Leveraging FAIR principles for efficient management of meteorological radar data

Alfonso Ladino[1], Max Grover[2], Stephen Nesbitt[1], Kai Mühlbauer[3],

[1]Department of Climate, Meteorology and Atmospheric Sciences, University of Illinois at Urbana-Champaign, Urbana, IL, USA
[2]Argonne National Laboratory, Lemont, IL, USA
[3] University of Bonn, Bonn, Germany

INTRODUCTION

Near Real Time → Offline

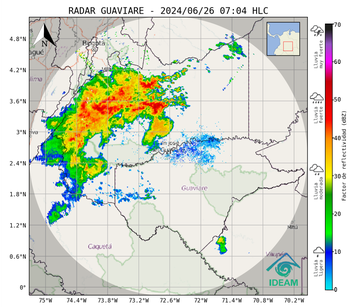
Carimagua Radar (Colombia)

Offline radar products demand **extensive input-output (I/O)** operations over data stored in proprietary (binary) formats

MOTIVATION

- Time-series data model to **arrange**, **manage**, and **store** radar data in cloud-storage buckets efficiently using **Analysis-Ready Cloud-Optimized** (ARCO) format [1].
- Use a **hierarchical tree** structure based on the Climate and Forecast (CF) format-based **FM301** (World Meteorological Organization) **[2].**
- Align with the **open data** paradigm, emphasizing the **FAIR** principles (**Findable, Accessible**, **Interoperable**, **Reusable**)

DATA

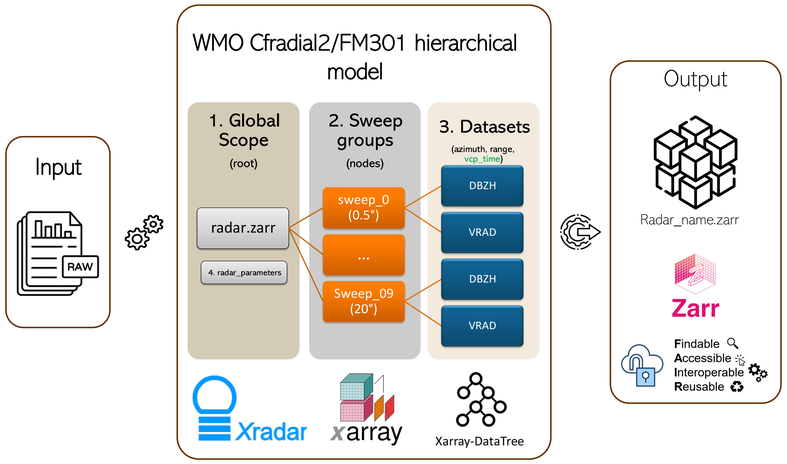


Guaviare Radar (Colombia)

- 10 elevations (0.5 to 20 degrees)
- 1 moth of consecutive data (from 08/01/2022 to 08/31/2022)
- Sigmet files (Binary format)
- 5-min VCP
- Data currently available at: https://registry.opendata.aws/ideam-radares/

METHODS

1. Hierarchical tree-like radar data model (time series)



https://github.com/aladinor/raw2zarr

2. Testing on radar products

a. Quasi-Vertical Profile (QVP) [3]

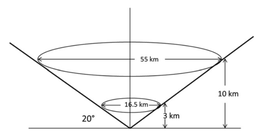


FIG. 2. Conical volume representing azimuthally averaged quasi-vertical profiles of radar variables.

b. Radar Quantitative Precipitation Estimation (QPE)

$$Z = 200R^{1.6}$$ Marshall & Palmer (1948) [4]

