RADAR DATATREE: A FAIR AND CLOUD-NATIVE FRAMEWORK FOR SCALABLE WEATHER RADAR ARCHIVES

A PREPRINT

Alfonso Ladino-Rincon
University of Illinois Urbana-Champaign
alfonso8@illinois.edu

Stephen W. Nesbitt
University of Illinois Urbana-Champaign
snesbitt@illinois.edu

October 30, 2025

ABSTRACT

We introduce **Radar DataTree**, the first dataset-level framework that extends the WMO FM-301 standard from individual radar volume scans to time-resolved, analysis-ready archives. Weather radar data are among the most scientifically valuable yet structurally underutilized Earth observation datasets. Despite widespread public availability, radar archives remain fragmented, vendor-specific, and poorly aligned with FAIR (Findable, Accessible, Interoperable, Reusable) principles—hindering large-scale research, reproducibility, and cloud-native computation. Radar DataTree addresses these limitations with a scalable, open-source architecture that transforms operational radar archives into FAIR-compliant, cloud-optimized datasets. Built on the FM-301/CfRadial 2.1 standard and implemented using `xarray.DataTree`, Radar DataTree organizes radar volume scans as hierarchical, metadata-rich structures and serializes them to Zarr for scalable analysis. Coupled with `Icechunk` for ACID-compliant storage and versioning, this architecture enables efficient, parallel computation across thousands of radar scans with minimal preprocessing. We demonstrate significant performance gains in case studies including Quasi-Vertical Profile (QVP) and precipitation accumulation workflows, and release all tools and datasets openly via the `Raw2Zarr` repository. This work contributes a reproducible and extensible foundation for radar data stewardship, high-performance geoscience, and AI-ready weather infrastructure.

Keywords FAIR data · Weather radar · Cloud-native · ARCO · Zarr · Data stewardship

1 Introduction

Ground-based weather radars are a cornerstone of atmospheric observation, capturing high-resolution data on precipitation, wind, and storm structures at fine temporal intervals. These systems support a wide range of societal and scientific applications, including severe weather forecasting, aviation safety, flood risk management, and climate-scale analysis. As radar networks expand globally and archives grow in volume and value, the potential for long-term, high-resolution atmospheric datasets continues to increase.

Yet despite their importance, radar data remains structurally challenging to use at scale. Existing archives are organized as millions of standalone binary files—often in proprietary or semi-standardized formats—designed for real-time operations, not scientific reuse. Metadata is inconsistently encoded, temporal indexing is absent, and no widely adopted framework currently supports organizing radar data as structured, versioned, and queryable datasets. Even open-access archives, such as NEXRAD Level II or Colombia's IDEAM radar network, present significant technical barriers to FAIR data use [Saltikoff et al., 2019].

In this paper, we introduce the **Radar DataTree** framework: a cloud-native, FAIR-aligned data model and storage architecture that transforms fragmented radar files into scalable, structured datasets. Radar DataTree extends existing file-level standards (such as WMO FM-301 / CfRadial 2.1) to support time-aware, hierarchical organization of radar scans using the `xarray.DataTree` data model. The resulting datasets preserve the full sweep-level structure of each radar scan, while aligning collections along a shared time dimension for analysis.

Radar DataTree is implemented and deployed using the open-source `Raw2Zarr` pipeline [Ladino, 2025]. This end-to-end Python framework ingests raw radar files, decodes and restructures them using `Xradar`, and serializes the result to Zarr format using `Icechunk`—a transactional storage engine that provides ACID-compliant versioning and concurrent-safe cloud updates. This architecture enables scalable, reproducible workflows for multi-scan radar analysis, real-time ingestion, and machine learning pipelines.

We demonstrate the Radar DataTree framework on operational radar archives, including NEXRAD and SIGMET datasets, and benchmark its performance on core radar analysis tasks such as Quasi-Vertical Profile (QVP) generation and Quantitative Precipitation Estimation (QPE). Results show up to two orders of magnitude speedup over traditional file-based workflows with full support for cloud-native execution and reproducibility.

