NHR-Nord@Göttingen

Holistic HPC I/O

Storage API

Mohammad Hossein Biniaz, Kevin Lüdemann

Table of contents

Learning Objectives

- Learn about different types of metadata
- Understand self contained data files
- Develop a POSIX/NetCDF/HDF5 data model for a given use case
- Implement a data module with an HDF5 file

What is Metadata?

- Filename
- File owner and group
- Parent file structure/folder
- Creation date, date of last change
- Access permissions
- Characteristic (file description)
- Parameters for the creation of this file
- Hardware architecture
- ...

How to store metadata?

- File/folder name
- Describing file
- Elaborate folder structure
- Indexed in database
- Self contained data format

Folders and files

- Run the list command in CMD to see metadata
- List filename, folder name, change date, permissions
- Create elaborate folder structure
- Create explaining file for additional metadata

Advantages/Disadvantages

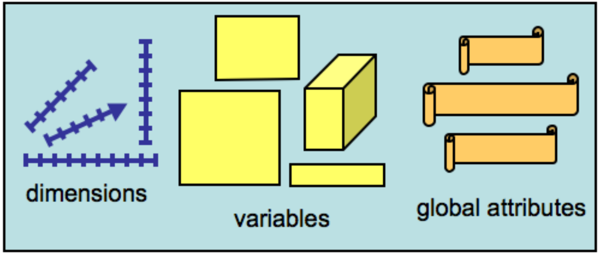
- Advantages
  - Native to Linux
  - Many API for interaction exists
  - Human readable without difficult tools
  - Easy to understand
- Disadvantages
  - Stores metadata for all files separate
  - Accessing each file has latency
  - Additional metadata requires more files
  - Too simple for large data sets with much metadata

NetCDF

- NetCDF is an example for a "high-level" I/O-API and ecosystem
- In a simple view, NetCDF is:
  - A data model
  - A file format
  - A set of APIs and libraries for various programming languages
- Together, the data model, file format, and APIs support
  - creation, access, and **sharing** of scientific data
- Allows to describe multidimensional data and include metadata which further characterizes the data
- APIs are available for most programming languages used in geo-sciences

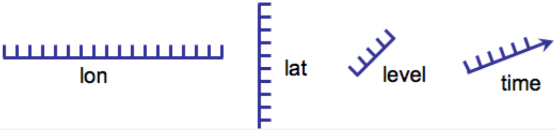
The Classic NetCDF Model

- NetCDF files are containers for Dimensions, Variables, and Global Attributes.
- A NetCDF file (dataset) has a path name and possibly some dimensions, variables, global (file-level) attributes, and data values associated with the variables.



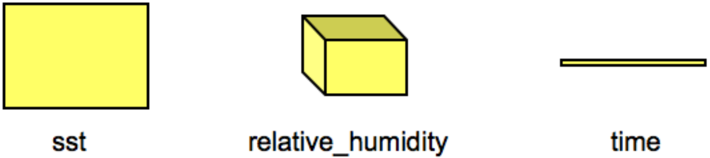
The Classic NetCDF Model – Dimensions

- Dimensions are used to specify variable shapes, grids, and coordinate systems.
- A dimension has a name and a length.
- A dimension can be used to represent a real physical dimension
  - Example: time, latitude, longitude, or height
- A dimension can also be used to index other quantities
  - Example: station or model run number
- The same dimension can be used in multiple variables.



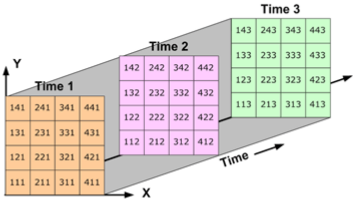
The Classic NetCDF Model – Variables

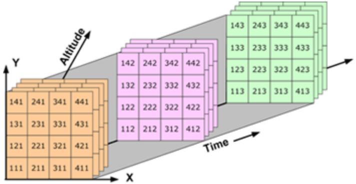
- A variable holds a multidimensional array of values of the same type
- A variable has a name, type, shape (according to dimensions), attributes, and values
- In the classic data model, the type of a variable is the external type of its data as represented on disk, one of: `char` (text character), `byte` (8 bits), `short` (16 bits), `int` (32 bits), `float` (32 bits), `double` (64 bits)