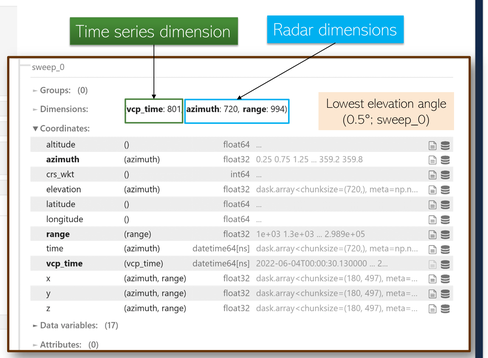
$$R[\mathrm{mm/hr}] = \left[\frac{Z[mm^6m^{-3}]}{200}\right]^{\frac{1}{1.6}}$$

RESULTS

Pythonic representation of the WMO Cfradial2/FM301 standard data model

```
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
import xradar as xd
import pandas as pd
from xarray.backends.api import open_datatree
from dask.distributed import Client, LocalCluster
import warnings
warnings.filterwarnings('ignore')

cluster = LocalCluster()

path_datatree = "D:/Alfonso/python/raw2zarr/zarr/Guaviare_V2.zarr"

radar_dtree = open_datatree(path_datatree,
                            engine='zarr',
                            chunks={})

radar_dtree

datatree.DataTree
 Groups: (11)
 Dimensions:
 Coordinates: (0)
 Data variables: (8)
 Attributes: (9)

print(f'radar datatree size: {radar_dtree.nbytes / (1024 ** 3)} GB')

radar datatree size: 99.72293827682734 GB
```

radar_parameters,
sweep_0 (0.5°),
sweep_1 (1.5°),
...,
sweep_09 (20°)



The tree-like data model encompasses **all sweeps** set up within the radar operation and the "radar_parameters" group. Each **'sweep_xx'** includes a **dataset** with **'azimuth'** and **'range'** as **radar dimensions** and **coordinates**. The additional **'vcp_time'** dimension enables the dataset to represent a **time series**.

RESULTS (Cont.)

Quasi-Vertical Profile (QVP)

Accessing the 20° elevation angle Dataset

```
ds_20deg = radar_dtree['sweep_9'].ds
```

Taking the mean for the radar reflectivity (DBZH) over the **azimuth** dimension

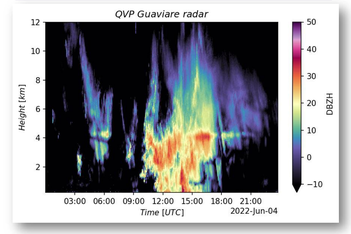
```
%%time
qvp = 10 * np.log10((10 ** (ds_20deg.DBZH / 10)).mean('azimuth'))
qvp = qvp.assign_coords(range=(qvp.range.values / 1000 *
                               np.sin(ds_20deg.sweep_fixed_angle.mean(skipna=True).values
                               * np.pi / 180. )))
```

CPU times: total: 719 ms
Wall time: 691 ms

Plotting the QVP for the 6/24/22

```
%%time
fig, ax = plt.subplots(figsize=(6,4), dpi=150)
qvp.sel(range=slice(0, 12)).sel(vcp_time='2022-06-04').plot(x='vcp_time',
                                                            y='range',
                                                            cmap='ChaseSpectral',
                                                            vmin=-10,
                                                            vmax=50,
                                                            ax=ax)
ax.set_ylabel(r"$Height \ [km]$")
ax.set_xlabel(r"$Time \ [UTC]$");
ax.set_title(r"$QVP \ Guaviare \ radar$");
fig.tight_layout()
plt.savefig('../results/Guaviare_QVP.jpeg',  bbox_inches='tight')
```

CPU times: total: 2.11 s
Wall time: 1.39 s



Radar QPE

Lowest elevation angle(0.5)

```
ds_05deg = radar_dtree['sweep_0'].ds
```

Rainfall rate accumulation

```
def rain_rate(z, a=200, b=1.6, dt=5):
    """
    Function that computes rain rate from the Marshall & Palmer (1948) relationship. This function also
    converts rain rates into rain depth by accumulating it at a given time delta
    """
    z_lin = 10 ** (z / 10)
    return ((1 / a) ** (1 / b) * z_lin ** (1 / b)) * (dt / 60)
```