This work contributes:

- A dataset-level data model for weather radar collections, compatible with FM-301 and Climate and Forecast (CF) conventions.
- An open-source implementation—`Raw2Zarr`—for ETL from operational radar archives into scalable, Icechunk-managed Zarr datasets.
- Empirical validation of the model's performance and usability on real-world archives and scientific workflows.

By bridging radar-specific standards with modern data infrastructure, Radar DataTree provides a foundation for interoperable, scalable, and AI-ready weather radar science.

2 Background and Motivation

Weather radar systems produce some of the most spatiotemporally detailed observations in atmospheric science. Each radar volume scan captures reflectivity, Doppler velocity, and polarimetric variables across a three-dimensional cone of the atmosphere by rotating 360° at multiple fixed elevation angles. These scans are grouped into Volume Coverage Patterns (VCPs), which define the radar's sweep strategy—number of angles, rotation speed, and dwell time—based on the operational mode (e.g., clear-air or storm surveillance). Scans are typically repeated every 4–10 minutes, resulting in dense four-dimensional datasets (x, y, z, t) that are critical for both real-time operations and retrospective analysis.

Despite their observational richness, radar archives are difficult to use due to how data is structured and stored. Most operational radar networks—including NEXRAD (U.S.), IDEAM (Colombia), and FMI (Finland)—store each VCP as an individual binary file in vendor-specific formats. These formats were designed for near-real-time visualization and have limited support for long-term reuse, metadata discovery, or spatial-temporal subsetting.

FM-301–compliant archives, while structurally improved, still require scan-by-scan parsing in the absence of a dataset-level abstraction. This file-centric model is fundamentally misaligned with FAIR data principles [Saltikoff et al., 2019]. While many archives are technically findable and accessible, they are rarely interoperable or reusable without expert knowledge and repeated effort.

Efforts like FM-301, Zarr, and `Xradar` have made strides in improving radar data structure. Radar DataTree builds upon this ecosystem to address what remains unsolved: a cohesive, time-aware, analysis-ready model for full radar collections.

3 Related Work

Radar data interoperability and standardization have long been active areas of development across both meteorological agencies and the open-source software community. Prior efforts have primarily focused on improving the usability of individual radar scans, with less emphasis on collection-level structure, temporal alignment, or cloud-native scalability.

File-level standards. The most widely adopted specification for radar data is the WMO FM-301 standard, published as part of the Manual on Codes (WMO-No. 306) [World Meteorological Organization (WMO), 2025]. FM-301 defines a hierarchical schema for radar data based on the CfRadial 2.1 convention and CF metadata conventions. It allows individual radar sweeps to be encoded as NetCDF4 or HDF5 files with standardized variable names, coordinate systems, and metadata groups. However, FM-301 applies only at the level of single radar volume scans and does not specify how multiple files should be organized, indexed, or queried as a cohesive dataset over time.

Open-source radar libraries. Several community-maintained Python packages provide tools for ingesting, decoding, and analyzing radar data. `Py-ART` offers a general-purpose radar processing toolkit that supports multiple formats

and provides functionality for gridding, visualization, and filtering [Helmus and Collis, 2016]. `Xradar` extends this ecosystem by providing strict support for FM-301/CfRadial 2.1 decoding and seamless integration with `xarray` objects [Grover et al., 2025]. These libraries focus primarily on volume-level workflows and do not address scalable data storage, temporal indexing, or dataset-level modeling.

Cloud-native formats. The geoscience community has increasingly adopted formats like Zarr for cloud-optimized, chunked storage of multidimensional arrays. Zarr's compatibility with object storage and parallel I/O has made it a preferred backend in frameworks such as Pangeo. However, raw Zarr stores lack native support for versioning, transactions, or structured group hierarchies unless coupled with additional tooling. Icechunk [Abernathey et al., 2024] addresses this gap by providing ACID-compliant transactional storage for Zarr datasets, enabling concurrent writes, atomic updates, and version-controlled metadata in distributed environments.

