Maintaining netCDF:
Updating Java Tutorial Code and Performance Testing in Python

Unidata 2021 Summer Internship

Isabelle Pfander

Project Motivation

- Working with the netCDF library
- Promoting education
- Learn about storage and efficiency

Maintaining a Codebase

- Documentation & tutorials
- Adjusting to new technology

Documentation

Updated netCDF-Java user guide docs for 6.x and 5.x

Coordinate Systems

- Coordinate attribute convention & examples
- Coordinate transforms

Scientific Feature Types

- Grid datastes
- Coverage datasets

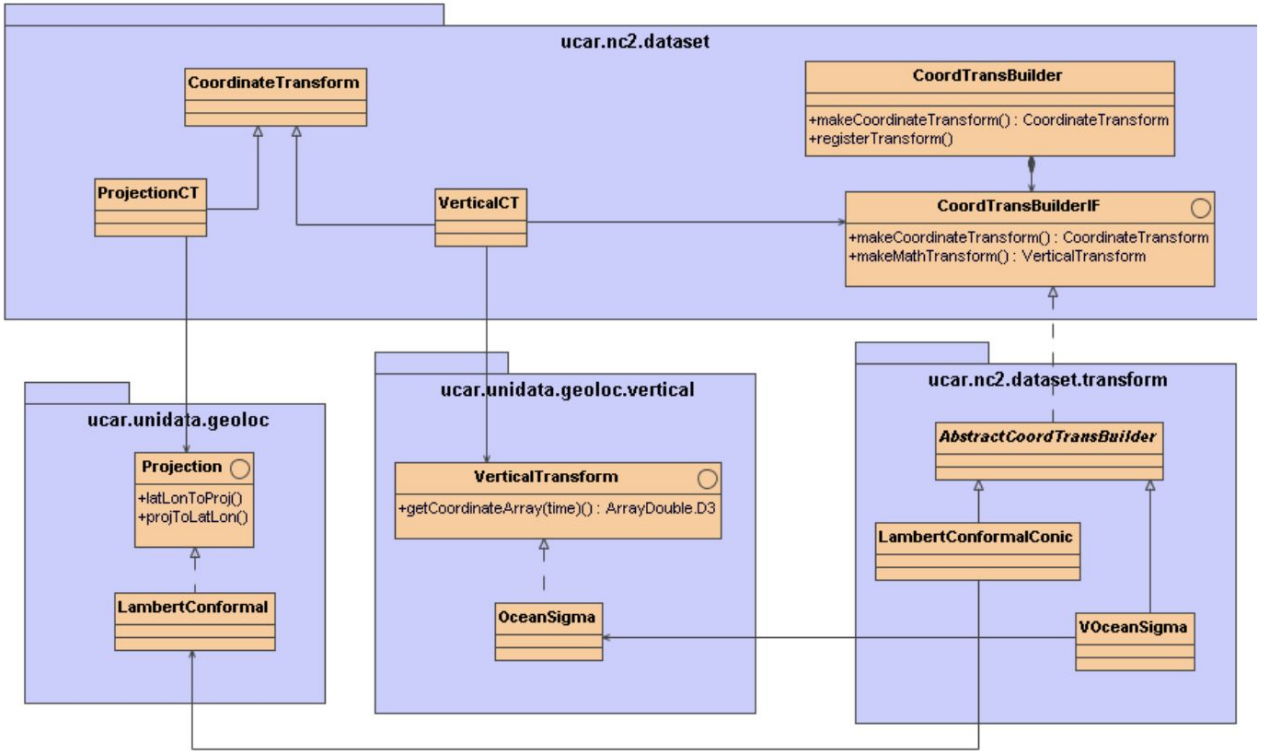
Runtime Configuration

- Runtime loading
- System properties
- netCDF C library

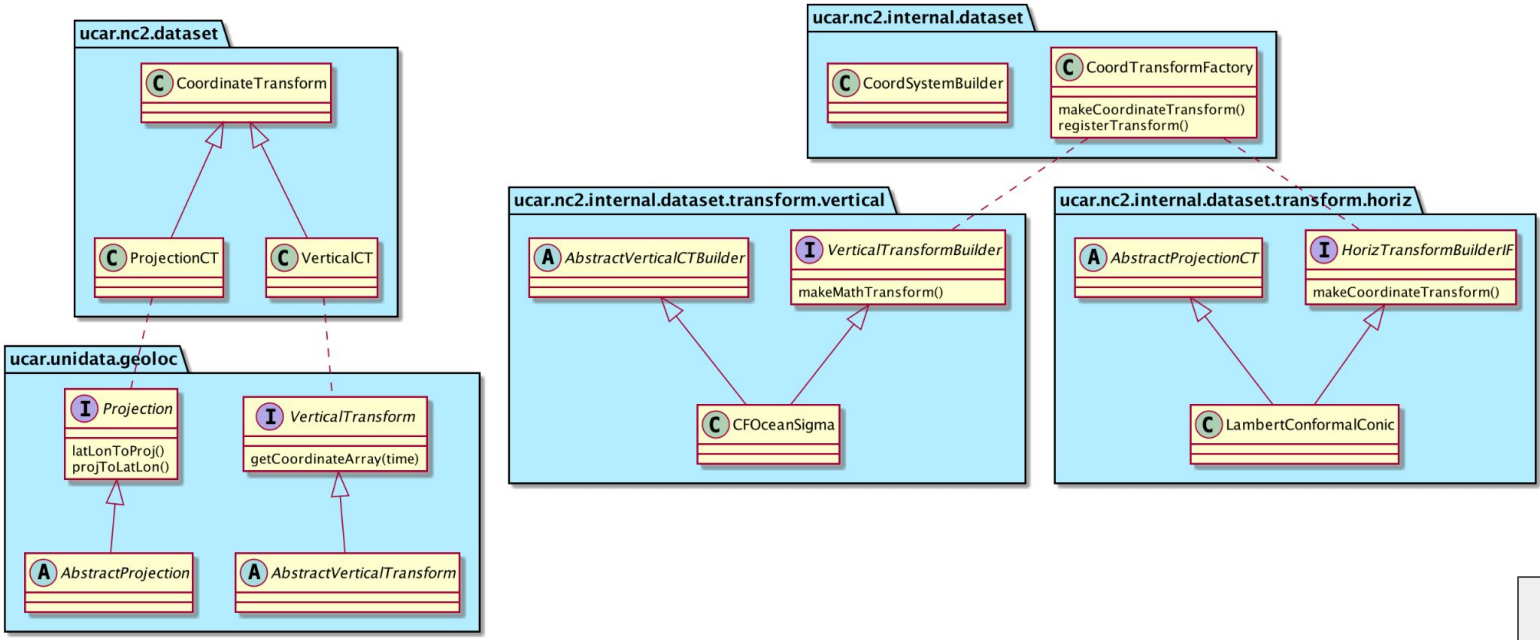
Documentation Writing Changes

- Raw HTML to Markdown
- Format Updates
- Linking to relevant sites
- Updating images

