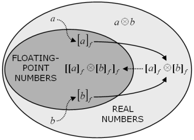
Working with big data

Graziano Giuliani
International Centre for Theorethical Physics - Trieste Earth System Physics Section

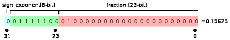
ICTP - Earth System Physics Section

Workshop on Advanced Techniques for Scientific Programming and Management of Open Source Software Packages
Trieste, 10 Mar - 21 Mar 2014

Data and format



- Computer data are formatted abstractions
- Internal storage representations
  - IEEE 754 standard



- External persistent and exchange format
  - Data Digital Storage
  - Data Model
  - Data Description
  - Data Manipulation
  - Data Distribution

Big Data

... A collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications...

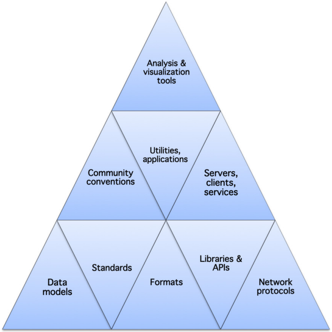
- ... as of 2012, every day 2.5 exabytes (2.5×10^{18}) of data were created ...
- Big Science
  - 25 petabytes annual rate before replication for LHC at CERN
  - 200 GB per day produced by the SDSS, going to get five times more when LSST will get online
  - The CMIP5 Project will produce 3 petabytes of data, which is 100 times the data produced by CMIP3. Data traffic of the CMIP3 was 900 GB per month, and for the CMIP5 a data federation grid has been built. DKRZ node has 60 petabytes of data storage.
- Private Sector
  - Google processs 24 petabyte of data per day, and in 2009...
- ... and one petabyte of MP3 songs requires 2000 years to play.

Data Digital Storage

- File concept : Operating System Dependent File System
  - Data hierarchy
  - Naming limitations
  - Size limitations
  - Permission management
- Access is through I/O channels and libraries
  - Block Storage
  - Sequential Access
  - File system data structure
  - Programming Languade dependent STANDARD API
- Application Point of view
  - Reconstruct values and meaning of data

Data Model

- Physical Data Model
  - How the data are stored and transmitted
- Logical Data Model
  - How the data are physically organized
  - Structural Metadata and data objects
- Conceptual Data Model
  - Semantic of the data, what the data is about
  - Operations and Rules of consistency and integrity



Data Description

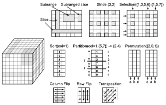
- Metadata Concept

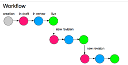
  METADATA ARE DATA ABOUT THE DATA

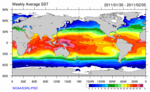
  - How the data have been created ?
    - Model ? Experiment ?
  - Purpose of the data
    - Project XYZ
  - Creator or author of the data
    - University of BLABLA , Research Center Y, Miss Purple
  - Where are the data HOME
    - URL to Download
  - Standards used and meaning

Data Manipulation

- Extract partial information
    - Subsetting of multidimensional datasets
- Adding data or metadata to a dataset
- Perform data analysis to add value to dataset
- Versioning of data
- Visualize data







Data Distribution

Data Authoring and content access control

- Data Provider
  - Catalogues of data
  - Search Engine
  - Can Bridge different Providers
  - User interface
  - Support through protocols Multiple clients
- Data User
  - Browse a Catalog
  - Finds a particular dataset
  - Single point of access
  - Uniform experience
  - Uses desktop analysis tools

netCDF Data Format

What is a netCDF format

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

... Software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data ...

Basic Concepts

- Self-describing - Metadata and data together
- Portable - Same API on a large number of OSes
- Scalable and allowing network read access through OpenDAP protocol
- Append and Share capability among different processes/threads
- Archivable : Unidata is committed to always provide back compatibility

Standard

What is a standard : Any norm, convention or requirement

- Requires an effort to conform when writing data
- Reduces time of data access, storage and usage

The price is just meaningless given the result, it does not exist if using an API. The open-create/read-write/close sequence does not change.