The Classic NetCDF Model – Data

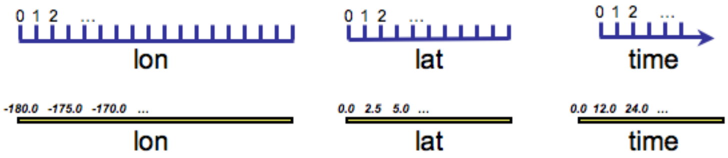
- The data in a NetCDF file is stored in the form of arrays. For example:
  - Temperature varying over time at a location is stored as a **one-dimensional array**
  - Temperature over an area for a given time is stored as a **two-dimensional array**
  - Three-dimensional (3D) data, like temperature over an area varying with time, or four-dimensional (4D) data, like temperature over an area varying with time and altitude, is stored as a **series of two-dimensional arrays**





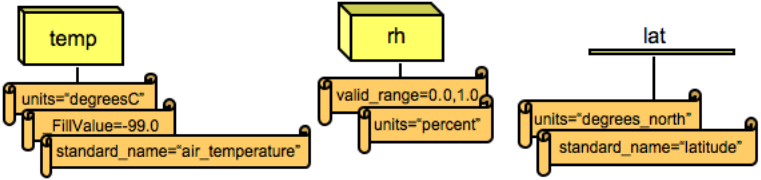
The Classic NetCDF Model – Coordinate Variables

- A 1D variable with the same name as a dimension is a **coordinate variable**
- The coordinate variable is associated with a dimension of one or more data variables and typically defines a physical coordinate corresponding to that dimension
- Many programs that read NetCDF files recognize coordinate values they find



The Classic NetCDF Model – Attributes

- Attributes hold metadata (data about data)
- Attributes contain information/properties of a variable or the whole dataset
- Attributes are scalars or 1-D arrays
- An attribute has a name, type, and values.
- Attributes are used to specify such properties as units, standard names (that identify types of quantity), special values, maximum and minimum valid values, scaling factors, offsets, ...



Common Data form Language (CDL)

- Notation used to describe NetCDF object is called CDL (network Common Data form Language)
  - Provides a convenient way of describing NetCDF datasets
- Utilities allow producing CDL text files from binary NetCDF datasets and vice-versa
- File contains dimensions, variables, and attributes
- Components are used together to capture the meaning of data and relations among data fields

```
netcdf filename {
dimensions:
        lat = 3 ;
        lon = 4 ;
        time = UNLIMITED ; // (2 currently)

variables:
        float lat(lat) ;
                lat:long_name = "Latitude" ;
                lat:units = "degrees_north" ;
        float lon(lon) ;
                lon:long_name = "Longitude" ;
                lon:units = "degrees_east" ;
        int time(time) ;
                time:long_name = "Time" ;
                time:units = "days since 1895-01-01" ;
                time:calendar = "gregorian" ;
        float rainfall(time, lat, lon) ;
                rainfall:long_name = "Precipitation" ;
                rainfall:units = "mm yr-1" ;
                rainfall:missing_value = -9999.f ;

// global attributes:
                :title = "Historical Climate Scenarios" ;
                :Conventions = "CF-1.0" ;

data:
 lat = 48.75, 48.25, 47.75;
 lon = -124.25, -123.75, -123.25, -122.75;
 time = 364, 730;
 rainfall =
   761, 1265, 2184, 1812, 1405, 688, 366, 269, 328, 455, 524, 877,
   1019, 714, 865, 697, 927, 926, 1452, 626, 275, 221, 196, 223;
}
```

