A primer on netCDF Format

G. Giuliani International Centre for Theorethical Physics - Trieste Earth System Physics Section

ICTP - Earth System Physics Section

July 26, 2011

Research and Data

- What Science is About
  - See phenomena and/or build experiments
  - Collect Information as numeric values
  - Build knowledge on it with analysis

Research and Data

- What Science is About
  - See phenomena and/or build experiments
  - Collect Information as numeric values
  - Build knowledge on it with analysis
  - Or mix the previous three points

Research and Data

- What Science is About
  - See phenomena and/or build experiments
  - Collect Information as numeric values
  - Build knowledge on it with analysis
  - Or mix the previous three points
- Data Lifecycle
  - Science is knowledge AND information
  - New knowledge needs to validate against existing information
  - New knowledge may require reshape of information

Research and Data

- What Science is About
  - See phenomena and/or build experiments
  - Collect Information as numeric values
  - Build knowledge on it with analysis
  - Or mix the previous three points
- Data Lifecycle
  - Science is knowledge AND information
  - New knowledge needs to validate against existing information
  - New knowledge may require reshape of information
  - Persistent Data Repositories support knowledge

Data and Storage

- What are today data

Data and Storage

- What are today data
  - Numeric Values

Data and Storage

- What are today data
  - Numeric Values
  - Metadata, or data about the data = Data Model

Data and Storage

- What are today data
  - Numeric Values
  - Metadata, or data about the data = Data Model
  - Policies

Data and Storage

- What are today data
  - Numeric Values
  - Metadata, or data about the data = Data Model
  - Policies
  - Computer encoding of data

Data and Storage

- What are today data
  - Numeric Values
  - Metadata, or data about the data = Data Model
  - Policies
  - Computer encoding of data
  - Software libraries

Data and Storage

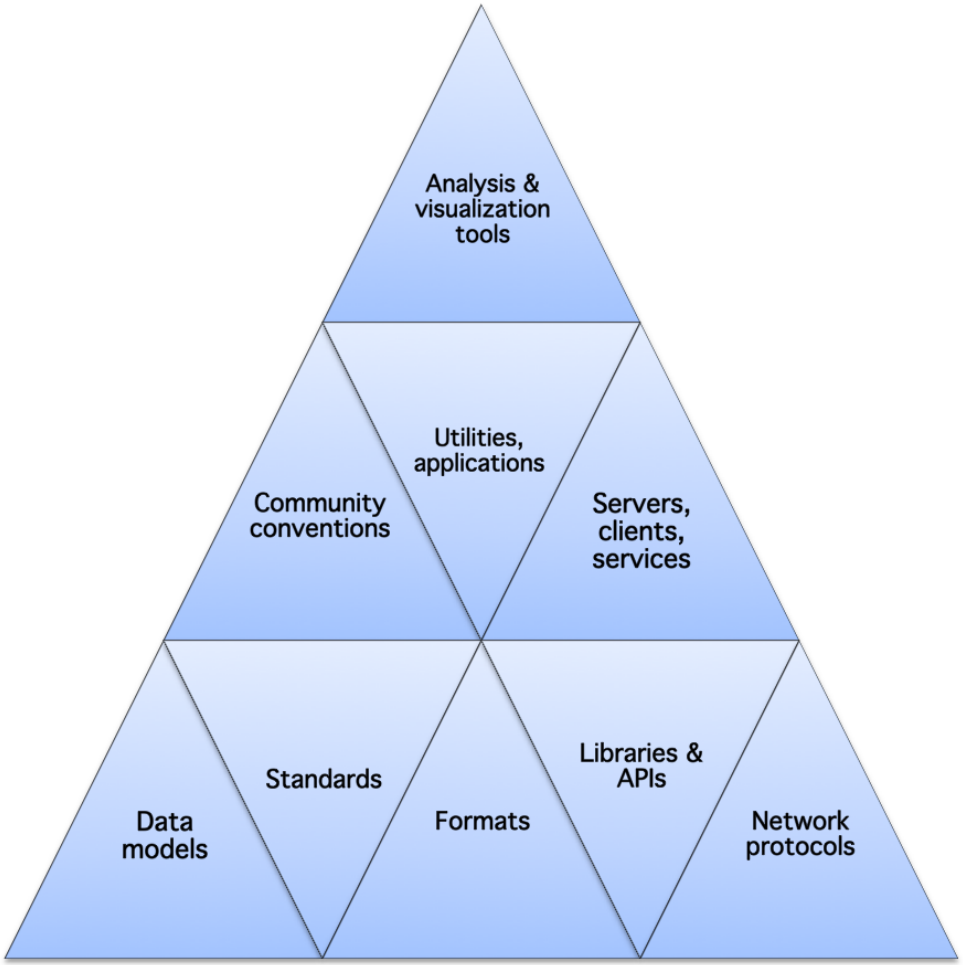
- What are today data
  - Numeric Values
  - Metadata, or data about the data = Data Model
  - Policies
  - Computer encoding of data
  - Software libraries
  - Storage hardware