- NASA Earth Science Data Systems (ESDS) format for data
- Integrated Ocean Observing System (IOOS) Data Management and Communications (DMAC) Subsystem
- US Federal Geographic Data Committee (FGDC)
- Open Geospatial Consortium (OGC) approved "OGC Network Common Data Form (NetCDF) Core Encoding Standard version 1.0"

Conventions

The netCDF format is the de-facto standard in the Climate Research

- The CF convention for Climate and Forecast data fixes
  - Standard to describe the dataset
  - Standard to describe the data
  - Standard for expressing coordinate systems
  - Standard for lossy and lossless data compression
  - Standard for expressing Sampling Geometries
  - Standard to represent time and space statistics
  - Fixed names for variables, metadata and units

Who uses netCDF

- NOAA's Climate Analysis Branch (CAB)
- EUMETSAT Satellite data distribution
- NASA's Halogen Occultation Experiment
- The Woods Hole Field Center of the USGS
- The CSIRO Division of Atmospheric Research in Australia
- General purpose finite element data model at SANDIA
- Multi-body dynamics analysis systems
- Analytical Data Interchange Protocols for chromatography and mass spectrometry
- The Positron Imaging Laboratories and the Neuro-Imaging Laboratory of the Montreal Neurological Institute
- molecular dynamics and the simulation of biomolecules for the AMBER project
- Culham Centre for Fusion Energy
- The ICTP RegCM model

netCDF file on disk

- Three on disk file format with one variant, one API.
  - NetCDF 3 classic : binary file with maximum size of 2 GB
  - NetCDF 3 64 bit offset : binary file with max 4GB per variable
  - NetCDF 4 : HDF5 file format with netCDF API on top
  - NetCDF 4 Classic : HDF5 file format with some limitations

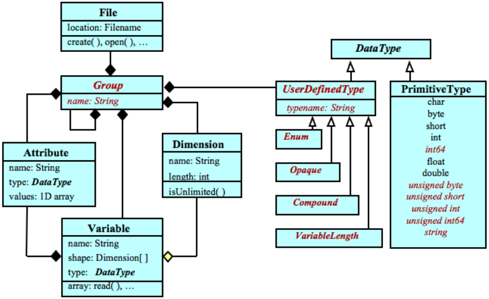
Java API supports also different file formats on top of which it offers the same netCDF API.

netCDF-4 objects (I)

All objects are identified with a name as an UTF8 character string.

- GROUP : each file contains one or more groups. They have distinct namespaces, and can be considered equivalent to on disk directores. Any group can contains multiple GROUPs If not defined a GROUP, all other objects are in the ROOT group.
  - DIMENSION : an integer number, fixed or growing, defining a dimensionality
  - DATATYPE : User defined or primitive data structure
  - VARIABLE : The actual data which has any number of dimensions and any number of attrybutes and one DATATYPE
  - ATTRIBUTE : any data type associated either to a GROUP or to a VARIABLE

netCDF-4 objects (II)



netCDF-4 objects (III)

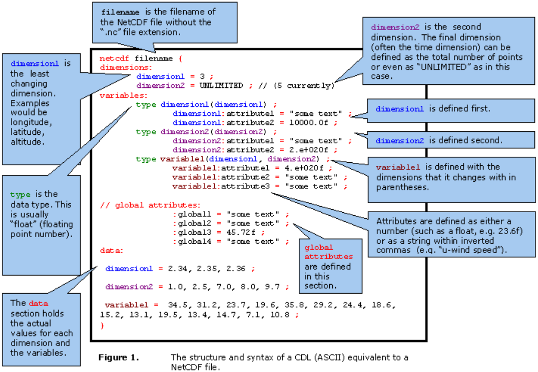


Figure 1. The structure and syntax of a CDL (ASCII) equivalent to a NetCDF file.

Usage Pattern I : API

- Unidata supports netCDF APIs in C, C++, FORTRAN 77, Fortran 90, and Java.
- netCDF API for Python , perl , Ruby , R , Matlab , IDL

A couple of usage pattern, using the Fortran 90 API , the C++ API and the simple but complete netcdf4 Python API

- `https://code.google.com/p/netcdf4-python/`

Usage Pattern Fortran 90 I : Create a netCDF f

```
program example_netcdf
  use netcdf

  [...]

  errval = nf90_create(filename, NF90_NOCLOBBER, ncid)
  if (errval /= NF90_NOERR) then
    print *, 'Cannot write output file ', trim(filename)
    print *, nf90_strerror(errval)
    stop
  end if

  errval = nf90_put_att(ncid, nf90_global, 'title', 'The title')
  [...]
  errval = nf90_def_dim(ncid, 'X', nx, idims(1))
  [...]
  errval = nf90_def_dim(ncid, 'Y', nx, idims(2))
  [...]
  errval = nf90_def_dim(ncid, 'time', nf90_unlimited, idims(3))
  [...]
  errval = nf90_def_var(ncid, 'randomwalk', nf90_double, idims, ivar)
  [...]
  errval = nf90_put_att(ncid, ivar, 'units', 'dimensionless')
  [...]
  errval = nf90_enddef(ncid)
```