Community radar infrastructure. The broader radar science community has increasingly embraced open-source software as a foundation for reproducible research and long-term data stewardship. Heistermann et al. [Heistermann et al., 2015] surveyed the emergence of open-source radar tools, highlighting the role of community-led initiatives in lowering technical barriers and fostering interoperability.

Gap in dataset-level radar modeling. To date, no framework combines radar-specific standards (e.g., FM-301) with scalable, transactional, time-aware modeling of entire radar archives. Existing tools operate primarily at the level of single files or in-memory arrays. Radar DataTree addresses this gap by providing a formal data model that represents radar collections as hierarchical, metadata-rich datasets aligned along time. By building on FM-301, implementing with `xarray.DataTree`, and persisting with Icechunk-managed Zarr stores, Radar DataTree enables scalable, reproducible radar workflows aligned with FAIR and cloud-native design principles.

To date, no framework combines radar-specific standards (e.g., FM-301) with scalable, transactional, time-aware modeling of entire radar archives. Existing tools operate primarily at the level of single files or in-memory arrays. Radar DataTree builds upon these efforts by providing a formal data model that represents radar collections as hierarchical, metadata-rich datasets aligned along time.

4 Data Model and Architecture

Radar data is inherently hierarchical and time-resolved. Each radar volume scan (VCP) consists of multiple sweeps at different elevation angles, with each sweep containing arrays of polarimetric variables (e.g., reflectivity, velocity, differential phase) on a polar grid. These scans repeat every few minutes, producing a nested, multivariate time series that spans days to decades. However, traditional radar archives store each scan as an isolated binary file, with no higher-level abstraction for organizing, aligning, or querying data over time.

Radar DataTree structure. The **Radar DataTree** model addresses this limitation by representing a radar archive as a hierarchical, metadata-rich tree of datasets aligned along a common time axis. At its core, each radar volume scan is represented as a subtree containing its original FM-301 / CfRadial 2.1 group structure—preserving sweeps, coordinate metadata, and variable attributes. These subtrees are combined into a parent tree organized by observation time, forming a unified dataset that maintains both sweep-level detail and time-series navigability.

This structure is implemented using `xarray.DataTree`, a new hierarchical container in the xarray ecosystem that enables nested datasets with shared coordinate alignment and efficient traversal. Radar DataTree leverages this abstraction to organize heterogeneous scans into a coherent time-indexed archive while preserving full metadata integrity.

Metadata alignment and FM-301 compliance. Each volume scan in the Radar DataTree preserves its original FM-301-compliant structure, including:

- Nested groups for each sweep, encoded with consistent dimension names (e.g., time, range, azimuth)
- CF-compliant coordinate variables for spatial referencing
- Global and per-variable metadata for instrument settings, calibration, and units

To enable analysis across time, the Radar DataTree model introduces a top-level 'time' dimension indexing all volume scans. This allows downstream tools to treat the archive as a temporally continuous dataset, with sweep-level metadata preserved per scan.

Table 1: Core components of the Radar DataTree ecosystem

Component	Role in Architecture
FM-301	File-level standard for radar volumes and sweeps
`xarray.DataTree`	Hierarchical in-memory representation of scans
Zarr	Chunked, compressed, cloud-native storage format
Icechunk	ACID-compliant transactional engine for Zarr datasets

Zarr serialization and Icechunk persistence. To support cloud-native access and scalable analytics, the Radar DataTree is serialized to the Zarr format. Zarr stores multidimensional arrays as compressed, chunked files in object storage, enabling efficient partial reads, lazy evaluation, and parallel processing. Each scan is stored as a group within a time-indexed hierarchy, preserving the internal sweep structure and associated metadata.

Radar DataTree archives are persisted using `Icechunk` [Abernathey et al., 2024], a transactional storage engine that wraps Zarr stores with ACID guarantees. Icechunk maintains metadata catalogs, write-ahead logs, and atomic commit protocols to support safe concurrent access and version-controlled updates in distributed environments. This ensures that ingest pipelines, real-time workflows, and downstream users can operate on the same archive without risking data corruption or race conditions.