Data and Storage

- What are today data
  - Numeric Values
  - Metadata, or data about the data = Data Model
  - Policies
  - Computer encoding of data
  - Software libraries
  - Storage hardware
  - Archive and distribution software

Data Infrastructure

From data to the user



Data format

- ASCII

Data format

- ASCII
  - Easy to read if small

Data format

- ASCII
  - Easy to read if small
  - Can get large

Data format

- ASCII
  - Easy to read if small
  - Can get large
  - Have to know structure to make plots

Data format

- ASCII
  - Easy to read if small
  - Can get large
  - Have to know structure to make plots
  - Slow to read, write

Data format

- ASCII
  - Easy to read if small
  - Can get large
  - Have to know structure to make plots
  - Slow to read, write
- Binary

Data format

- ASCII
  - Easy to read if small
  - Can get large
  - Have to know structure to make plots
  - Slow to read, write
- Binary
  - Smaller, faster than ASCII

Data format

- ASCII
  - Easy to read if small
  - Can get large
  - Have to know structure to make plots
  - Slow to read, write
- Binary
  - Smaller, faster than ASCII
  - Have to know structure

Data format

- ASCII
  - Easy to read if small
  - Can get large
  - Have to know structure to make plots
  - Slow to read, write
- Binary
  - Smaller, faster than ASCII
  - Have to know structure
  - Not necessarily portable

Data format

- ASCII
  - Easy to read if small
  - Can get large
  - Have to know structure to make plots
  - Slow to read, write
- Binary
  - Smaller, faster than ASCII
  - Have to know structure
  - Not necessarily portable
  - Opaque from outside application

Data format

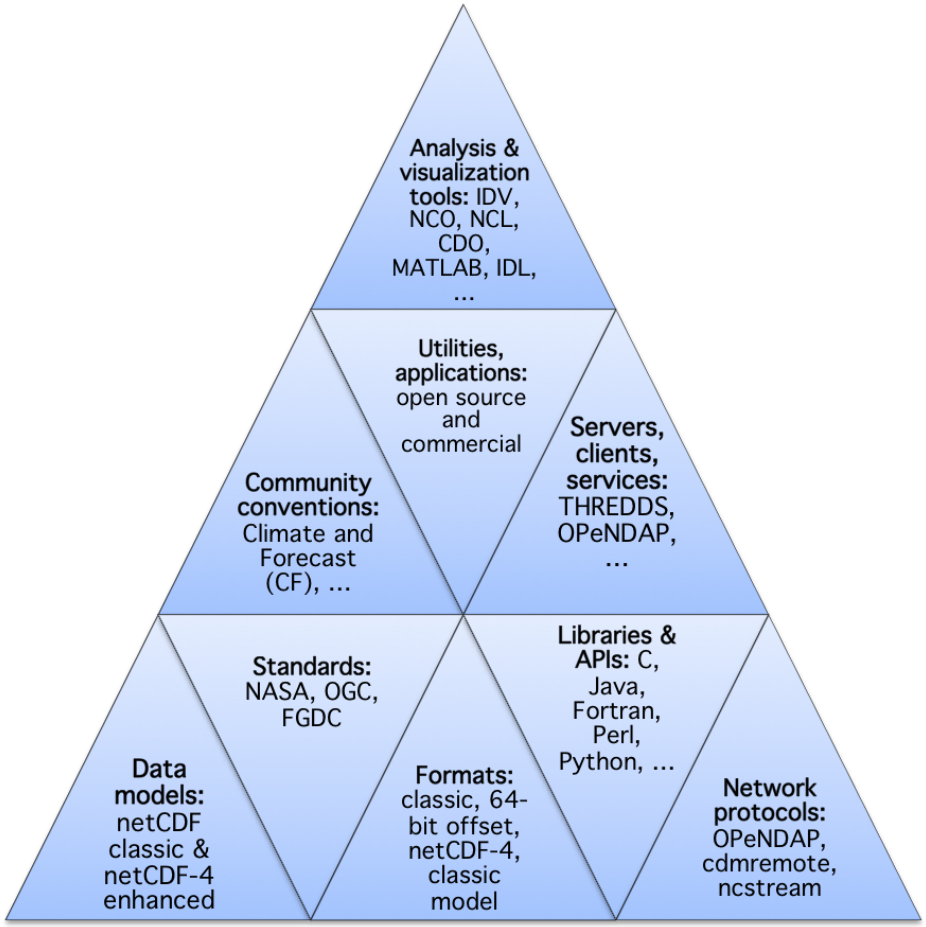
- ASCII
  - Easy to read if small
  - Can get large
  - Have to know structure to make plots
  - Slow to read, write
- Binary
  - Smaller, faster than ASCII
  - Have to know structure
  - Not necessarily portable
  - Opaque from outside application
- netCDF
  - Binary indexed portable format with standard access API for both data and metadata

