Introduction to NetCDF

Russ Rew, UCAR Unidata
ICTP Advanced School on High Performance and Grid Computing
13 April 2011
Overview

• Background
• What is netCDF?
• Data models and formats
• Utilities: ncdump, nctgen, nccopy
• Exercises
  • Application Programming Interfaces (API’s)
  • Remote access and OPeNDAP
  • Chunking and compression
  • Parallel I/O
Background: What is Unidata?

- Where netCDF is developed and maintained
- Funded primarily by US National Science Foundation through UCAR
- Staff of about 22, including 13 developers
- Mission: data services, tools, and community leadership to advance Earth system science, enhance educational opportunities, and broaden participation
- Open source software for data access and distribution, analysis and visualization, community advocacy, workshops, and software support
What is netCDF?
NetCDF History

1988: NetCDF developed at Unidata
1991: NetCDF v2.0 new API
1996: NetCDF v3.0 new API
2004: NetCDF v3.6.0 64-bit offset format
2008: NetCDF v4.0 uses HDF5
2010: NetCDF v4.1.1 openslap/HDF4/pnetcdf
NetCDF: not just a format

*network* Common *Data* Form

- **Data model** for scientific data and metadata
  - Widely used in ocean, climate, atmospheric science
  - Used in some other disciplines: molecular dynamics, neuro-imaging, fusion research ...

- **File format** for portable data
  - Array-oriented scientific data and metadata
  - NetCDF data is self-describing, portable, direct access, appendable, networkable, extensible, sharable, archivable

- **Application programming interfaces (APIs)**
  - C, Java, C++, Fortran (Developed and supported by UCAR / Unidata)
  - Python, Ruby, Perl, MATLAB, IDL, ... (3rd party APIs)

*Together, the data model, file format, and APIs support the creation, access, and sharing of scientific data*
Infrastructure for sharing scientific data

- Applications depend on lower layers
- Sharing requires agreements
  - formats
  - protocols
  - conventions
- Data needs metadata
- Is all this infrastructure really necessary?
NetCDF infrastructure

- Provides format and library for netCDF data model
- Endorsed by several standards bodies
- Active conventions communities
- OPeNDAP protocol
- Several servers for remote data access
- Many open source and commercial utilities and applications

April 14, 2011

NetCDF infrastructure

- Analysis & visualization tools: IDV, NCO, NCL, CDO, MATLAB, IDL, ...
- Utilities, applications: open source and commercial
- Servers, clients, services: THREDDS, OPeNDAP, ...
- Libraries & APIs: C, Java, Fortran, Perl, Python, ...
- Network protocols: OPeNDAP, cdmremote, ncstream
- Community conventions: Climate and Forecast (CF), ...
- Standards: NASA, OGC, FGDC
- Data models: netCDF classic & netCDF-4 enhanced
- Formats: classic, 64-bit offset, netCDF-4, classic model

April 12, 2011

Intro to netCDF
Development Milestones

1989: portable, self-describing data format, data model, and software for creation, access, and sharing of scientific data

1990's: growth of use in ocean and climate models, 3rd-party software support (NCO, NCL, IDL, MATLAB)

2002: Java version with OPeNDAP client support

2003: NASA funded netCDF-4/HDF5 project; Argonne/Northwestern parallel netCDF

2004: netCDF-Java plug ins for reading other formats, NcML aggregation service

2007: netCDF-Java Common Data Model (access to other formats through netCDF interface)

2008: netCDF-4 C and Fortran library with HDF5 integration, enhanced data model, parallel I/O

2009: “netCDF classic format” becomes NASA standard

2010: version 4.1.1 - OPeNDAP client support for C/Fortran libraries; udunits, CF library support; pnetcdf, HDF4 access, FGDC standardization

2011: version 4.1.2 – speedups, refactoring, bug fixes, new functions, nccopy supports compression and chunking, OGC standardization, Windows version
Who uses netCDF?