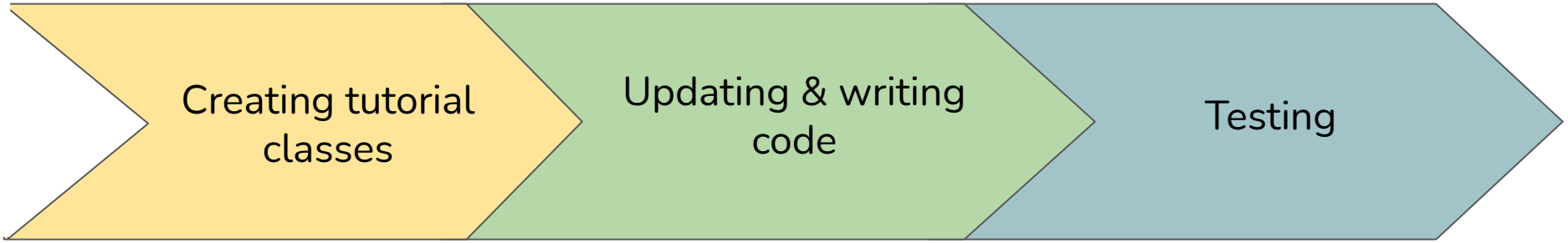
Documentation: before



Documentation: after



Tutorial Code



Tutorial Code

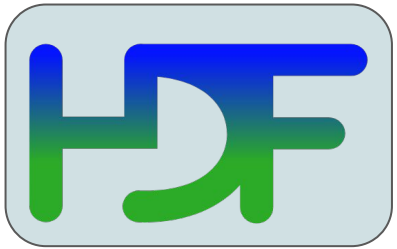
```
- You can use the <b>_GridCoordSystem_</b> to find the value of a grid at a specific lat, lon point:

- ~~~
-   // open the dataset, find the variable and its coordinate system

-   GridDataset gds = ucar.nc2.dt.grid.GridDataset.open(location);
-   GridDatatype grid = gds.findGridDatatype( "myVariableName");
-   GridCoordSystem gcs = grid.getCoordinateSystem();
-

-   double lat = 8.0;
-   double lon = 21.0;
-
-   // find the x,y index for a specific lat/lon position
-   int[] xy = gcs.findXYindexFromLatLon(lat, lon, null); // xy[0] = x, xy[1] = y
```

```
+ You can use the `GridCoordinateSystem` to find the indices and coordinates of the 2D grid from the (x,y) projection point:

+ {% capture rmd %}
+ {% includecodeblock netcdf-java&docs/userguide/src/test/java/examples/featuretypes/GridDatasetsTutorial.java&findLatLonVal %}
+ {% endcapture %}
+ {{ rmd | markdownify }}
+
+ Most `GridCoordinateSystems` have a `CoordinateAxis1DTime` time coordinate. If so, you can get the list of dates from it.
```

Separate Java class

