NetCDF-Java Overview

John Caron

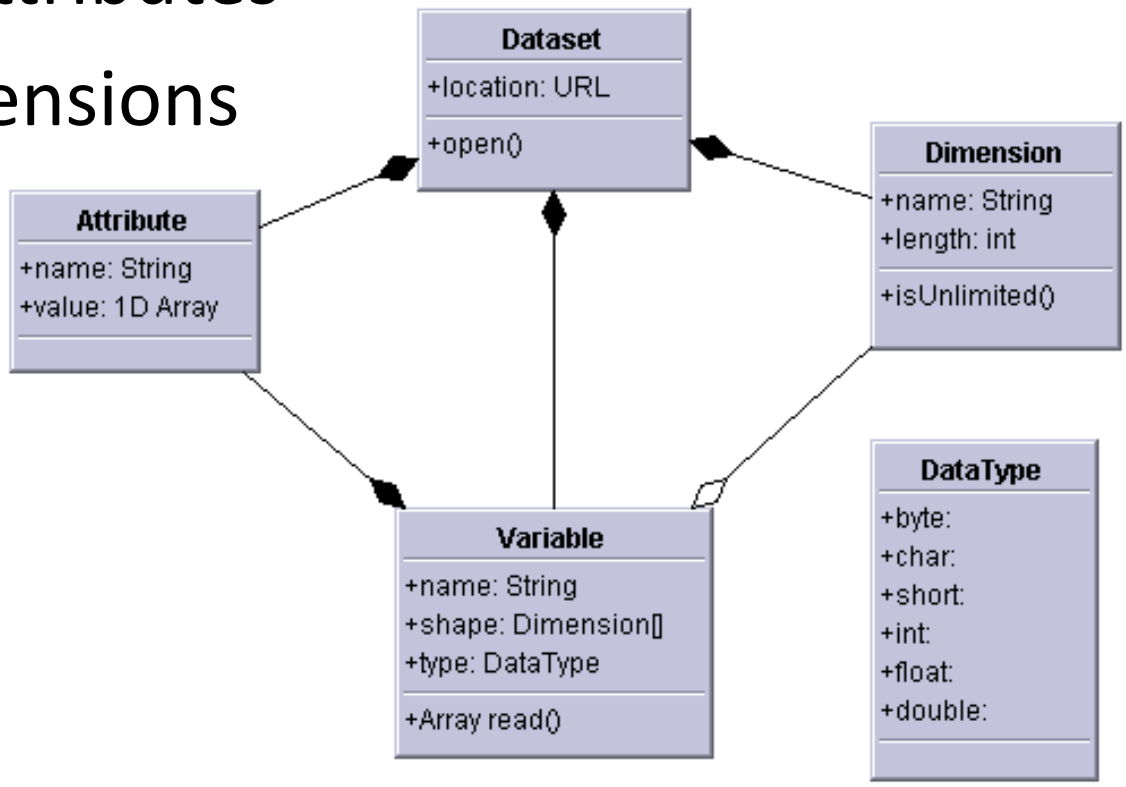
Oct 29, 2010

Contents

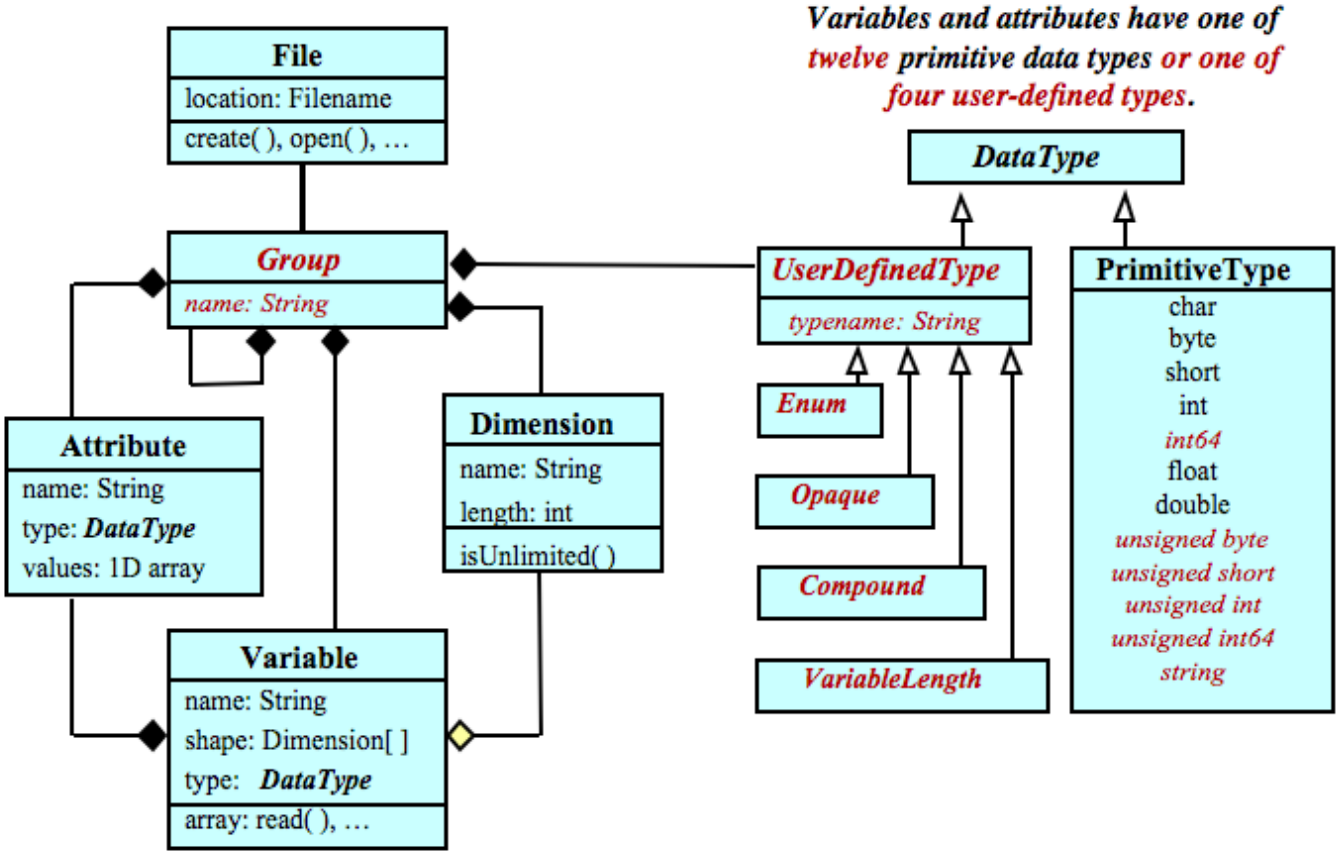
- Data Models / Shared Dimensions
- Coordinate Systems
- Feature Types
- NetCDF Markup Language (NcML)
- THREDDS Data Server (TDS)

NetCDF-3 data model

- Multidimensional arrays of primitive values
  - byte, char, short, int, float, double
- Key/value attributes
- Shared dimensions
- Fortran77