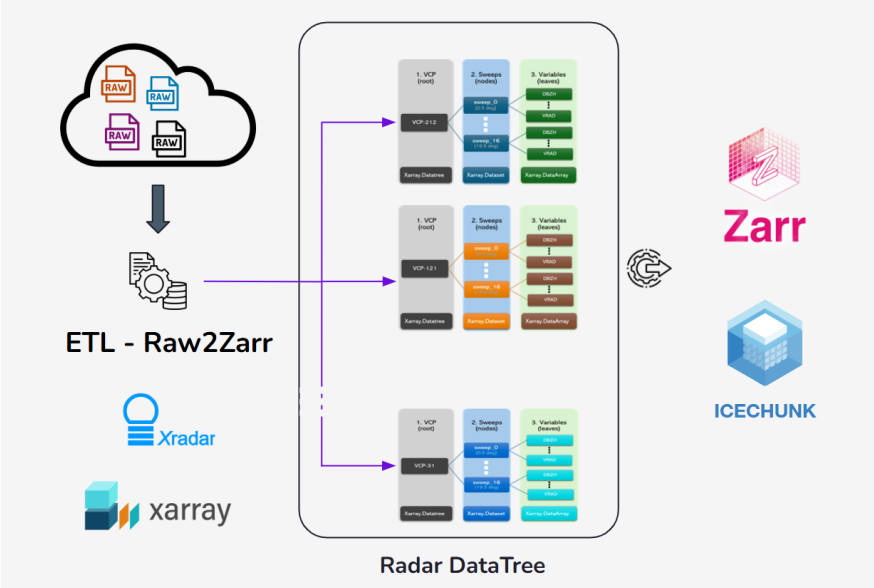

Figure 1: End-to-end ETL pipeline for radar archive ingestion. Raw radar files hosted in cloud storage (e.g., NEXRAD Level II, SIGMET) are decoded using `Xradar`, structured into a time-aligned hierarchical dataset using `xarray.DataTree`, and serialized into transactional `Zarr` stores managed by `Icechunk`. The entire process is implemented in the open-source `Raw2Zarr` package.

Raw2Zarr pipeline. The full architecture is operationalized in the open-source `Raw2Zarr` Python package [Ladino, 2025], which performs end-to-end extraction, transformation, and loading (ETL) of radar archives, as illustrated in Figure 1.

The pipeline consists of four stages:

1. **Extraction:** Raw Level II or SIGMET files are downloaded from cloud buckets or S3-compatible object stores.
2. **Transformation:** Files are decoded using `Xradar` [Grover et al., 2025], parsed into FM-301-compliant `xarray.Dataset` objects, and grouped by time.
3. **Tree construction:** Each dataset is inserted as a node into a `DataTree`, preserving sweep metadata and scan structure.

4. **Loading:** The complete tree is serialized to a Zarr store and committed to Icechunk for ACID-compliant persistence and version tracking.

This design enables scalable, reproducible, and cloud-ready radar archives that can support a wide range of scientific and operational applications without repeated preprocessing or bespoke file handling.

4.1 Interactive Access and Exploration in Practice

To demonstrate the practical use of the Radar DataTree model, Figure 2 shows an interactive session accessing the full May 2011 archive of the KVNX radar site using the `xarray.DataTree` interface and the `arraylake` client. This archive spans over 765 GB of Zarr-encoded data persisted in an `Icechunk` store.

Once loaded, the radar collection is exposed as a time-indexed tree of datasets, with each node representing a distinct Volume Coverage Pattern (VCP). As shown in the interface, users can access any individual sweep using intuitive, path-like syntax (e.g., `"VCP-212/sweep_0"`). Variables such as radar reflectivity (DBZH) are immediately accessible as self-described, chunked arrays with rich metadata, standard-compliant attributes, and an optimized layout for Dask-based processing.