```
+     public static void findLatLonVal(String yourLocationAsString, Formatter errlog, double xPointAsDouble, double yPointAsDouble) throws IOException {
+         // open the dataset, find the variable and its coordinate system
+         GridDataset gds = ucar.nc2.grid.GridDatasetFactory.openGridDataset(yourLocationAsString, errlog);
+         GridHorizCoordinateSystem gcs = (GridHorizCoordinateSystem) gds.getGridCoordinateSystems();
+
```

Performance Testing



File format for netCDF-4

Python-based storage format

Benchmark Tests

netCDF-3
(no compression or chunking)

netCDF-4
(zlib compression)

Zarr directory store
(Blosc compression)

Reading with Xarray
(netCDF-4 and Zarr)

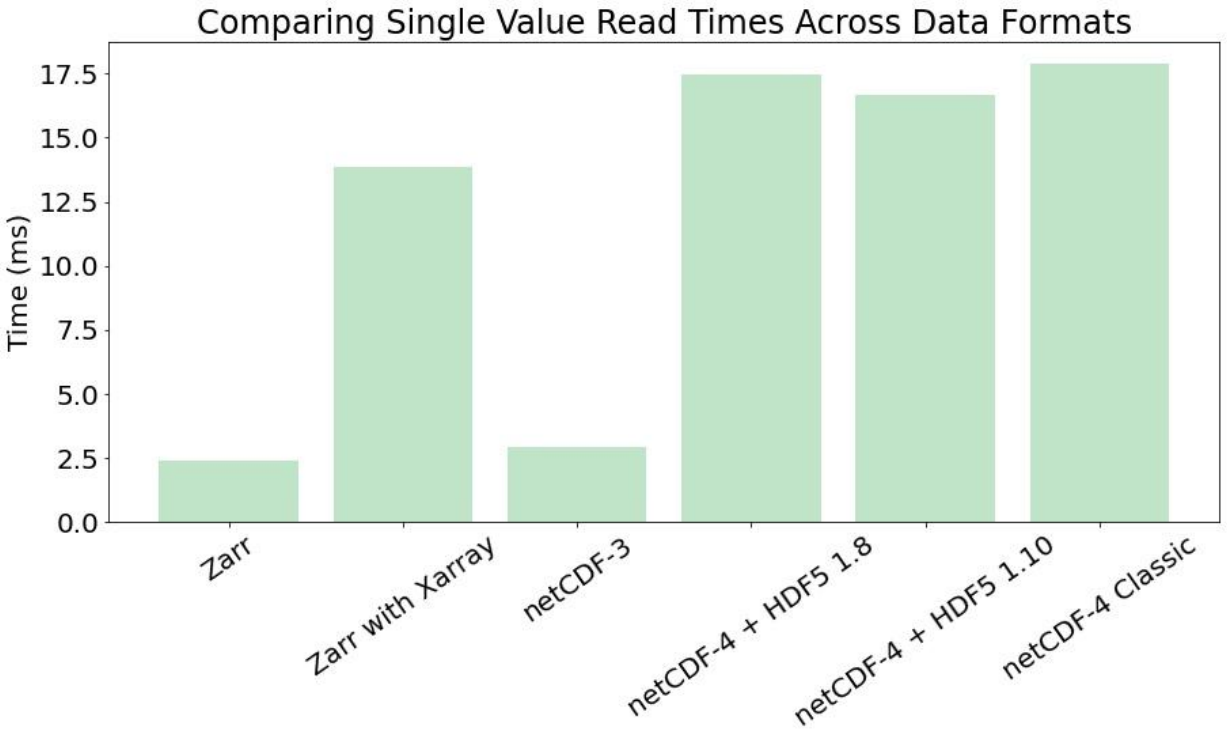
netCDF-4 Classic
(zlib compression)

Benchmark Tests

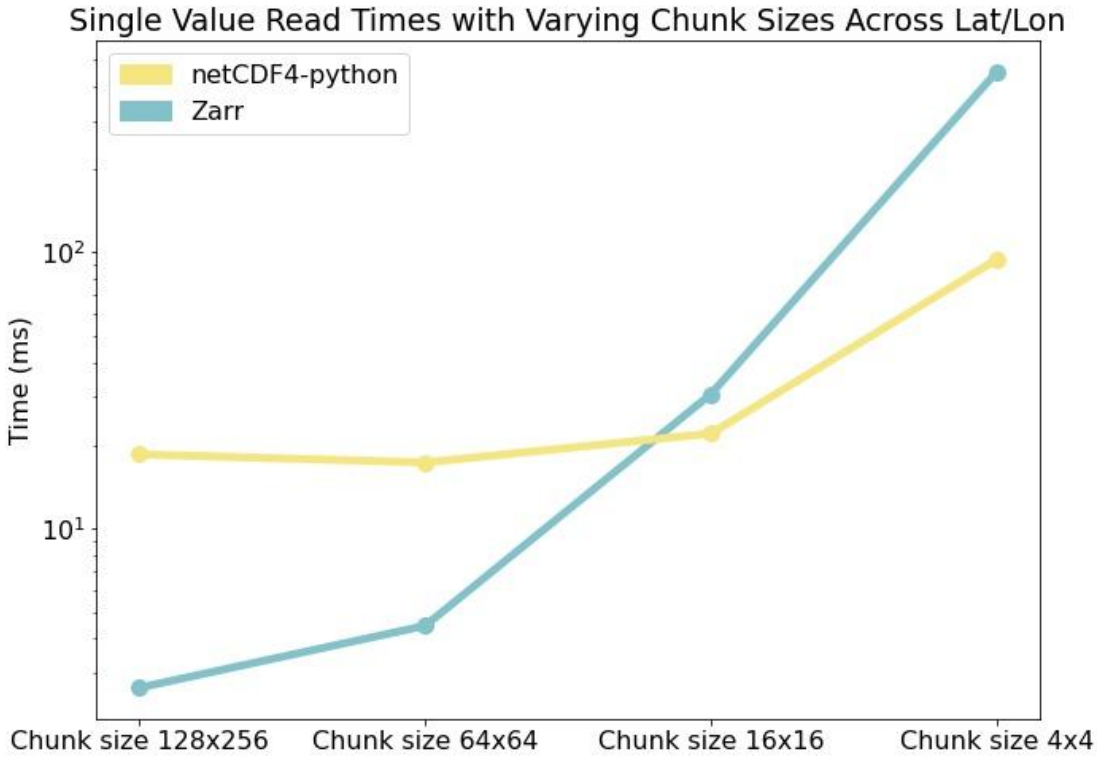
Timing netCDF-4