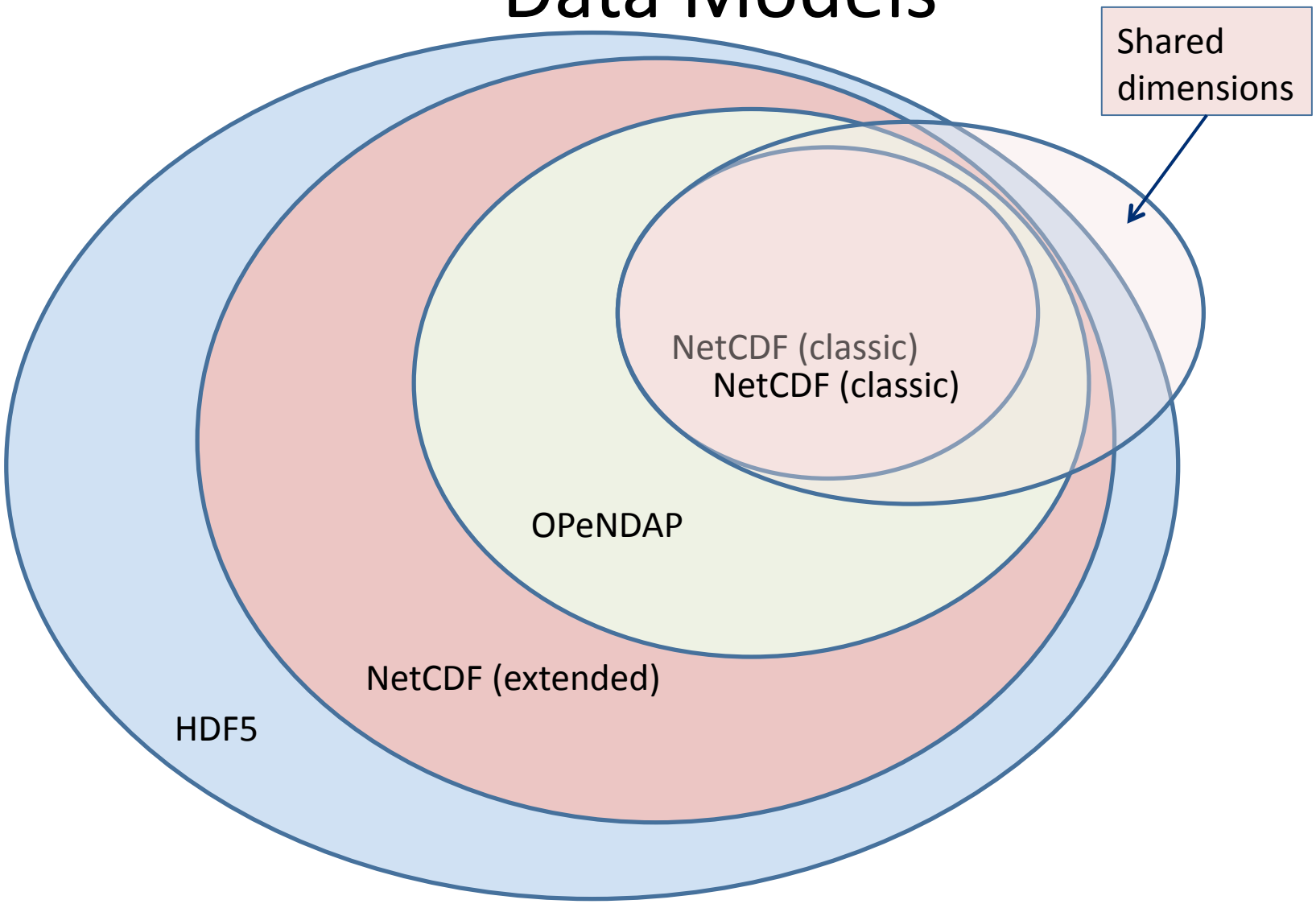


NetCDF-4 Data Model based on HDF5



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.

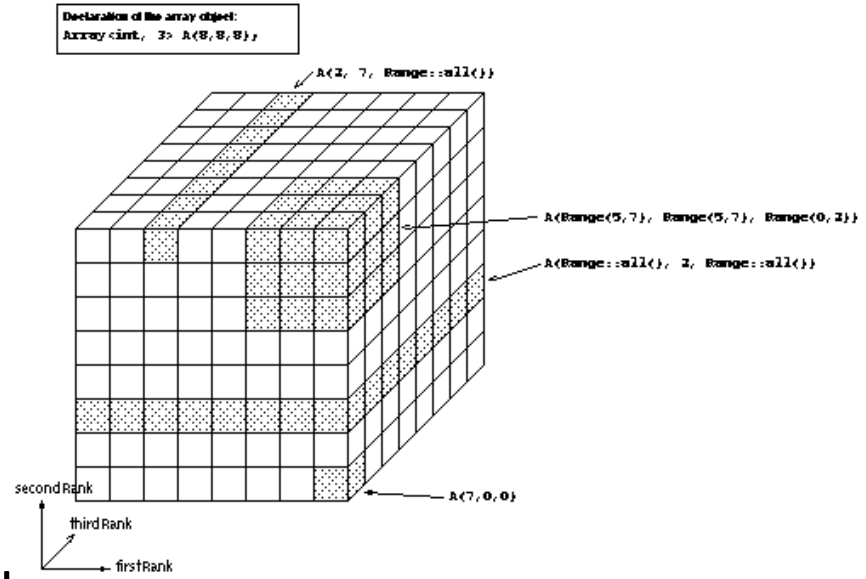
NetCDF, HDF5, OPeNDAP Data Models



Gridded Data

- Cartesian coordinates
- Data is 2,3,4D
- All dimensions have 1D coordinate variables

```
float gridData(t,z,y,x);
  float t(t);
  float y(y);
  float x(x);
  float z(z);
```

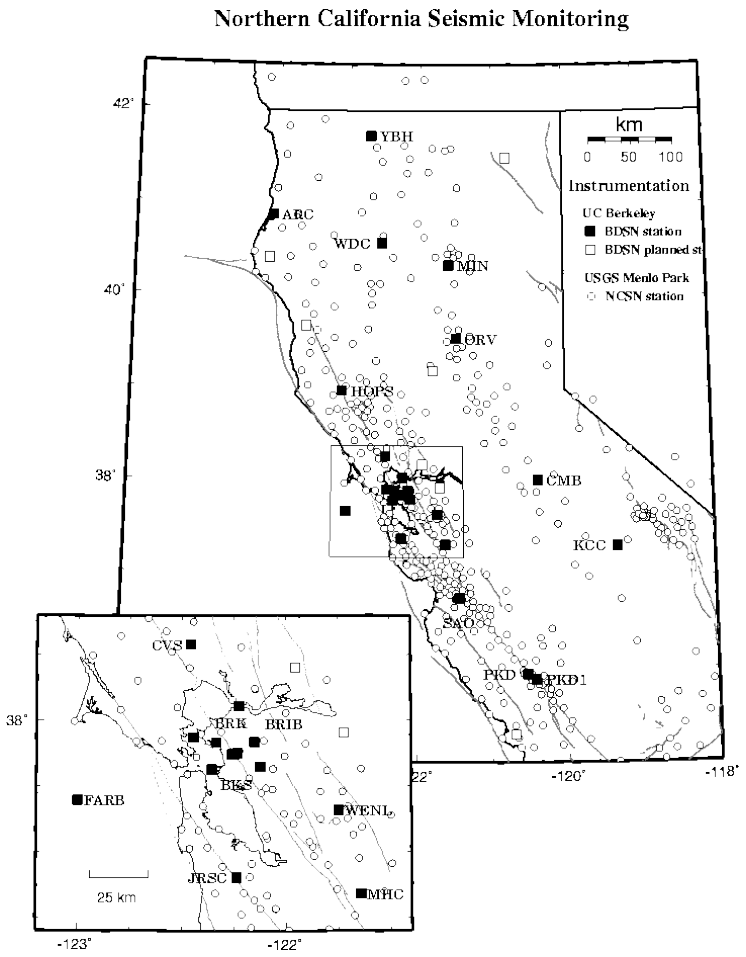


- netCDF: coordinate variables
- OPeNDAP: approx grid map variables
- HDF: dimension scales