This capability transforms radar data from a disjoint collection of binary files into a fully navigable scientific dataset. It enables:

- Seamless subsetting and on-demand querying across large temporal ranges,
- Efficient use of cloud-native analysis tools and parallel processing,
- Reproducible exploration from within standard scientific Python environments such as Jupyter.

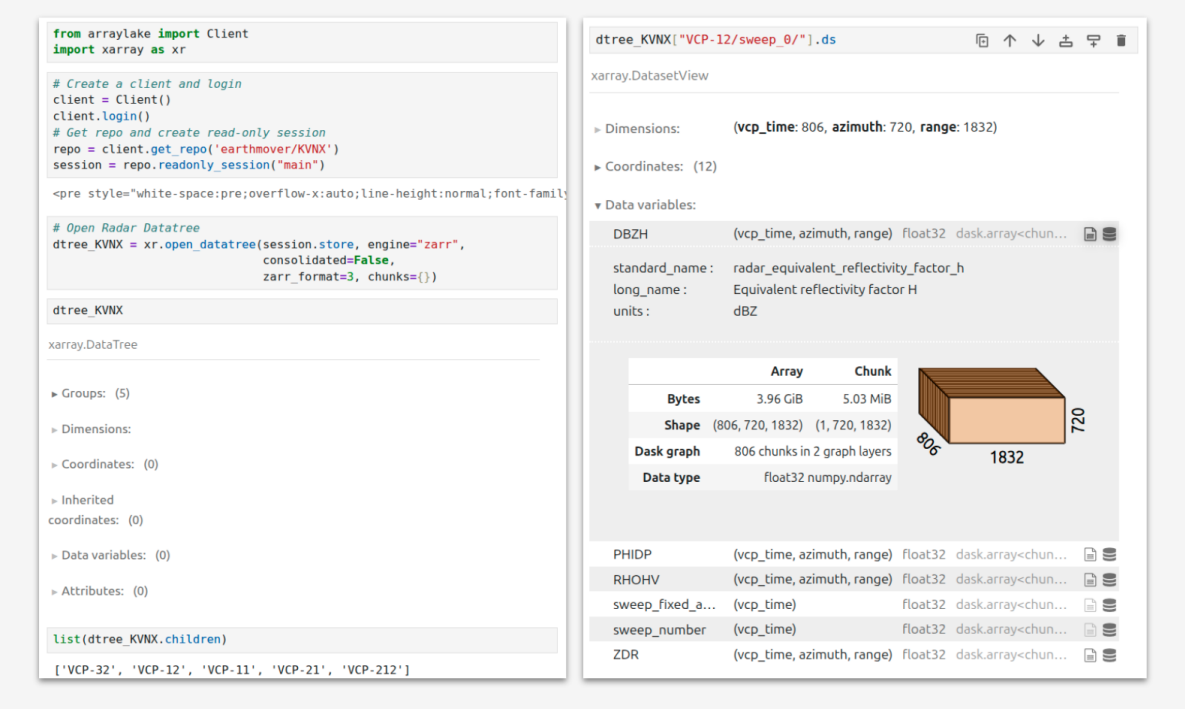

Figure 2: Interactive exploration of the KVNX Radar DataTree using `xarray.DataTree` and the `arraylake` Python client. The full May 2011 archive (~765 GB) is loaded as a single navigable object. Each node represents a Volume Coverage Pattern (VCP) and can be accessed using a simple path syntax. Radar reflectivity variables (e.g., DBZH) are self-described, chunked, and cloud-optimized for scalable analysis.

5 Case Studies and Performance

To evaluate the usability and efficiency of the Radar DataTree framework, we apply it to several common radar analysis workflows. These case studies highlight the benefits of dataset-level organization, parallel access, and cloud-native storage over traditional file-based methods.

5.1 Quasi-Vertical Profile (QVP) Generation

Quasi-Vertical Profiles (QVPs) summarize vertical trends in radar variables by compositing azimuthal means from constant-elevation sweeps over time [Ryzhkov et al., 2016]. This technique provides insights into storm microphysics, melting layer detection, and hydrometeor classification.