Usage Pattern Fortran 90 II : Create a netCDF

```
  do it = 1 , nt
    call random_number(dxy)
    x = x + dxy(1)
    y = y + dxy(2)
    do j = 1 , nx
      do i = 1 , nx
        matrix(i,j) = dlog((dble(i-hx)-x)**2+(dble(j-hx)-y)**2+0.1)
      end do
    end do
    istart(1) = it
    istart(2) = 1
    istart(3) = 1
    icount(1) = 1
    icount(2) = nx
    icount(3) = nx
    errval = nf90_put_var(ncid, ivar, matrix, istart, icount)
    [...]
  end do

  errval = nf90_close(ncid)
  [...]

end program example_netcdf
```

Usage Pattern C++ I : Create a netCDF file

```
[...]
#include <netcdf>

#define NC_ERR 2
using namespace netCDF;
using namespace netCDF::exceptions;

int main(int argc, char *argv[])
{
  [...]
  NcFile newf;
  try {
    NcFile newf(argv[3], NcFile::replace);
    newf.putAtt("title","Random walk of a log surface");
    xdim = newf.addDim("X", nx);
    ydim = newf.addDim("Y", nx);
    tdim = newf.addDim("time");
    std::vector<NcDim> dims;
    dims.push_back(tdim);
    dims.push_back(ydim);
    dims.push_back(xdim);
    xvar = newf.addVar("randomwalk", ncDouble, dims);
  }
  catch(NcException& e)
  {
    e.what();
    return NC_ERR;
  }
```

Usage Pattern C++ II : Create a netCDF file

```
  [...]
  std::vector<size_t> startp , countp;
  startp.push_back((size_t) 0);
  startp.push_back((size_t) 0);
  startp.push_back((size_t) 0);
  countp.push_back((size_t) 1);
  countp.push_back((size_t) nx);
  countp.push_back((size_t) nx);
  for (size_t it = 0; it < (size_t) nt; it++)
  {
    [...]
    try {
      startp[0] = it;
      xvar.putVar(startp,countp,matrix);
      std::cout << " It " << it << std::endl;
    }
    catch(NcException& e)
    {
      e.what();
      return NC_ERR;
    }
  }

  return 0;
}
```

Usage Pattern Python I : Create a netCDF file

```
#!/usr/bin/env python
from netCDF4 import Dataset
import numpy
from numpy.random import uniform
import time

lats =  numpy.arange(-90,91,2.5)
lons =  numpy.arange(-180,180,2.5)
nlats = numpy.size(lats)
nlons = numpy.size(lons)

rootgrp = Dataset('test.nc', 'w', format='NETCDF4')
rootgrp.description = 'bogus example script'
rootgrp.history = 'Created ' + time.ctime(time.time())
fcstgrp = rootgrp.createGroup('forecasts')
analgrp = rootgrp.createGroup('analyses')
level = rootgrp.createDimension('level', 10)
lat = rootgrp.createDimension('lat', nlats)
lat = rootgrp.createDimension('lon', nlons)
time = rootgrp.createDimension('time', None)
times = fcstgrp.createVariable('time','f8',('time',))
levels = fcstgrp.createVariable('level','i4',('level',))
latitudes = fcstgrp.createVariable('latitude','f4',('lat',))
longitudes = fcstgrp.createVariable('longitude','f4',('lon',))
temp = fcstgrp.createVariable('temp','f4',('time','level','lat','lon',))
```

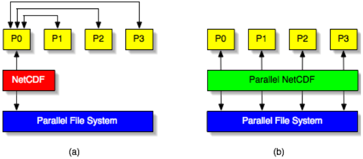
Usage Pattern Python II : Create a netCDF file

```
latitudes.units = 'degrees north'
longitudes.units = 'degrees east'
levels.units = 'hPa'
temp.units = 'K'
times.units = 'hours since 1949-01-01 00:00:00.0'
times.calendar = 'gregorian'

latitudes[:] = lats
longitudes[:] = lons
temp[0:5,0:10,:,:] = uniform(size=(5,10,nlats,nlons))
levels[:] =  [1000.,850.,700.,500.,300.,250.,200.,150.,100.,50.]
rootgrp.close()
```