Point Observation Data

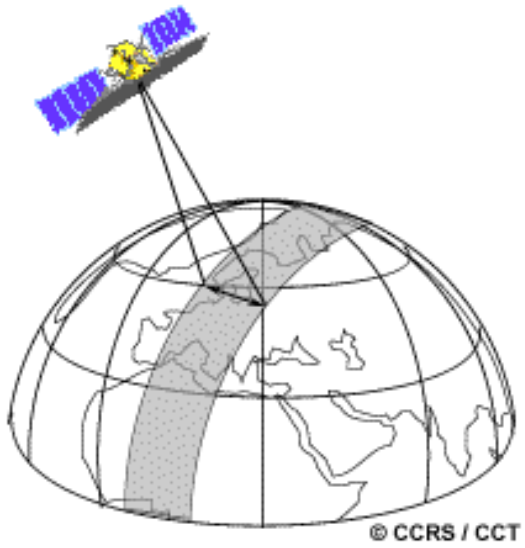
- Set of measurements at the same point in space and time = obs
- Collection of obs = dataset
- Sample dimension not connected

```
float obs1(sample);
float obs2(sample);
  float lat(sample);
  float lon(sample);
  float z(sample);
  float time(sample);
```



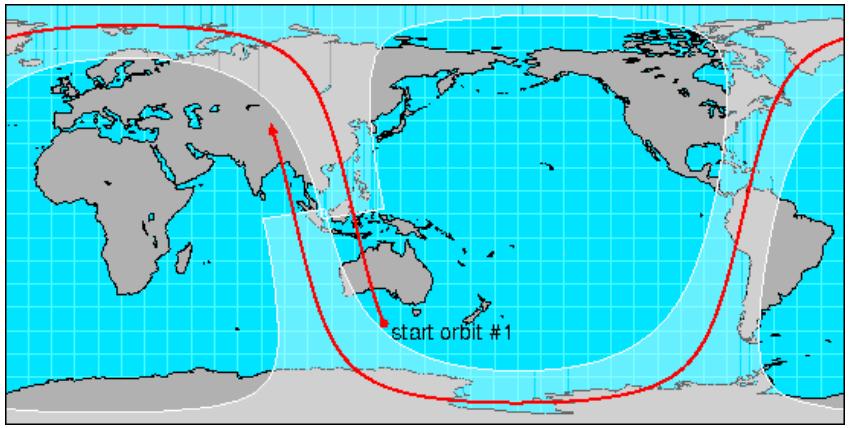
- netCDF-CF: auxiliary coordinate variables
- OPeNDAP: ~~grid~~ map variables (?)
- HDF: dimension -> dimension scales must be 1-N

Swath



- two dimensional
- track and cross-track
- not separate time dimension
- aka *curvilinear coordinates*

float swathData(track, xtrack)
  float lat(track, xtrack)
  float lon(track, xtrack)
  float alt(track, xtrack)
  float time(track)



- netCDF-CF: auxiliary coordinate variables
- OPeNDAP: not in DAP-2
- HDF: no

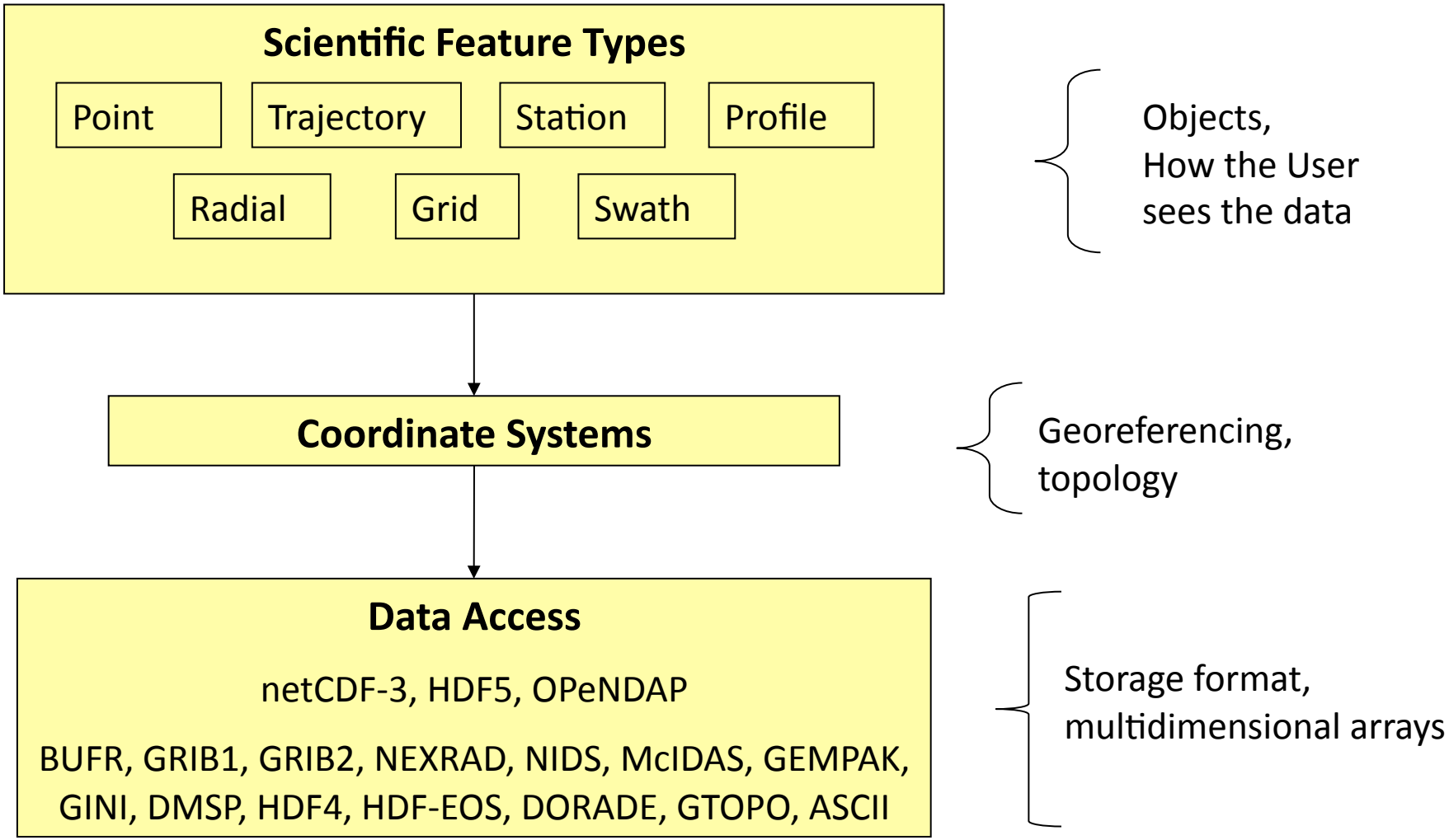
Shared Dimensions Summary

- netCDF
  - Shared dimension plus conventions is general solution for coordinates
  - *:coordinates = "lat lon alt time"*
- OPeNDAP
  - No shared dimensions in current data model
  - Shared dimensions will be added to DAP-4 (still waiting)
  - *:coordinates = "lat lon alt time"*
- HDF5
  - No shared dimensions in current data model
  - HDF-EOS added shared dimensions in metadata (NPOESS took them out)
  - NetCDF-4 adds a workaround
    - NetCDF-4 not a subset of HDF-5
  - NetCDF-4 does not read all HDF-5 objects
    - HDF-5 not a subset of NetCDF-4