In a traditional archive, each radar volume must be decoded independently, relevant sweep angles must be located and matched across files, and metadata inconsistencies handled manually. With Radar DataTree, QVP workflows are simplified: sweeps are uniformly stored, aligned in time, and queryable via a single interface.

On a one-week NEXRAD archive, QVP generation using Radar DataTree yielded over a 100× speedup compared to baseline Py-ART-based scripts operating on raw Level II files. Parallelized extraction using Dask enabled scalable compute across cloud-backed Zarr stores with minimal I/O overhead.

This demonstrates not only computational gains but also a substantial reduction in workflow complexity and code maintenance.

5.2 Time-Series Extraction at Fixed Location

Extracting radar time series at fixed spatial locations is essential for sensor intercomparisons, validation, and data assimilation workflows. Traditionally, this involves directory scanning, decoding binary files, coordinate transformation, and interpolation.

Radar DataTree abstracts these steps by treating each scan as a time-aligned, metadata-consistent node. Queries can be executed directly across the archive to extract pointwise time series without loading entire volumes into memory.

On a month-long NEXRAD archive, Radar DataTree reduced extraction latency and memory footprint by over an order of magnitude, allowing sub-minute retrieval of multi-week time series using cloud-native compute.

5.3 Quantitative Precipitation Estimation (QPE)

QPE uses reflectivity-derived rain rates to compute precipitation accumulations over time. One standard method applies the Marshall–Palmer Z–R relationship, which links reflectivity Z to rain rate R via an exponential distribution of drop sizes [Marshall and Palmer, 1948].

Conventional QPE pipelines require scan-by-scan decoding, calibration, resampling, and accumulation—all steps which must be repeated for each archive. Radar DataTree enables direct access to reflectivity fields aligned in time and space, significantly reducing overhead.

In a three-week, multi-radar QPE case study, we observed 70–150× performance gains in total compute time, with reproducible accumulation fields generated using Icechunk-managed updates and versioned datasets.

Figure 3 shows interactive computation of Quasi-Vertical Profiles (QVPs) and Quantitative Precipitation Estimation (QPE) directly from the Radar DataTree, computed with cloud-native tools. Both examples are available in the Radar DataTree Demo repository[1], which demonstrates reproducible workflows using Earthmover infrastructure.

All parallel processing benchmarks were run on a 10-worker Dask cluster[2].

5.4 Transactional Updates and Reproducibility

Radar DataTree's integration with Icechunk allows radar archives to support safe concurrent access and real-time ingestion. New volume scans can be appended as ACID-compliant transactions, enabling downstream analytics to operate on live data without reprocessing the entire archive.

[1] `https://github.com/earth-mover/radar-data-demo`

[2] Cluster deployed using `coiled` with 10 `m6i.large` workers (2 vCPUs, 8 GiB RAM each) and a `m6i.xlarge` scheduler in `us-east-1`. Software environment: `nexrad-env`. Full dashboard: `https://cluster-gxyov.dask.host`.