- Climate modelers
  - Program for Climate Model Diagnosis and Intercomparison (PCMDI)
  - Earth Systems Grid
- Ocean and atmospheric sciences
  - Forecast models
  - Atmospheric chemistry
- Neuroimaging
  - MINC - Medical Image NetCDF
  - NiBabel
- Fusion research
  - Culham Centre for Fusion Energy (C++ API for netCDF-4)
- Molecular dynamics simulations (e.g. AMBER)
NetCDF standards endorsements

- **2009-02-05**: NASA Earth Science Data Systems (ESDS) Standards Process Group endorsed **netCDF classic and 64-bit offset formats** as appropriate for NASA Earth Science data.
- **2010-03-01**: Integrated Ocean Observing System (IOOS) Data Management and Communications (DMAC) Subsystem endorsed **netCDF with Climate and Forecast (CF) conventions** as a preferred data format.
- **2010-09-27**: Steering Committee of the US Federal Geographic Data Committee (FGDC) officially endorsed **netCDF** as a Common Encoding Standard.
- **2011-03-07**: Open Geospatial Consortium (OGC) approved "**OGC Network Common Data Form (NetCDF) Core Encoding Standard version 1.0**" as a new OGC standard. Thanks to Dr. Ben Domenico (Unidata) and Dr. Stefano Nativi (University of Florence, CNR-IMAA).
Data models and formats
What is a data model?

• Formally:
  – A collection of **data objects** such as lists, tables, relations, ...
  – A collection of **operations** that can be applied to the objects such as retrieval, update, subsetting, averaging, ...
  – A collection of integrity **rules** that define the legal states (set of values) or changes of state (operations on values)

• We won't be that formal, will just draw pictures and wave our hands
  – to describe what netCDF data objects are and what you can do with them
  – independent from data format details
  – independent from programming language
Data model examples

• Relational data model
  - Concepts: tables, rows, columns, types
  - Operations: create, replace, update, delete, find, index, ...
  - Rules: normal forms, integrity constraints

<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
<th>Role</th>
<th>ID #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td><a href="mailto:alice@univ.edu">alice@univ.edu</a></td>
<td>Student</td>
<td>123</td>
</tr>
<tr>
<td>Bob</td>
<td><a href="mailto:bob@abc.com">bob@abc.com</a></td>
<td>Student</td>
<td>456</td>
</tr>
</tbody>
</table>

• Geospatial information system data model
  - Concepts: locations, lines, polygons, features, surfaces
  - Operations: create, replace, update, delete, intersects
  - Rules: adjacent features share a common edge, ...
The netCDF "classic" data model, in words

- A netCDF file has named **variables, attributes, and dimensions**.
- Variables are for data, attributes are for metadata (data about data)
- Dimensions are for specifying shapes of variables
- Attributes may apply to a whole file or to a single variable
- Variables may share dimensions, indicating a common grid.
- One dimension may be of unlimited length.
- Each variable or attribute has 1 of 6 types: char, byte, short, int, float, double
### Example of netCDF classic data model

<table>
<thead>
<tr>
<th>Dimensions</th>
<th>Variables</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>lon</td>
<td>lon</td>
<td>units: &quot;degrees_east&quot;</td>
</tr>
<tr>
<td>lat</td>
<td>lat</td>
<td>units: &quot;degrees_north&quot;</td>
</tr>
<tr>
<td>time</td>
<td>time</td>
<td>units: &quot;days since 1901-1-1&quot;</td>
</tr>
<tr>
<td></td>
<td>IR_flux</td>
<td>units: &quot;W m-2&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_Fill_value: -999</td>
</tr>
<tr>
<td></td>
<td>snow_cover</td>
<td>units: &quot;kg m-2&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_Fill_value: -1.0</td>
</tr>
<tr>
<td></td>
<td>snow_cover</td>
<td>standard_name: &quot;surface_snow_amount&quot;</td>
</tr>
</tbody>
</table>

- **Title**: "Global monthly surface averages"
- **Units**: "W m-2" for IR_flux
- **Fill Value**: -999 for IR_flux, -1.0 for snow_cover
- **Standard Name**: "downwelling_longwave_flux_in_air" for IR_flux, "surface_snow_amount" for snow_cover
NetCDF Data has:

- **Variables** (e.g., temperature, pressure)
- **Attributes** (e.g., units)
- **Dimensions** (e.g., lat, lon, level, time)

Each variable has:
- Name, shape, type, attributes
- N-dimensional array of values

Each attribute has:
- Name, type, value(s)

Each dimension has:
- Name, length

Variables *may share* dimensions:
- Represents shared coordinates, grids

Variable and attribute values are of type:
- Character: arrays of **char** for text
Common Data Language (CDL)