Editorial

- We've "solved" the syntactic issues
  - Hardware, OS, application independence
  - Encapsulate file storage details
  - Efficient strided array access
  - TODO: parallel I/O, very large files, adapting to new hardware (SSD, etc)
- The next 5 years are about adding "semantic" support
  - Unidata : Earth Science domain
  - Georeferencing (space / time subsetting)
  - Virtual datasets: Encapsulate file collection details

Unidata's Common Data Model



C

```
count[0] = 1;
count[1] = NLVL;
count[2] = NLAT;
count[3] = NLON;
start[1] = 0;
start[2] = 0;
start[3] = 0;

/* Read and check one record at a time. */
for (rec = 0; rec < NREC; rec++) {
  start[0] = rec;
  nc_get_vara_float(ncid, pres_varid, start,
          count, &pres_in[0][0][0])));
  // process data
}
```

Java

```
int[] origin = new int[4];
int[] shape= precVar.getShape();
shape[0] = 1; // only one rec per read

// loop over the rec dimension
for (int rec = 0; rec < shape[0]; rec++) {
  origin[0] = rec;  // read this index
  Array presArray = presVar.read(origin, shape);
  // process data
}
```

“Index Space” Data Access

OPeNDAP

http://motherlode.ucar.edu:8080/thredds/dodsC/
NAM_CONUS_80km_20081028_1200.grib1.dods?
Precipitable_water[5][5:1:30][0:2:77]

Netcdf-Java

NetcdfFile ncfile = NetcdfFile.open(“NAM_CONUS_80km_20081028_1200.grib1”);

Variable v = ncfile.findVariable(“**Precipitable_water”**);

Array data = v.read(“**(5)(5:30)(0:77:2)”**);

“Coordinate Space” Access

```
float gridData(t,lev,lat,lon);
  float t(t);      // coordinate variables
  float lev(lev); // must be monotonic
  float lat(lat);
  float lon(lon);
```

```
 float[] level = new float[]{23.6, 78.0};
 float[] lat = new float[]{40.0, 42.0};
 float[] lon = new float[]{-103.6, -78.0};

 for (float time : timeCoords) {
  Array data=gridData.read(time,level,lat,lon);
  // process
}
```

“Georeferenced” access

Netcdf Subset Service:

http://motherlode.ucar.edu:8080/thredds/ncss/grid/
NAM_CONUS_80km_20081028_1200.grib2?
var=Precipitable_water&
time=2008-10-28T12:00:00Z&
north=40&south=22&west=-110&east=-80

WMS:

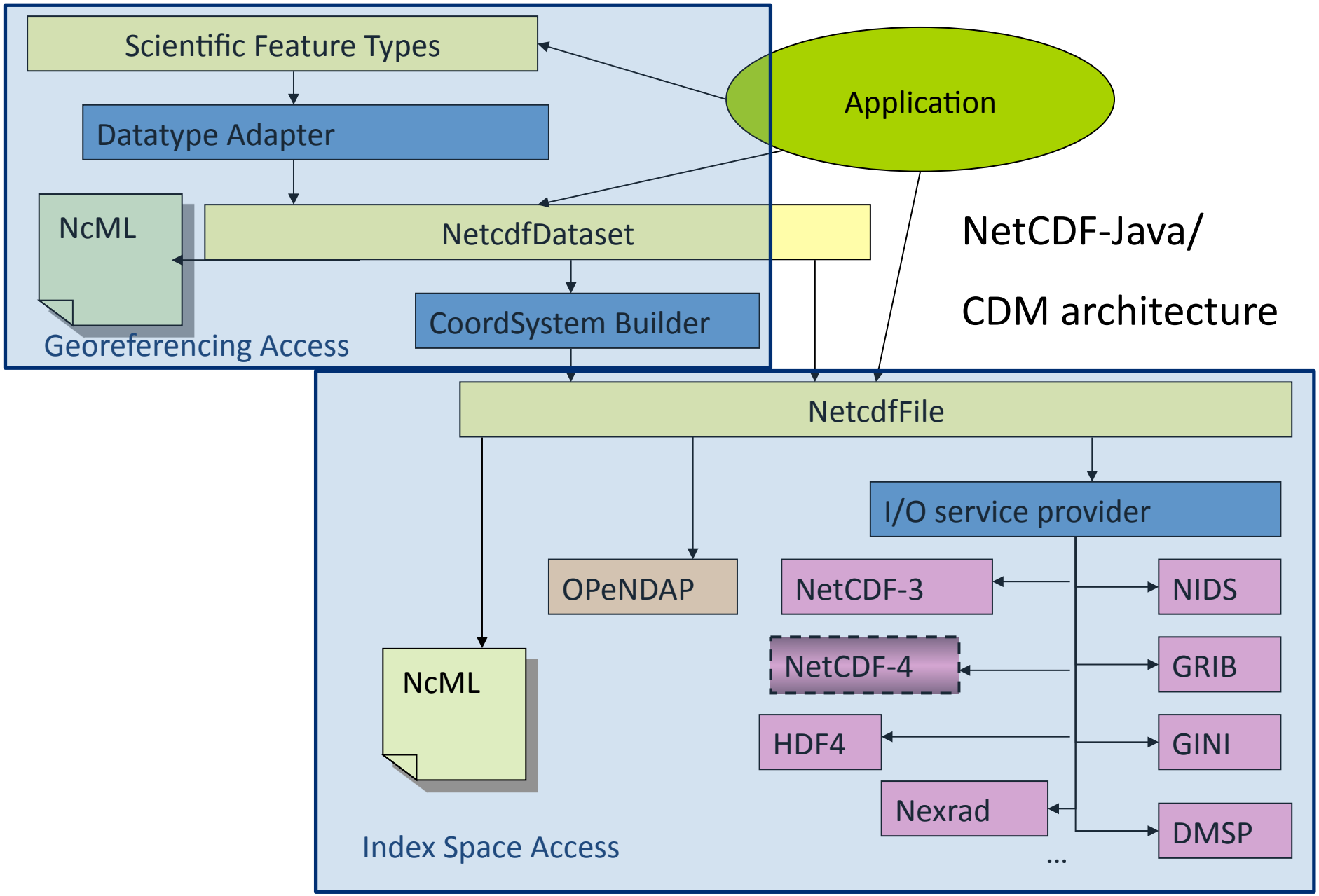
http://motherlode.ucar.edu:8080/thredds/wms/
NAM_CONUS_80km_20081028_1200.grib2?
SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&
CRS=EPSG:4326&WIDTH=800&HEIGHT=400&FORMAT=image/png&
STYLES=boxfill/redblue&COLORSCALERANGE=-10,10&
LAYERS=**Precipitable_water**&
BBOX=**-40,22,-110,-80**&
TIME=**2008-10-28T12:00:00Z**

TimeSeries “Feature Type”