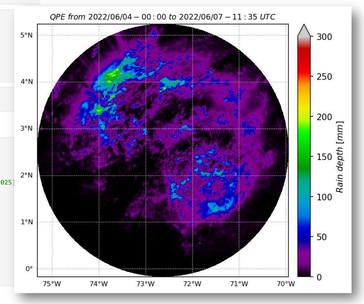
```
%%time
rr_depth = rain_rate(z=ds_05deg.DBZH, a=250, b=1.2).sum('vcp_time').load()
```

CPU times: total: 35.9 s
Wall time: 10.3 s

Plotting QPE

```
%%time
proj_crs = xd.georeference.get_crs(ds_05deg)
data_crs = ccrs.Projection(proj_crs)
fig, ax = plt.subplots(subplot_kw={'projection': ccrs.PlateCarree()}, dpi=150)
rr_depth.plot(x='x', y='y', cmap='nipy_spectral', cbar_kwargs={'label': r"$Rain \ depth \ [mm]$", 'pad':0.025},
              ax=ax,  transform=data_crs, vmin=0, vmax=300)
gl = ax.gridlines(draw_labels=True, ls='--', lw=0.5)
gl.xlabel_style = {'size': 8}
gl.ylabel_style = {'size': 8}
ax.set_title(f'$Guaviare \ Radar \ QPE$')
gl.top_labels,  gl.right_labels = False, False
start, end = pd.to_datetime(ds_05deg.vcp_time.values[0]), pd.to_datetime(ds_05deg.vcp_time.values[-1])
ax.set_title(f'$QPE \ from \ {start:%Y/%m/%d-%H:%M} \  to \  {end:%Y/%m/%d-%H:%M} \ UTC $', size=9)
fig.tight_layout()
plt.savefig('../results/Guaviare_QPE.jpeg',  bbox_inches='tight')
```

CPU times: total: 2.75 s
Wall time: 2.68 s



Radar data stored in ARCO format allow us to perform operations on the entire dataset with just a few lines of code, and the results will be ready in a few seconds, as shown in the red squares.

CONCLUSIONS

- The hierarchical radar data model, based on the WMO Cfradial2/FM301 standard, provides an effective solution for storing historical radar data. Adhering to FAIR principles and optimized for cloud storage.
- The time series at each node enables efficient analysis of historical datasets, climatology computation, and offline product generation without extensive computing resources and within reasonable times.
- The sequential translation from RAW to ARCO formats preserves the chronological order of radar scans, which is required for this data model despite its time-consuming.

REFERENCES

[1] Abernathey, R. P., Augspurger, T., Banihirwe, A., Blackmon-Luca, C. C., Crone, T. J., Gentemann, C. L., Hamman, J. J., Henderson, N., Lepore, C., McCaie, T. A., Robinson, N. H., & Signell, R. P. (2021). Cloud-Native Repositories for Big Scientific Data. *Computing in Science & Engineering*, *23*(2), 26–35. https://doi.org/10.1109/MCSE.2021.3059437

[2] Dixon, M. J., Curtis, M., Michelson, D., Hardin, J., Kehoe, K., & Haimov, S. (2019). CfRadial2 data file format: CF2 NetCDF format for RADAR and LIDAR data in radial coordinates - v2.0. doi:10.5065/fy2k-x587

[3] Ryzhkov, A., Zhang, P., Reeves, H., Kumjian, M., Tschallener, T., Trömel, S., & Simmer, C. (2016). Quasi-Vertical Profiles—A New Way to Look at Polarimetric Radar Data. *Journal of Atmospheric and Oceanic Technology*, *33*(3), 551–562. https://doi.org/10.1175/JTECH-D-15-0020.1

[4] Marshall, J. S., & Palmer, W. M. K. (1948). THE DISTRIBUTION OF RAINDROPS WITH SIZE. Journal of the Atmospheric Sciences, 5(4), 165–166. https://doi.org/10.1175/1520-0469(1948)005<0165:TDORWS>2.0.CO;2