- Text notation for netCDF metadata and data

```c
netcdf snow{
  // example of CDL notation
  dimensions:
    lon= 9 ;
    lat= 7 ;
    time = unlimited ;  // 3 currently
  variables:
    float IR_flux(lon, lat) ;
    IR_flux:units = "W m-2" ;
    IR_flux:_Fill_value = -999 ;
    IR_flux:standard_name= "downwelling_longwave_flux_in_air";
    float snow_cover(time, lon, lat) ;
    snow_cover:units = "kg m-2" ;
...
  // global attributes
    :title = "simple example, lacks some conventions" ;
  data:
    IR_flux = 200, 201, ... ;
    snow_cover = 0.1, 0.2, 0.0, ... ;
}
```
NetCDF format characteristics

- **Self-Describing:** A netCDF file includes metadata as well as data: names of variables, data locations in time and space, units of measure, and other useful information.
- **Portable:** Data written on one platform can be read on other platforms.
- **Direct-access:** A small subset of a large dataset may be accessed efficiently, without first reading through all the preceding data.
- **Appendable:** Data may be efficiently added to a netCDF file without copying the dataset or redefining its structure.
- **Extensible:** Adding new dimensions, variables, or attributes to netCDF files does not require changes to existing programs that read the files.
- **Sharable:** One writer and multiple readers may simultaneously access the same netCDF file. With Parallel netCDF, multiple writers may efficiently and concurrently write into the same netCDF file.
- **Archivable:** Access to all earlier forms of netCDF data will be supported by current and future versions of the software.
- **Networkable:** The netCDF library provides client access to structured data on remote servers through OPeNDAP protocols.
NetCDF classic data model

Strengths

✓ Data model simple to understand and explain
✓ Efficient implementation freely available
✓ Generic applications easy to develop
✓ Representation good for gridded multidimensional data
✓ Shared dimensions useful for coordinate systems

Limitations

▪ Small set of primitive types
▪ Flat data model limited to multidimensional arrays, lists, (name, value) pairs
▪ Flat name space not ideal for organizing many data objects
▪ Lacks nested structures, variable-length types, enumerations
NetCDF classic format

Strengths

✓ Simple to understand and explain
✓ Supported by many applications
✓ Standard used in many archives, data projects
✓ Mature conventions and best practices have evolved

Limitations

▪ No support for efficient compression
▪ Only one dimension can grow efficiently
▪ Portable representation favors big-endian platforms
▪ Schema changes can be costly
A file has a top-level unnamed group. Each group may contain one or more named subgroups, user-defined types, variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One or more dimensions may be of unlimited length.

Variables and attributes have one of twelve primitive data types or one of four user-defined types.
NetCDF enhanced data model

Strengths

- Simpler than HDF5, with similar representational power
- Adds shared dimensions to HDF5 data model
- Continues support for existing data, software, and conventions
- Eliminates netCDF classic model limitations
- Provides nested structures: hierarchical groups, recursive data types
- Independent features permit incremental adaptation, adoption

On the other hand

- More complex than classic data model
- More effort required to develop general tools and applications
- Not yet widely adopted
- Hence, no comprehensive best practices and conventions yet
NetCDF-4 format

- Uses HDF5 as a storage layer
- Provides performance advantages of HDF5
  - Compression
  - Chunking
  - Parallel I/O
  - Efficient schema changes
- Useful for larger or more complex datasets
- Suitable for high-performance computing
To ensure future access to existing data archives, Unidata is committed to compatibility of:

- **Data access**: new versions of netCDF software will provide read and write access to previously stored netCDF data.
- **Programming interfaces**: C and Fortran programs using documented netCDF interfaces from previous versions will work without change with new versions of netCDF software.
- **Future versions**: Unidata will continue to support both data access compatibility and program compatibility in future netCDF releases.
NetCDF-4 classic-model: a transitional format

- NetCDF-3
  - Compatible with existing applications
  - Simplest data model and API

- NetCDF-4 classic model
  - Uses classic API for compatibility
  - Uses netCDF-4/HDF5 storage for compression, chunking, performance
  - To use, just recompile, relink

- NetCDF-4
  - Not compatible with some existing applications
  - Enhanced data model and API, more complex, powerful
Common Data Language (CDL), again