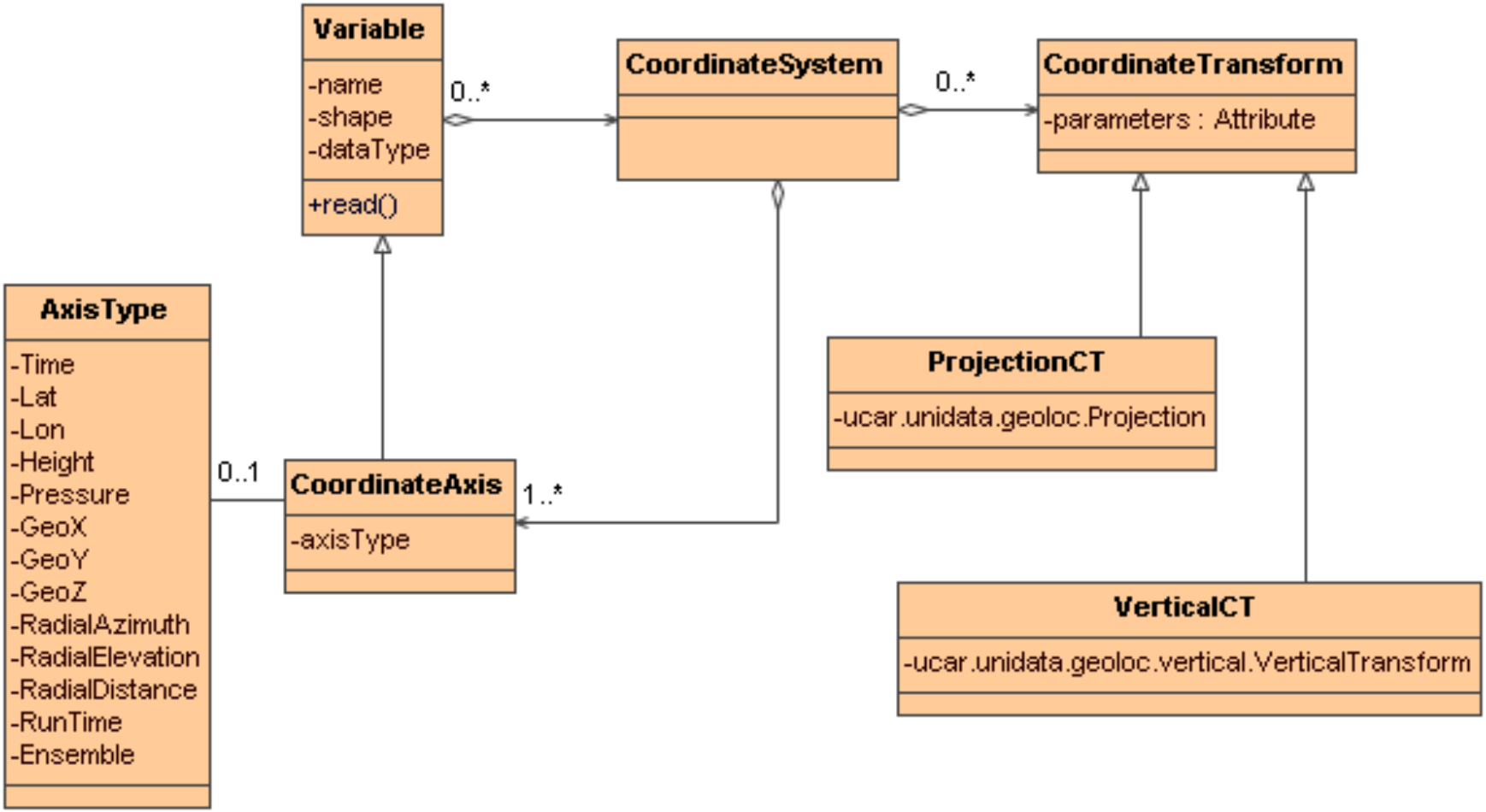
```
LatLonRect bb=new LatLonRect(new LatLonPointImpl(40.0, -105.0),
                             new LatLonPointImpl(42.0, -100.0));
Date start= DateFormatter.getISODate(“2010-10-12T00:00:00”);
Date end  = DateFormatter.getISODate(“2010-10-22T12:00:00”);
DateRange dr = new DateRange(start, end);

PointFeatureCollection points = stationTimeSeriesCollection.subset
   (bb, range);

while( points.hasNext() ) {
  PointFeature pointFeature = points.next();
  ...
}
```

Grid “Feature Type”

```
LatLonRect bb = new LatLonRect(new LatLonPointImpl(40.0, -105.0),
                               new LatLonPointImpl(42.0, -100.0));
Date start= DateFormatter.getISODate(“2010-10-12T00:00:00”);
Date end  = DateFormatter.getISODate(“2010-10-22T12:00:00”);
DateRange dr = new DateRange(start, end);

Array data = grid.subset(bb, dateRange);
while(data.hasNext() ) {
  double d = data.next();
  ...
}
```


Scientific Feature Types
Datatype Adapter
NcML
NetcdfDataset
CoordSystem Builder
Georeferencing Access
Application
NetCDF-Java/
CDM architecture
NetcdfFile
I/O service provider
OPeNDAP
NetCDF-3
NIDS
NetCDF-4
GRIB
NcML
HDF4
GINI
Nexrad
DMSP
...
Index Space Access

Coordinate System UML



Netcdf-Java Library parses these Conventions

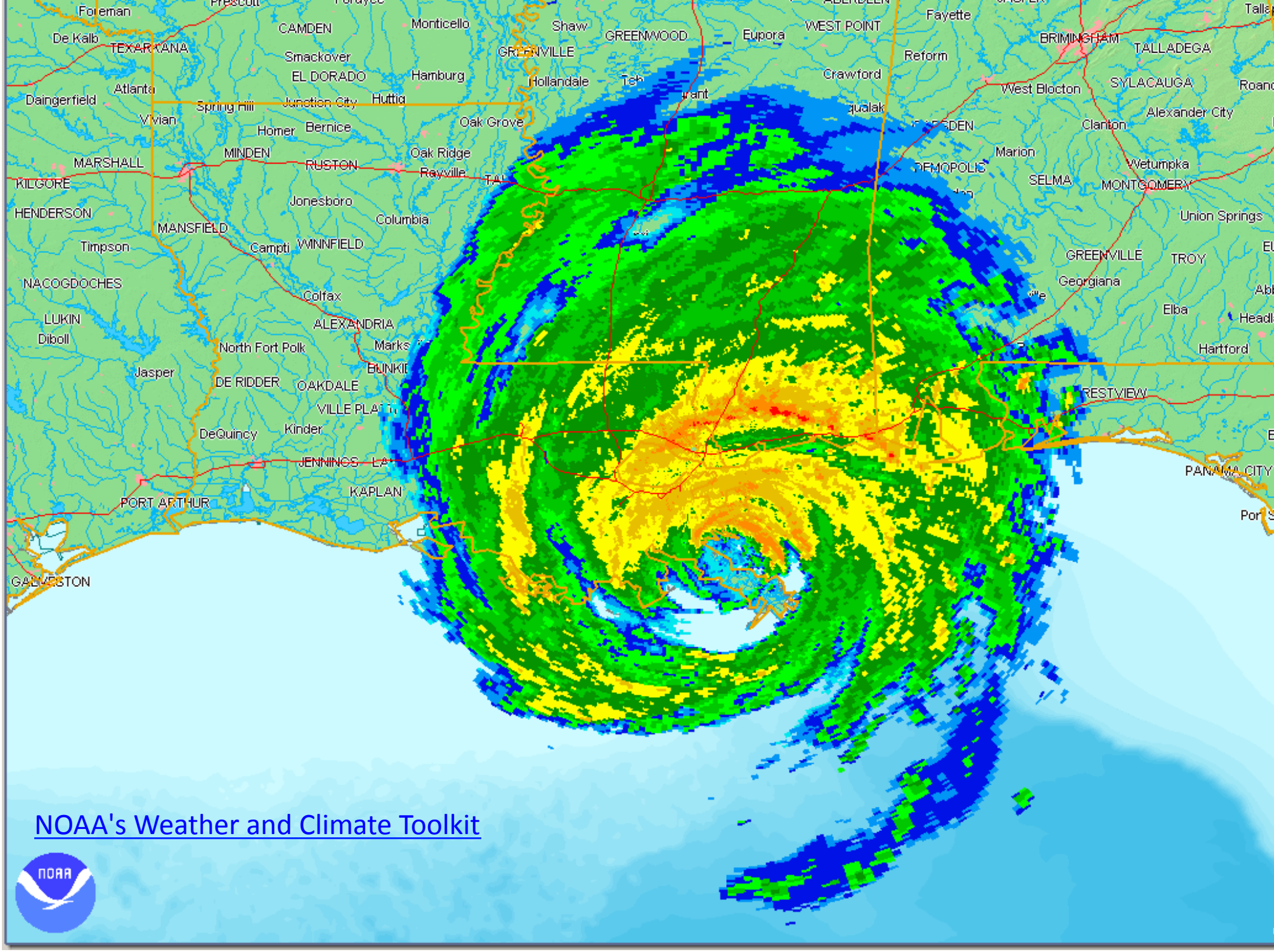
- CF Conventions (preferred)
- COARDS, NCAR-CSM, ATD-Radar, Zebra, GEIF, IRIDL, NUWG, AWIPS, WRF, M3IO, IFPS, ADAS/ARPS, MADIS, Epic, RAF-Nimbus, NSSL National Reflectivity Mosaic, FslWindProfiler, Modis Satellite, Avhrr Satellite, Cosmic, ….
- Write your own *CoordSysBuilder* Java class

