure Technology Watch Quarterly*, **1**, Number 2.

Li, J., W. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, M. Zingale, 2003: Parallel netCDF: A High-Performance Scientific I/O Interface. *SC2003*, Phoenix, Arizona, ACM.

NCO web site: http://nco.sourceforge.net/

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

Unidata's Common Data Model site: http://www.unidata.ucar.edu/software/netcdf/CDM/

Unidata's netCDF web site: http://www.unidata.ucar.edu/software/netcdf/