Parallel netCDF

- Parallel I/O with netCDF requires a parallel File System



- Used properly, it allows users to overcome I/O bottlenecks **in high performance computing environments**/
- Built on top of HDF5, on top of MPI-IO layer
- Only netCDF 4 file format is capable of parallel I/O
- HDF5 must be built with enable-parallel setting CC=mpicc
- netCDF configure script will detect the parallel capability of HDF5 and build the netCDF-4 parallel I/O features automatically

Opening/Creating Files for Parallel I/O

- nc_open_par and nc_create_par functions are used to create/open a netCDF file with the C API
- For Fortran 90 users the nf90_open and nf90_create calls have been modified to permit parallel I/O files to be opened/created using optional parameters MPI communicator and get back MPI status.
- The subsetting start and count arguments given to read or write functions allows reading and writing in parallel.
- All netCDF metadata writing operations are collective.
- Data reads and writes may be independent (the default) or collective.

Fortran 90 Example

```
[...]
  call MPI_Init(ierr)
  call MPI_Comm_rank(MPI_COMM_WORLD, my_rank, ierr)
  call MPI_Comm_size(MPI_COMM_WORLD, p, ierr)
[...]
  istatus = nf90_create(FILE_NAME, IOR(NF90_NETCDF4, NF90_MPIIO), &
        ncid, comm = MPI_COMM_WORLD, info = MPI_INFO_NULL)
[...]
  start = (/ 1, my_rank + 1/)
  count = (/ p, 1 /)
  istatus = nf90_put_var(ncid, varid, data_out, &
        start = start, count = count)
[...]
  istatus = nf90_close(ncid)
  call MPI_Finalize(ierr)
```

Data Compression

- Readers access data from compressed variables transparently, without needing to know they are compressed
- Compressed variables are stored with chunked storage
- Each chunk is compressed or uncompressed independently
- Better compression can be achieved with custom chunking
- HDF5 library allows the creation of custom filtering, the netCDF library exports only compression and shuffle filters.
- Chunking a variable to speed one I/O access pattern can greatly degrade performances with other access patterns.

Data distribution using OpenDAP

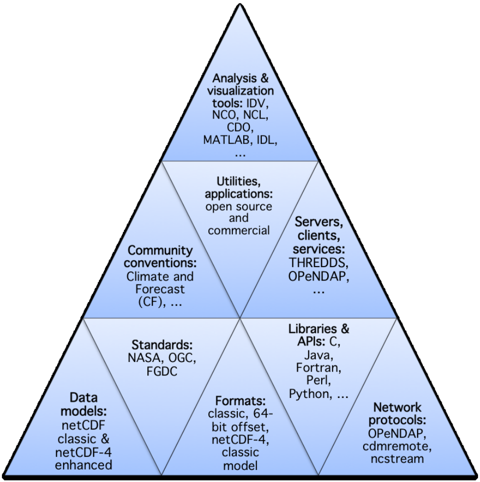
- OpenDAP is a client/server model, with a client that sends requests for data out onto the network to some server
- netCDF library can read data from remote OpenDAP server just using URI instead of local disk file path.
- Data from the original data file is translated by the OPeNDAP server into the OPeNDAP data model for transmission to the client.
- Upon receiving the data, the client translates the data into the data model it understands.

Data as a service

- The OPeNDAP data server is made up of two pieces
  - The front-end server is a Tomcat servlet, the OPeNDAP Lightweight Front-End Servlet (OLFS)
    - Receives request for data
    - Translate into the requested form (ASCII , SOAP)
    - Can provide Catalog responses and metadata
  - The Back-End Server (BES) is a high performance data access process, which provides data read from data files or a SQL database in XML encapsulated objects to the OLFS. It does not handle data conversion

netCDF data model

The netCDF data model pyramid can mow be filled.



Is this enough ?

What is missing in the picture ?

- Metadata query capabilities?
  - SciDB is on the way, a data management and analytical Software System
- Algebra Services which should be added on Server side:
  - Statistical summarization
    - Calculate means; form masks; accumulate counts (binning)
  - Regridding/resampling
    - Remap onto specd meshes...
  - Criteria-driven subset creation
    - Select/build data structures that satisfy a predicate
  - Feature extraction
    - Construct lines/polygons from images, grids, meshes...