Projections (CF)

- albers_conical_equal_area
- lambert_azimuthal_equal_area
- lambert_conformal_conic
- mcidas_area
- mercator
- orthographic
- rotated_pole
- stereographic (including polar)
- transverse_mercator
- UTM (ellipsoidal)
- vertical_perspective

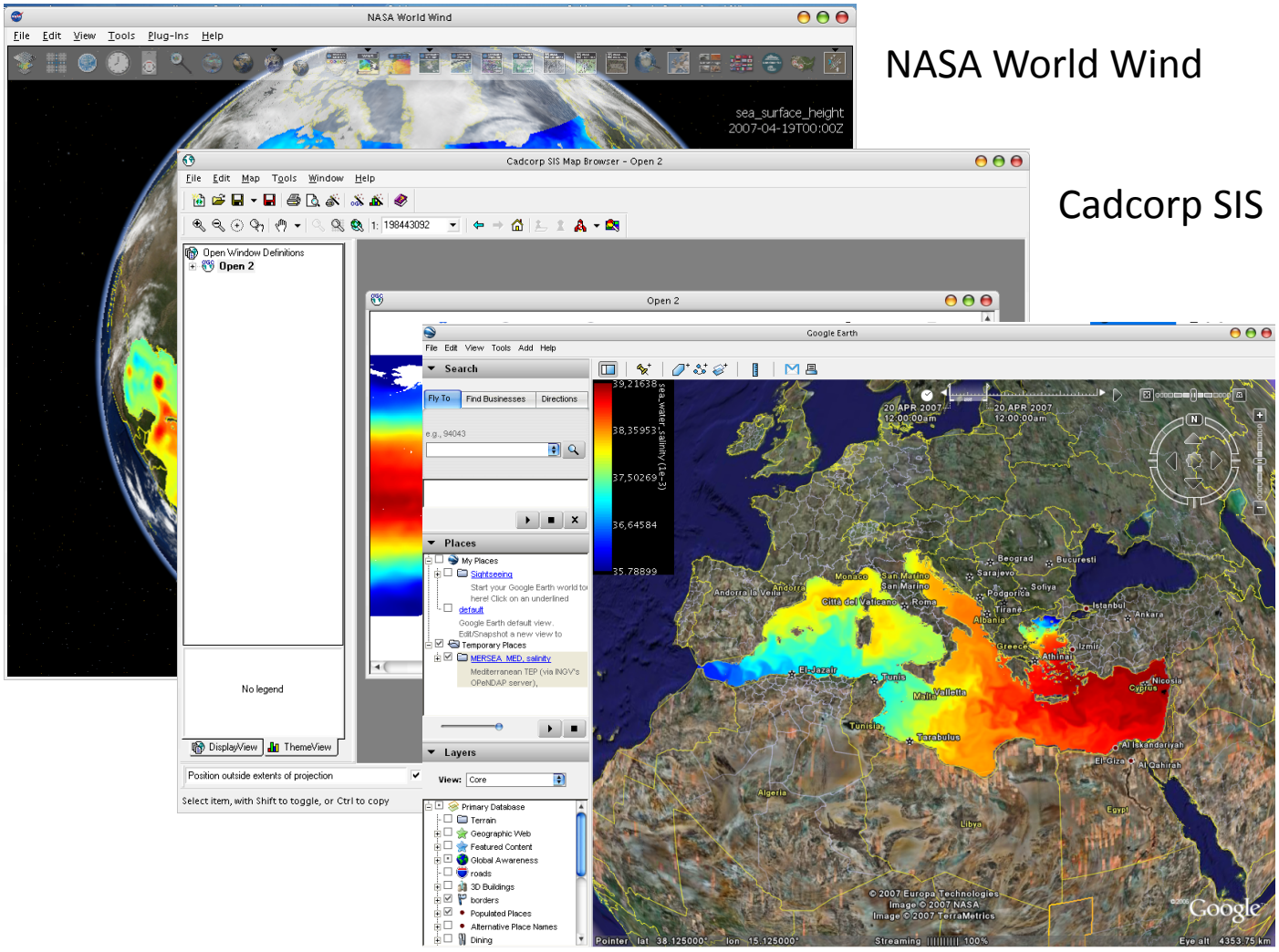
Vertical Transforms (CF)

- atmosphere_sigma
- atmosphere_hybrid_sigma_pressure
- atmosphere_hybrid_height
- ocean_s
- ocean_sigma
- existing3DField