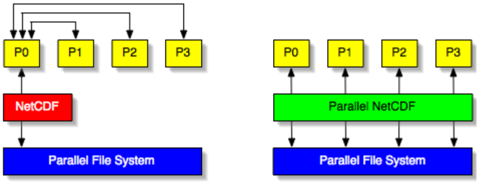
Coordinate variable

Variable attribute

Global attribute

Parallel I/O in NetCDF-4

- NetCDF-4 provides parallel file access to both classic and NetCDF-4/HDF5 files
- The parallel I/O to classic files is achieved through PNetCDF while parallel I/O to NetCDF-4 files is through HDF5 or ESDM, ZARR format support is coming
- NetCDF-4 exposes the parallel I/O features of HDF5
  - HDF5 provides easy-to-use parallel I/O feature



HDF5

- HDF5 == Hierarchical Data Format, v5
- Versatile data model consisting of groups and datatsets with a wide variety of metadata
- Open source software library running on a wide range of computational platforms, from cell phones to massively parallel systems
- Rich set of integrated performance features allowing access time and storage space optimizations
- Tools and applications for managing, manipulating, viewing, and analyzing the data in the collection
- Completely portable file format with no limit on the number or size of data objects stored

HDF5

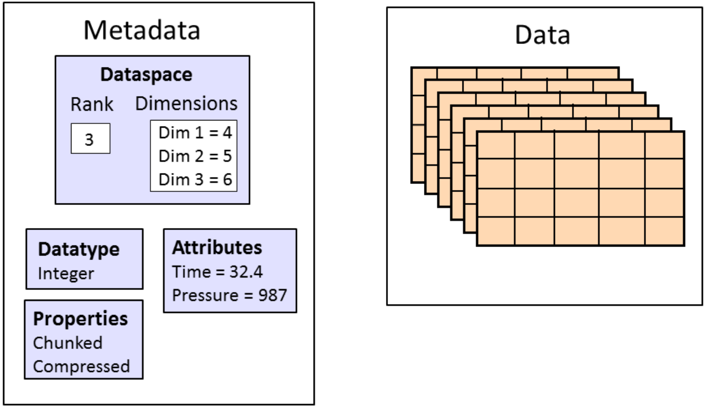


Figure: Dataset in HDF5 (Source: `https://docs.hdfgroup.org/hdf5/v1_14/v1_14_4/_intro_h_d_f5.html`)

HDF5 file drivers

- Memory only driver
- POSIX driver for single file
- Multi file driver
- Parallel driver using MPI-IO

HDF5 file features

- Hierarchical structure like POSIX file tree
- Metadata attached to each part of the tree
- Chunking of data blocks for appending and others
- Compression of data fields
- Probably encryption in future

HDF5 programming API

- C/C++ API, since HDF5 is written in C
- Fortran library
- Julia Lang API
- Python API, h5py

H5PY File example

```python
import h5py

f    = h5py.File('myfile.hdf5', 'w')

dset = f.create_dataset("mydataset", (100,50), dtype='f')

```

H5PY File example

```python
import h5py

f         = h5py.File('myfile.hdf5', 'w')

grp       = f.create_group("subgroup")

dset_grp = grp.create_dataset("mydataset", (100,50), dtype='f')

dset      = f['/subgroup/mydataset']
```

H5PY File attributes

```python
import h5py

f       = h5py.File('myfile.hdf5', 'w')
grp     = f.create_group("subgroup")
dset    = grp.create_dataset("mydataset", (100,50), dtype='f')

grp.attrs['description'] = 'birds on pond'
dset.attrs['name']       = 'duck'
dset.attrs['size']       = 128
dset.attrs['shape']      = (32,4)
f.attrs['date']          = '20240612_1430'
```

H5PY parallel IO

```python
from mpi4py import MPI
import h5py

comm = MPI.COMM_WORLD
rank = MPI.COMM_WORLD.rank

f    = h5py.File('parallel_test.hdf5','w',
                 driver='mpio',
                 comm=comm)

dset = f.create_dataset('test', (4,), dtype='i')

dset[rank] = rank

f.close()
```

Summary

- Metadata is highly important in science
- Storing and sharing Metadata can be difficult
- POSIX, NetCDF, and HDF5 store Metadata differently
- Self contained data models are easy to be shared
- Access storage with parallel threads using MPI-IO and HDF5

Exercise