netCDF Data format

- **Self-Describing**: A file includes metadata as well as data: description of variables, units of measure, etc.
- **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 file without copying the dataset or redefining its structure.
- **Extensible**: Adding new dimensions, variables, or attributes to files does not require changes to existing programs that read the files.
- **Sharable**: One writer and multiple readers may simultaneously access the same file. With Parallel netCDF , multiple writers may efficiently and concurrently write into the same file.
- **Archivable**: Access to all earlier forms of netCDF data will be supported by current and future versions of the software.
- **Networkable**: Client access to remote servers through OPeNDAP.

netCDF Infrastructure

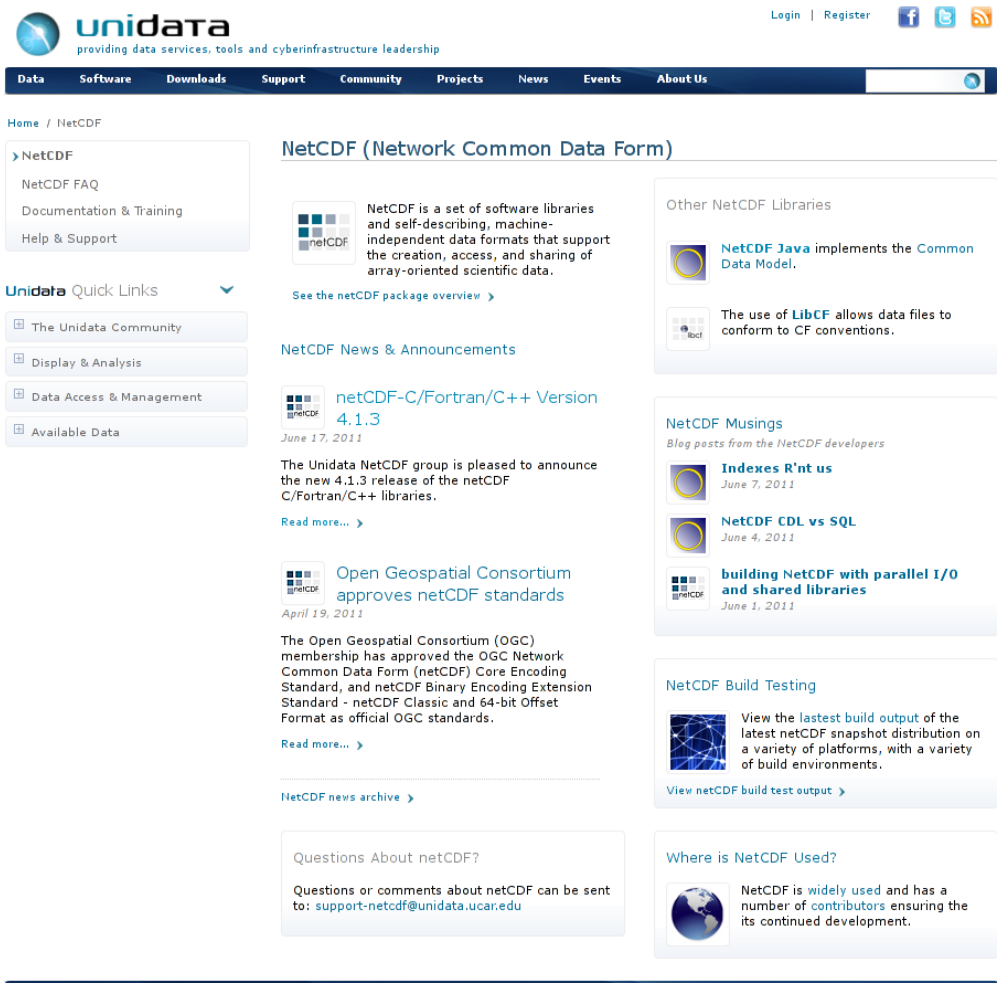
From data to the user



netCDF Home

http://www.unidata.ucar.edu/software/netcdf

Unidata Site



netCDF Users

- 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 Standard endorsement

- **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.
- **2010-03-1**: 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.

netCDF Classic Data Model

A netCDF "classic" file is composed of:

- Dimensions
- Variables
- Attributes
- Data

A file can have attributes, dimensions, and variables.
Dimensions are used to specify shapes of variables.
One dimension can be unlimited (record)
A variable can have dimensions and attributes.
Multiple variables can share dimensions (be on a grid).
Variables are of fixed primitive type (char, int, float)

Dimension

Dimensions are used to define shapes of variables.
Each dimension must have:

- Unique name in a file
- A length, i.e. an integer number

Attribute

An attribute is used to store metadata, either at file or variable level. Each attribute must have:

- Unique name in level (file or variable)
- A type
- A value

Metadata are used to establish conventions to share data.
For example, for the Climate and Forecast CF convention, a variable MUST have some attributes (for example units, standard name, etc.), and the convention name itself is a mandatory attribute at file level.