NOAA's Weather and Climate Toolkit

WMS Clients



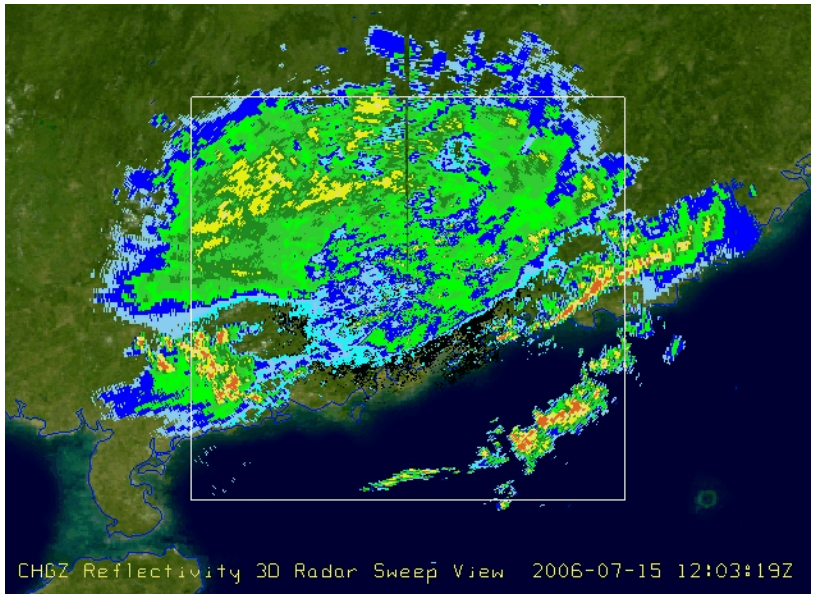
NASA World Wind

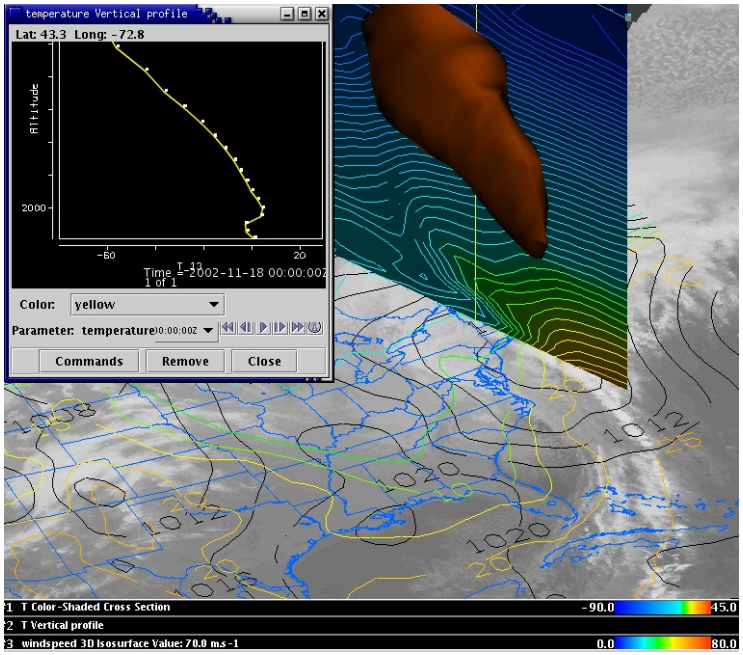
Cadcorp SIS

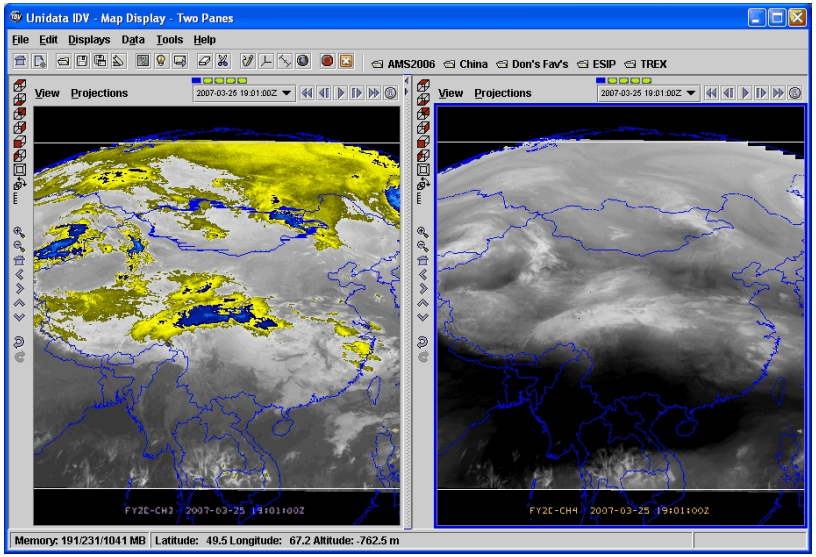
Google Earth

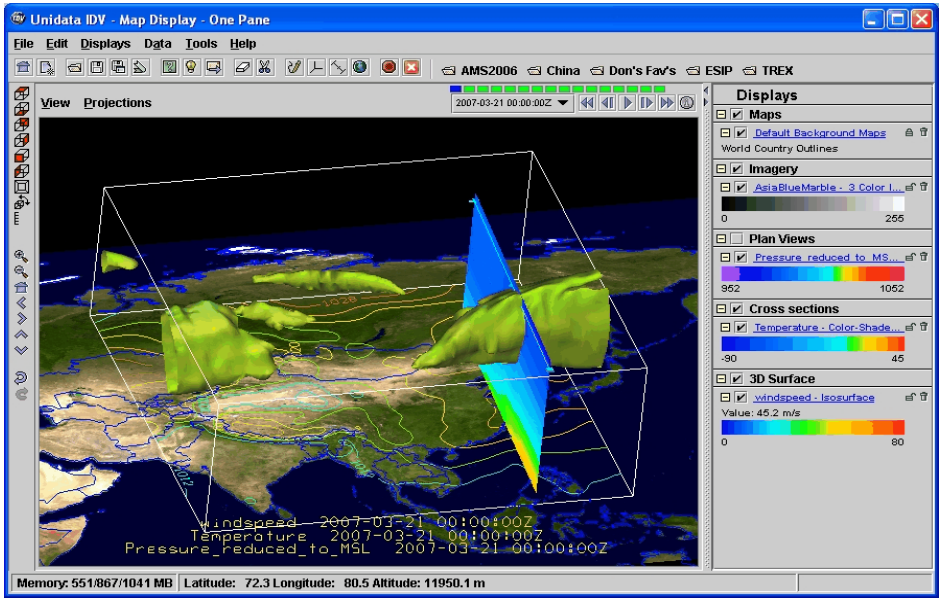
3rd-party clients can't use the custom WMS extensions

Unidata's Integrated Data Viewer

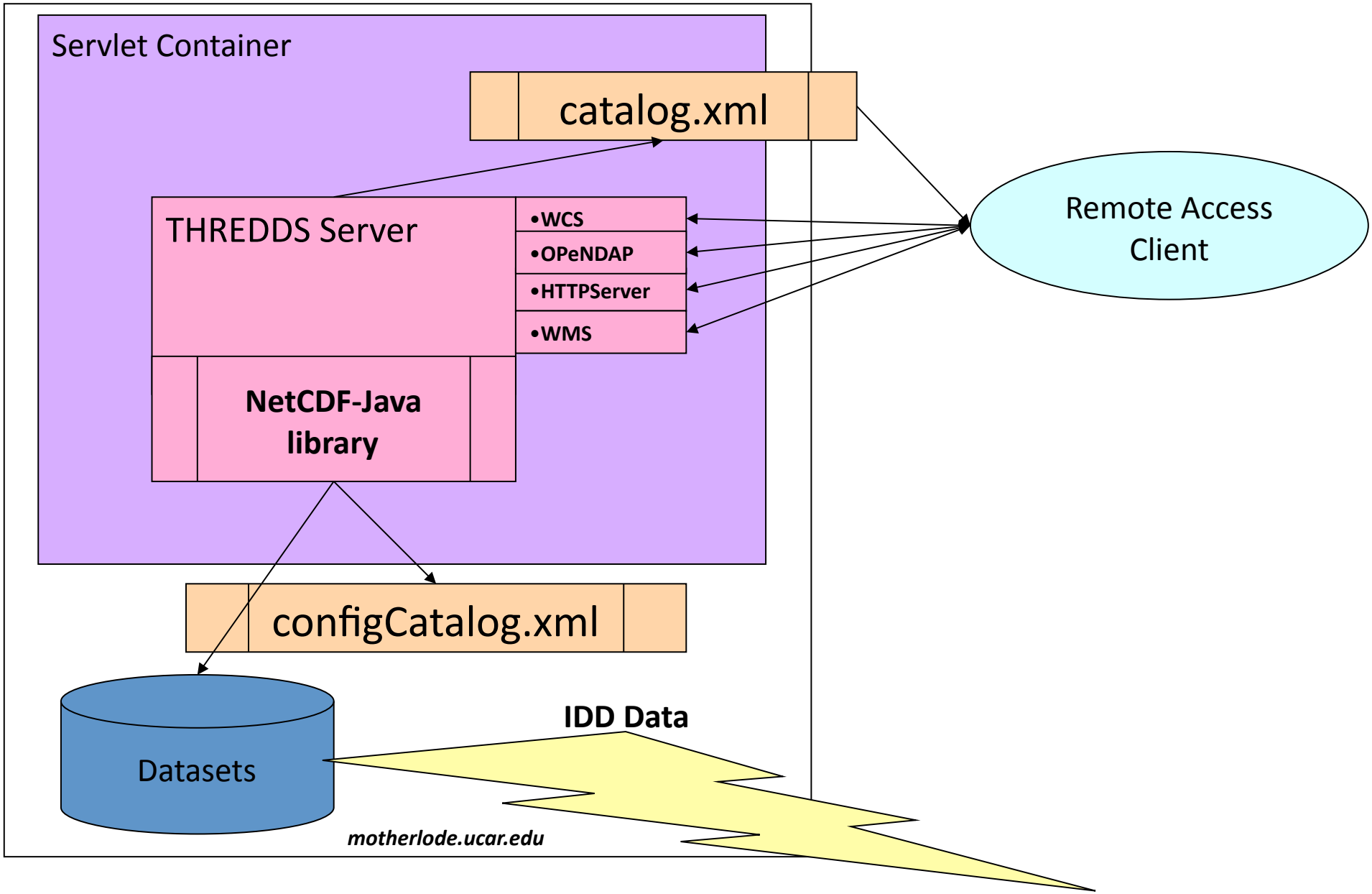








THREDDS Data Server



THREDDS Data Server (TDS)