```
import timeit

# import before timeing begins
imports = "import netCDF4 as nc"

# reading code to be timed
nc4_single_values = '''

# netCDF-4 with zlib compression
dataset = nc.Dataset('Data/outputNetcdf-4.nc')

# reading single values
tas = dataset.variables['tas'][0][0][0] # air temperature
plev = dataset.variables['plev'][0] # air pressure
pr = dataset.variables['pr'][0][0][0] # precipitation flux
ua = dataset.variables['ua'][0][0][0][0] # eastward wind with 17 levels of plev
lons = dataset.variables['lon'][0]
lat = dataset.variables['lat'][0]

dataset.close()
'''

# timeit statement in seconds average of 1000
nc4_single_values_time = (timeit.timeit(setup = imports,
                     stmt = nc4_single_values,
                     number = 1000)) / 1000
```

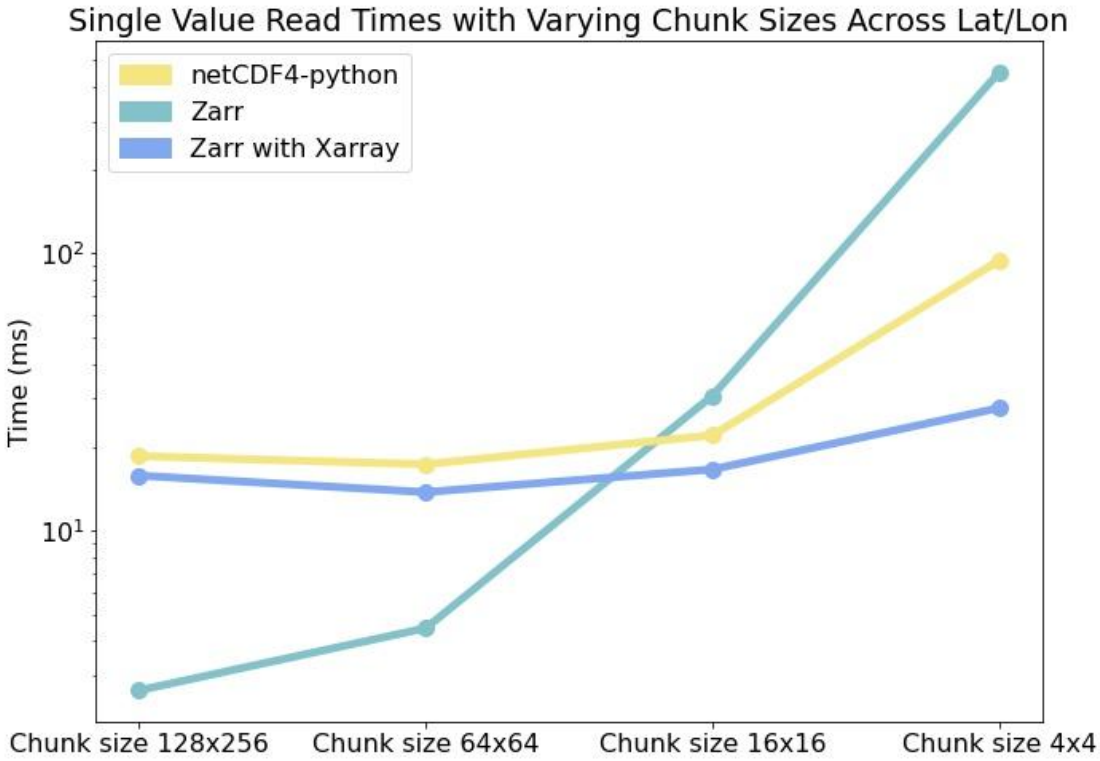
Results


Comparing Single Value Read Times Across Data Formats
Time (ms)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Zarr
Zarr with Xarray
netCDF-3
netCDF-4 + HDF5 1.8
netCDF-4 + HDF5 1.10
netCDF-4 Classic

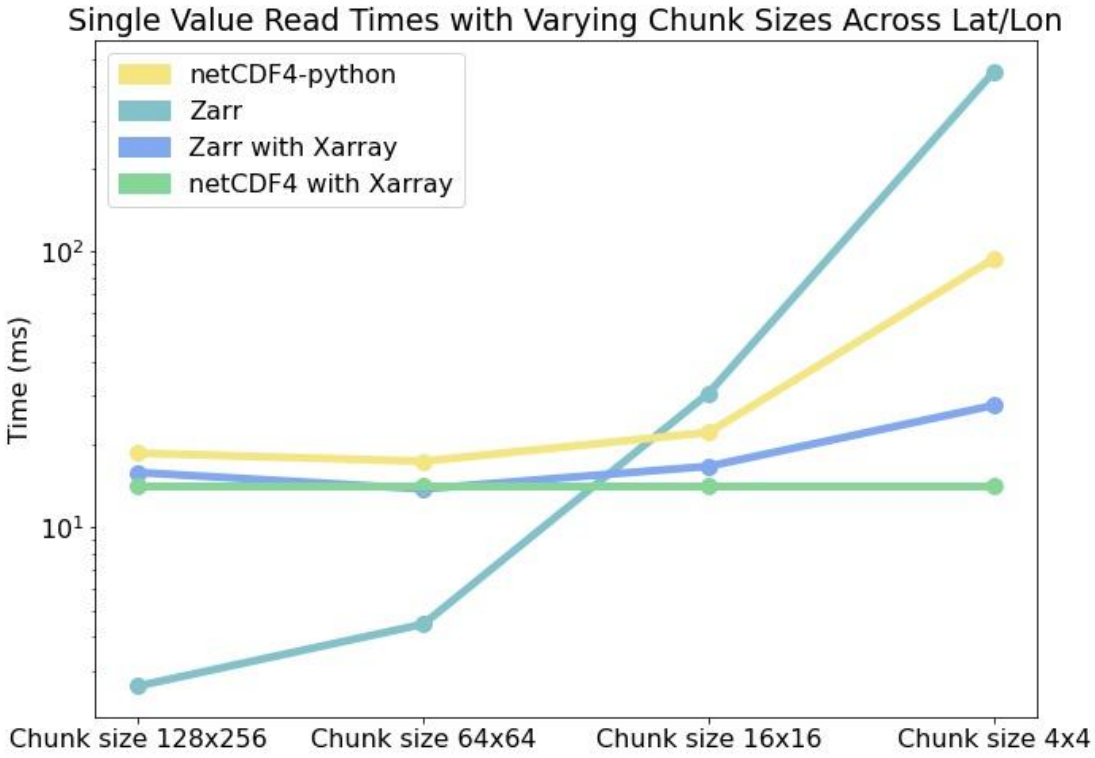
Results


Single Value Read Times with Varying Chunk Sizes Across Lat/Lon
netCDF4-python
Zarr
Time (ms)
10^2
10^1
Chunk size 128x256
Chunk size 64x64
Chunk size 16x16
Chunk size 4x4

Results


Single Value Read Times with Varying Chunk Sizes Across Lat/Lon
netCDF4-python
Zarr
Zarr with Xarray
Time (ms)
10^2
10^1
Chunk size 128x256
Chunk size 64x64
Chunk size 16x16
Chunk size 4x4