- Text notation for netCDF metadata and data

```c
netcdf example {
  // example of CDL notation
  dimensions:
    x = 2;
    y = 8;
  variables:
    float rh(x, y);
    rh:units = "percent";
    rh:long_name = "relative humidity";
  // global attributes
  :title = "simple example, lacks some conventions";
  data:
    rh =
      2, 3, 5, 7, 11, 13, 17, 19,
      23, 29, 31, 37, 41, 43, 47, 53;
}
```

- A netCDF file with 2 dimensions (x and y), 1 variable (rh), 2 variable attributes (units and long_name), 1 global attribute (title), and some data values.
Utility programs for netCDF to/from CDL

$ ncdump -h co2.nc

```plaintext
netcdf co2 {
  dimensions:
    T = 456 ;
  variables:
    float T(T) ;
      T:units = "months since 1960-01-01" ;
    float co2(T) ;
      co2:long_name = "CO2 concentration by volume" ;
      co2:units = "1.0e-6" ;
      co2:_FillValue = -99.99f ;

  // global attributes:
}
```

- "-h" is for "header only", just outputs metadata, no data
- "-c" outputs header and coordinate variable data
- The `ncgen` utility does the opposite of `ncdump`, converts CDL to netCDF
• Coordinate variables
  – have same name as a dimension
  – contain coordinate values for the dimension
  – should be one-dimensional
  – should contain no missing values
  – should have values that are strictly increasing or strictly decreasing

Dimensions:

Coordinate variables:
## Variables vs. Attributes

### Variables
- Intended for data
- Can hold arrays too large for memory
- May be multidimensional
- Support partial access (only a subset of values)
- Values may be changed, more data may be appended
- May have attributes
- Shape specified with netCDF dimensions
- Not read until accessed

### Attributes
- Intended for metadata
- For small units of information that fit in memory
- For single values, strings, or small 1-D arrays
- Atomic access, must be written or read all at once
- Values typically don't change after creation
- An attribute may not have attributes
- Read when file opened

---

April 12, 2011

Intro to netCDF
Utilities: ncdump, ncgen, nccopy
The ncdump utility

- Converts netCDF data to human-readable text form
- Useful for browsing data files

```
ncdump [-c|-h] [-v ...] [-k] file
```

- `-c` Coordinate variable data and header info
- `-h` Header information only, no data
- `-v var1[,,...]` Data for variable(s) var1,... Only
- `-k` Output kind of netCDF file
The nctgen utility

- Converts netCDF CDL to a binary netCDF file or a program
- Useful for generating netCDF files without programming

```
ncgen [-b] [-k file_format] [-l language] cdl_file
  [-b]  binary output as a netCDF file
  [-k]  kind of netCDF file
  [-l c|f77|java]  language of program generated to standard output
```
Using ncdump and nccgen together

- The ncdump and nccgen utilities are inverses of each other:

  ![Diagram showing the relationship between netCDF data, ncdump, CDL text, and nccgen.]

- To add, delete, or change metadata or data in a netCDF file:

  ![Diagram showing the process: netCDF data is converted to CDL text using ncdump, then edited using an editor, and finally converted back to netCDF data using nccgen.]

- This is not very practical for huge files or a large number of files. In that case you may need to write a program, using the netCDF library.
More of using `ncgen` and `ncdump` together

- To create a new netCDF file with lots of metadata:

  - Insert easy "var_put(...)") calls to the netCDF library for the data writing part of the task
  - Compile and run the program to create desired netCDF file
  - Use `ncdump` to verify the desired file is created.
The nccopy utility

- Copies or converts and optionally compresses netCDF data
- Can also "re-chunk" data for more optimized access

```
nccopy [-k n] [-d n] [-s] [-c chunkspec] [-u] [-m n] infile outfile
```

- `-k n` specify kind of netCDF output, default same as input
  - 1 classic, 2 64-bit offset, 3 netCDF-4, 4 netCDF-4 classic model
- `-d n` set compression level, default same as input (0=none 9=max)
- `-s` add shuffle option to deflation compression
- `-c chunkspec` specify chunking for dimensions
- `-u` convert unlimited dimensions to fixed-size in output

`infile` name of netCDF input

`outfile` name for netCDF output
Using nccopy

- Compress netCDF data to a specified level, compressing each variable separately

```
netCDF data
```
```
nccopy -d1
```
```
netCDF data
```

- Convert a netCDF-4 classic model file to a netCDF-3 classic file, uncompressing any compressed variables.

```
netCDF data
```
```
nccopy -k1
```
```
netCDF data
```
The nc-config utility