Variable

A variable is the shaped (by dimension) storage of data, defined by its metadata (attributes). Each variable must have:

- Unique name in a file
- A type
- Zero (scalar value) or more dimensions
- Zero or more attributes
- As many data values as specified by its shape

Actual scientific data are stored in variables.

netCDF Common Data Language

The file can be described using simple text.

```
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 V4

The netcdf data model is further extended with the new V4 format, built upon the HDF5 data format.

- Multiple unlimited dimensions
- User defined types and opaque types
- Data can be grouped together
- Compression and chunking
- Native Parallel and HPC oriented.

Operating System choice

Target of system usage

- **Data Archive**: netCDF data are single files on disk. A decent filesystem is needed, go for UNIX (or Linux).
- **Data Creation**: will depend on data source. If model, an HPC system may be running it, and it is usually some sort of cluster with a UNIX OS.
- **Data Analysis**: Usually performed using other tools, will be researcher choice. Be warned. Try to KISS.
- **Software Development**: as netCDF is distributed also as binary, any OS can be used (even commercial desktop solutions). Consider just TCO.
- **Data Distribution**: the DAP server stack is based on Apache Tomcat, so this task will be best served by a Linux system.

Installation type

- **Linux system**: usually netCDF comes already packed in all known distributions, use normal package manager to install. NOT OK for HPC.
- **Commercial desktop**: Find prebuilt binaries or use Java software
- **HPC or UNIX system**: Compile from source the library.
- **Tomcat Server**: Just deploy the application on the server.

Building from Source

Normally just the C compiler is needed, may also need Fortran or C++ if users are developers. Most of the software built using netCDF just use the C API. Some pieces of software are requested. Follow checklist.

- **Compression requested**: V4 feature. Need zlib, HDF5. If not explicitly asked, can avoid szip.
- **HPC system**: Probably parallel I/O requested. Need HDF5 and a vendor or opensource MPI stack.
- **Networked dataset access**: Will need the client OpenDAP, which requires curl development library installed (need xml support).

Full feature environment will require to envelop all requirements.

Normal Install

The "normal" installation procedure is just the four steps:

- configure : a script which serch required pieces. Written to build the most, if needed compilers / libraries are found on system.
- make : binary distribution library is compiled.
- make check : will build and run some test programs to verify against known results the previous step
- make install : copy all needed files to use the software in production. If installation prefix is on system space, administrator credentials are needed.

Configuring build

The most important step is configure. Verify that

- All requested features are correctly activated
- All not needed features are correctly deactivated
- You are using the same compiler for all software compilation
- The installation path is correct and writable.

If any of the make steps fails, restart from configure.

Data access or creation

Implementing a software to read or write data in netcdf format, check

- The BASE library is written in C. If performance needed, use C.
- A LOT OF DOCUMENTATION is available. So, RTFM.
- The C++ library is under heavy development. Give a try to the new V4 C++.
- If Java is chosen, use eclipse.
- If Fortran is needed, consider to start migrating to F90 standard.
- Why not going for an higher level language, such as python or perl?
- Do not use any undocumented feature. It may break and it will break.

Data analysis

Analyzing netCDF data, consider the following points:

- Why write Your own custom program to access dataset, if command line tools such as NCO can be used?
- Try to work at file level using operators: faster, easier.
- Work by steps: do not try to analyze terabyte of data at first try: subset data using operators, analyze subsets, and if procedure is OK, go for big goal.
- If a plot is needed, use user friendly tools like IDV, GrADS, NCL. Do not implement all back at low level: You are supposed to be a researcher, don't you?
- Try as much as possible to access data through network servers: a local copy of data is not Your target. You need data to build knowledge, let SysAdmins do their well paid jobs.

Data distribution

Distributing netCDF data, consider the following points:

- Ask for, add by yourself, integrate as much metadata as possible in distributing data. Be not redundant, but ensure any information ON data is distributed WITH data: this is one of the primary goal of netCDF format.
- To archive data, use exstensively compression and chunking, to have small storage space and fast data access.
- You are not just distributing numbers: you are distributing the base for the knowledge. Be ready to enhance on user request. A developer will ALWAYS be needed along a distribution site, to implement new features.
- Enforce from the beginning distribution policies, but have wider as possible distribution as a goal. Ask for free access, implement access as free as possible.

Thank You !