Results


Single Value Read Times with Varying Chunk Sizes Across Lat/Lon
netCDF4-python
Zarr
Zarr with Xarray
netCDF4 with Xarray
Time (ms)
10^2
10^1
Chunk size 128x256
Chunk size 64x64
Chunk size 16x16
Chunk size 4x4

Results

- Zarr directory store
  - The more chunks, the more individual files in the directory store
  - Opening ~2,000 files per variable
  - More time spent opening than reading
- netCDF-4
  - Only one open is needed
  - More seek operations to find the data
- Xarray
  - Leaves data unloaded until load is specified

Next Steps

- Compare reads for netCDF-Java HDF5 & Zarr implementations
- Poster at AGU Fall Meeting

Maintaining netCDF:
Updating Java Tutorial Code and Performance Testing in Python

UCAR COMMUNITY PROGRAMS

Isabelle Pfander
Mentors: Hailey Johnson, Sean Arms
Supervisor: Ryan May

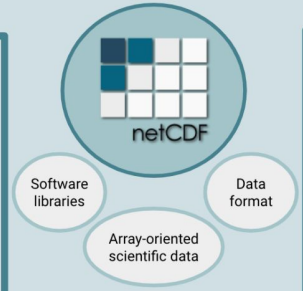
NSF unidata

1. Updating netCDF Tutorial Code

Network Common Data Form (netCDF) is a combination of software libraries and APIs describing a data model for scientific multidimensional arrays.

Maintaining this codebase requires documentation, user support, testing, and adapting to new technologies. I maintained netCDF-Java documentation by updating Java tutorial code, testing code snippets, and modernizing tutorial texts to improve user understanding.

Scan the QR code to view online tutorials



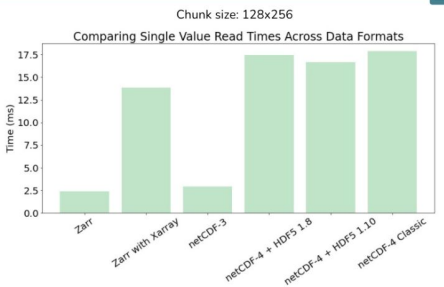
2. Comparing Data Formats