- Web server for scientific data
- Provides remote data access
  - HTTP file transfer
  - OPeNDAP (index space subsetting)
  - Open Geospatial Consortium (OGC) WMS and WCS (coordinate space subsetting)
  - Experimental data access protocols
    - NetCDF Subset Service
    - CdmRemote

OGC Web Map Service

- WMS 1.3
- Jon Blower's (Reading, UK) ncWMS integrated with TDS
- Coordinate Space subsetting
- Produces JPEG output
- Fast generation of images
- Reproject images into large number of coordinate systems (uses GeoToolkit)

OGC Web Coverage Service

- WCS 1.0
- Coordinate Space subsetting
- Returns data, not pictures
  - GeoTIFF floating point, grayscale
  - NetCDF-CF
- No reprojections, resamplings
- Restricted to CDM files that have Grid coordinate system
  - evenly spaced x,y

NetCDF Markup Language (NcML)

- Modify ("fix") existing datasets without rewriting them
- Often its possible to add missing coordinate information that allows georeferencing by standard tools

NetCDF-Java Summary

- NetCDF-Java Library is widely used, because:
  - Handles multiple file formats, single API
  - Grungy details of Coordinate System encodings
    - So many standards to choose from!
  - NcML allows dataset modification without rewrite
  - NcML aggregation for virtual datasets
  - Remote access is built in

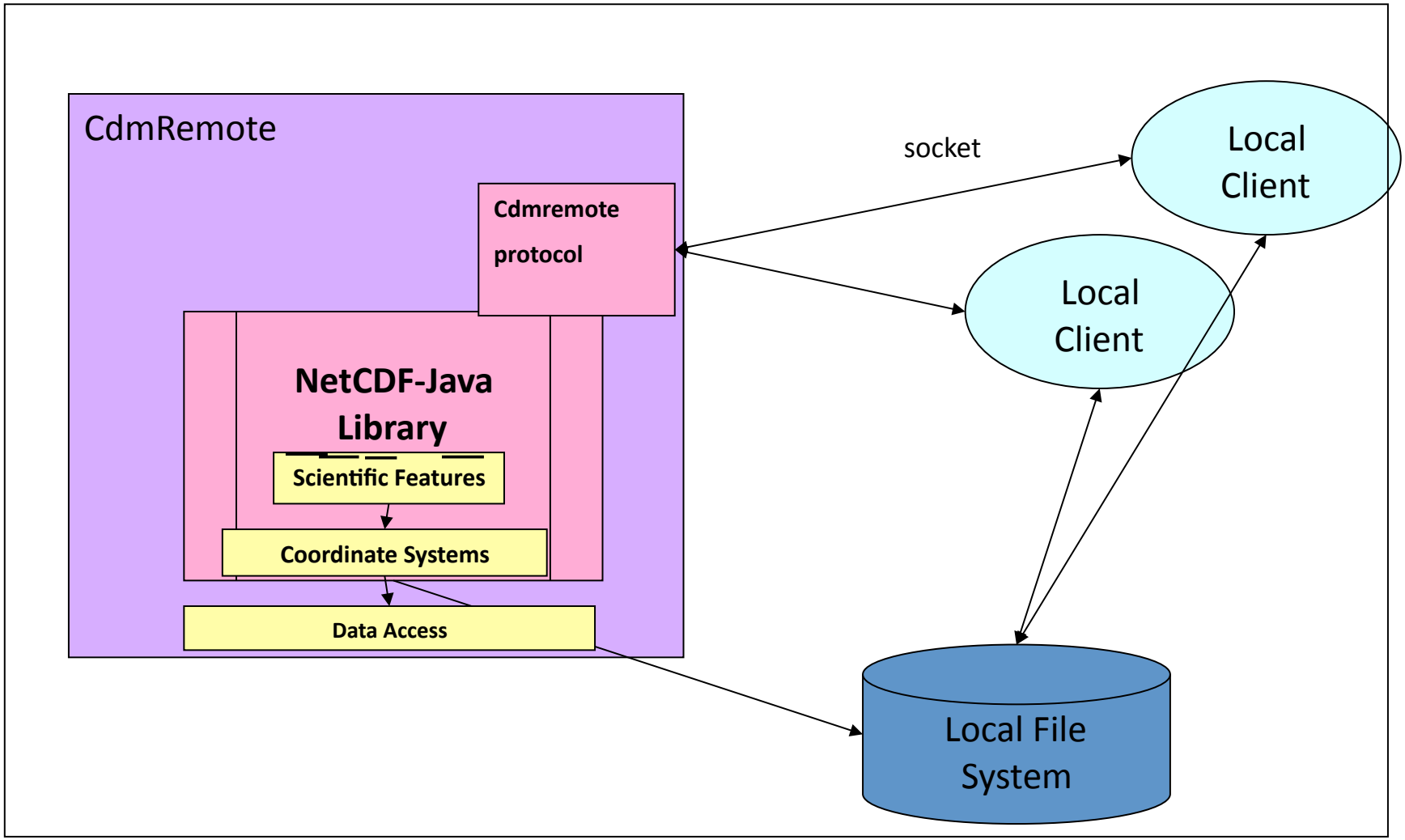
Issues

- HDF5 != netCDF-4
  - Should it?
- OPeNDAP hasn't delivered DAP4
  - Can't transport NetCDF-4 extendd data model
- Build libraries on top of data access libraries
  - Coordinate systems: standardize?
  - Feature Types : Grids, Swaths, Radial, Discrete Sampling, Unstructured Grids, Polylines

How to get this functionality into netCDF C library?

- Java has to run in its own process, cant be linked into C
- Reimplement CDM functionality in C library
  - 200K+ LOC
  - OO, inheritence
  - OTOH, avoid blind alleys, 3rd party libraries
- Leave Java in its own process, communicate across processes

CdmRemote Server



Possibility: CdmRemote

- Lightweight server for CDM datasets
  - Zero configuration
  - Local filesystem
  - Cache expensive objects
- Uses ncstream for on-the-wire protocol, probably over HTTP
- Java and C clients
  - Allow non-Java applications access to CDM stack
  - Coordinate space queries
  - Virtual datasets
  - Feature Types

Possibility: ncstream

- Write-optimized encoding of CDM datasets
  - append only
- Streaming data across network
- Binary encoding using Google's Protobuf
  - Efficient, extensible
- Ncstream <--> NetCDF4

Summary

- We should build higher level abstractions on top of existing data access libraries
- Standardize semantic conventions for coordinate systems, discovery metadata, etc.
- Deal with very large collections of scientific data files
- Keep pushing common tasks into libraries, let scientists do science


Thank You
netCDF
One interface to bind them