- nc-config reports on version installed and assists with setting compiler and linker flags for applications

- To compile and link a C application and a Fortran application, using nc-config:

  $ cc `nc-config --cflags` myapp.c -o myapp `nc-config --libs`
  $ f95 `nc-config --fflags` yrapp.f -o yrapp `nc-config --flibs`

- To report all the features of the netCDF installation you are using (support for remote access clients, netCDF-4, parallel IO, HDF4 access support, etc.)

  nc-config --all
• Many other useful netCDF utilities developed by third parties are available, including both open source (e.g. NCL, NCO, CDO) and commercial (MATLAB, IDL, ARCInfo) packages.

• For information about over 100 such packages, consult Unidata's Software for Manipulating or Displaying NetCDF Data or ARM's list of data tools, which includes some downloadable binaries.
More information

- Online netCDF workshop
  [www.unidata.ucar.edu/netcdf/workshops/2010/](www.unidata.ucar.edu/netcdf/workshops/2010/)

- Software support:
  support-netcdf@unidata.ucar.edu

- Software tools for manipulating or displaying netCDF data:
  [www.unidata.ucar.edu/netcdf/docs/software.html](www.unidata.ucar.edu/netcdf/docs/software.html)
Questions?
Exercises
Try ncdump utility

- Look at just the header information (also called the schema or metadata):
  $ ncdump -h mslp.nc
- Store entire CDL output for use later in ngen exercises
  $ ncdump mslp.nc > mslp.cdl
- Look at header and coordinate information, but not the data:
  $ ncdump -c mslp.nc
- Look at all the data in the file, in addition to the metadata:
  $ ncdump mslp.nc
- Look at a subset of the data by specifying one or more variables:
  $ ncdump -v lat,time mslp.nc
- Look at times in human-readable form:
  $ ncdump -t -v lat,time mslp.nc
- Look at what kind of netCDF data is in the file (classic, 64-bit offset, netCDF-4, or netCDF-4 classic model):
  $ ncdump -k mslp.nc
Try ngen utility

- Check a CDL file for any syntax errors:
  
  ```
  $ ngen mslp.cdl
  ```

- Edit mslp.cdl and change something (name of variable, data value, etc.).

- Use ngen to generate new binary netCDF file (my.nc) with your changes:
  
  ```
  $ ngen -o my.nc mslp.cdl
  
  $ ncdump my.nc
  ```

- Generate a C, Fortran, or Java program which, when compiled and run, will create the binary netCDF file corresponding to the CDL text file.
  
  ```
  $ ngen -l c mslp.cdl > mslp.c
  
  $ ngen -l f77 mslp.cdl > mslp.f77
  
  $ ngen -l java mslp.cdl > mslp.java
  ```

- Try compiling and running one of those programs. You will need to know where the netCDF library is to link your program.
Try nccopy utility

(Requires netCDF version 4.1.2 or later)

- Compress variables in a test file, test.nc, by using nccopy. Then check if adding the shuffling option improves compression:

  $ nccopy -d1 test.nc testd1.nc  # compress data, level 1
  $ nccopy -d1 -s test.nc testd1s.nc  # shuffle and compress data
  $ ls -l test.nc testd1.nc testd1s.nc  # check results

- Download just the variable named "Total_precipitation" and relevant metadata from an OPeNDAP server dataset into a netCDF file named precip.nc

  $ nccopy \
     'http://motherlode.ucar.edu/thredds/dodsC/fmrc/NCEP/GFS/\n      Hawaii_160km/NCEP-GFS-Hawaii_160km_best.ncd?Total_precipitation' \n     precip.nc
Try remote access

(Requires netCDF built with DAP support, vers. 4.1.1 or later)

• Look at what's in some remote data from an OPeNDAP server:

  $ ncdump -c http://test.opendap.org/opendap/data/nc/3fnoc.nc

• Copy 3 coordinate variables out of the file

  $ nccopy "http://test.opendap.org/opendap/data/nc/3fnoc.nc?lat,lon,time" coords.nc

• Copy subarray of variable $u$ out of the file into a new netCDF file

  $ nccopy "http://test.opendap.org/opendap/data/nc/3fnoc.nc?u[2:5][0:4][0:5]" u.nc
  $ ncdump u.nc