HDF5 is a file format used by netCDF-4 providing compression and chunking to the netCDF data model. Zarr is a Python-based storage format, which has support in netCDF-C and will soon in netCDF-Java.

This project compares read times for the data formats below:

- netCDF-3: no compression or chunking
- netCDF-4: zlib compression
- netCDF-4 Classic: zlib compression
- Zarr: Blosc compression
- Zarr & netCDF read with Xarray package

3. Benchmarking Results

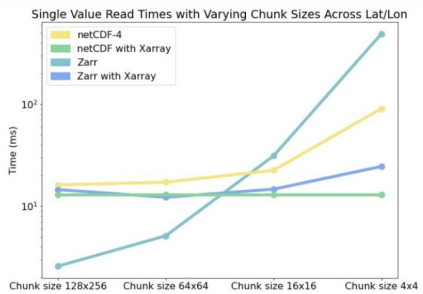


A netCDF-4 file stores all data in one .nc file, consequently more operations are needed to find the appropriate data, but only one open is required.

Zarr directory stores save chunked data as many subdirectories and files. The more chunks, the more individual files in one Zarr directory store. With small chunk size, this resulted in more time spent opening than reading.

In some cases, the Xarray Python package can read faster with both netCDF-4 and Zarr directory stores until loading is specified.

Note: read speeds will vary on an object store. These comparisons are for a posix file system only.



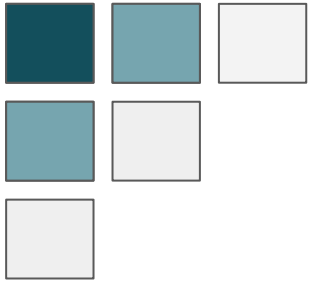
Future Work

- Compare reads for netCDF-Java HDF5 and Zarr implementations
- Test on datasets with varying dimensions and size

Scan to view code

Acknowledgements

Thank you to Unidata for the mentorship and community during this internship. This work was funded by the National Science Foundation (Grant AGS-1901712).



Thank you