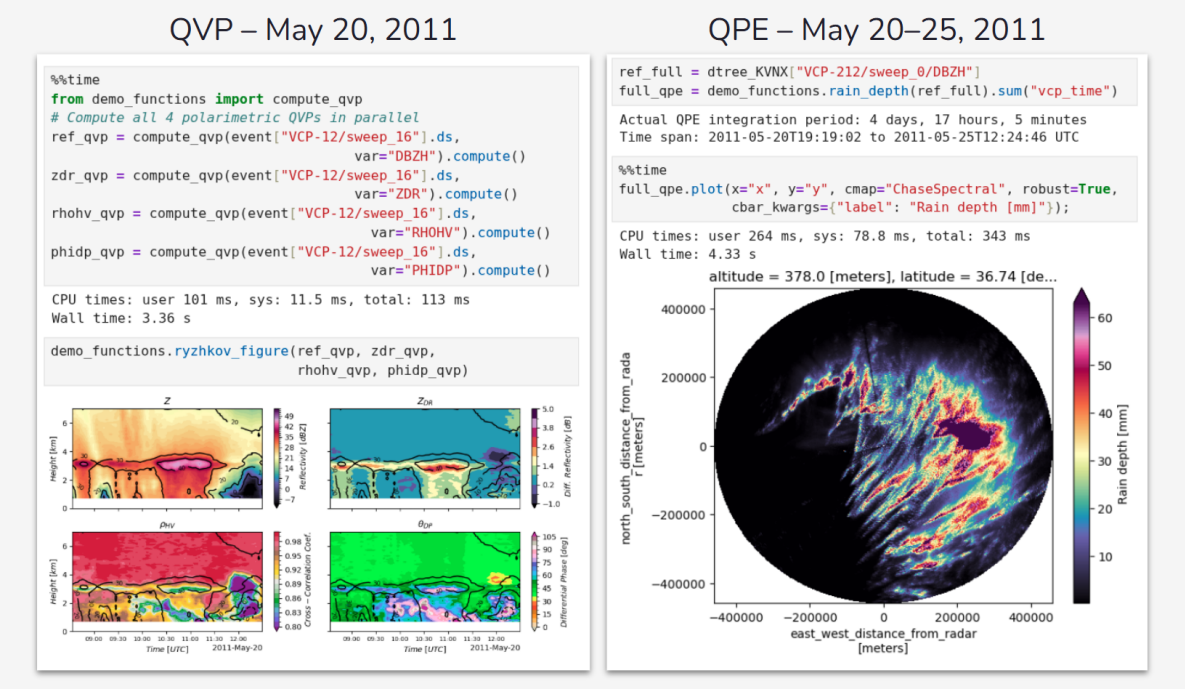

Figure 3: Interactive computation of two standard radar science products from the KVNX Radar DataTree using `xarray` and Dask. (**Left**) Quasi-Vertical Profiles (QVPs) of four polarimetric variables for VCP-12 on May 20, 2011, following the method of Ryzhkov et al. [2016]. Total compute time: 3.36 s. (**Right**) Quantitative Precipitation Estimation (QPE) by time-integrating radar reflectivity (DBZH) from VCP-212 sweeps over 4.7 days using a reflectivity–rainrate relation [Marshall and Palmer, 1948]. Compute time: 4.33 s on a 10-worker Dask cluster.

In a validation experiment, a month-long dataset was constructed incrementally using daily data streams. Versioned Icechunk commits allowed for rollback and re-execution of QVP and QPE analyses with bitwise-identical results, confirming both reproducibility and provenance tracking.

6 Conclusion and Impact

Despite its scientific value, weather radar data has remained structurally underserved in modern data systems. While archives such as NEXRAD Level II, IDEAM Colombia, and FMI Finland are now publicly hosted in the cloud, they remain locked behind file-based workflows, legacy formats, and metadata inconsistencies.

Radar DataTree addresses this gap by introducing a dataset-level abstraction aligned with FAIR data principles and optimized for cloud-native environments. By extending FM-301 from individual scans to time-indexed collections, and combining `xarray.DataTree`, Zarr, and Icechunk, we offer a complete framework for structured, scalable radar data workflows.

Intended audience. This framework is designed for radar data scientists, meteorologists, atmospheric researchers, open science advocates, and data infrastructure engineers working at the intersection of environmental data and scalable computing.

References

Elena Saltikoff, Katja Friedrich, Ari Huuskonen, Katharina Lengfeld, Michael Werner, Mahesh Chandra, et al. An overview of using weather radar for climatological studies: Successes, challenges, and opportunities. *Bulletin of the American Meteorological Society*, 100(12):2253–2270, 2019. doi:10.1175/BAMS-D-18-0166.1.

Alfonso Ladino. aladinor/raw2zarr: Release v0.5.0, 2025. URL `https://doi.org/10.5281/zenodo.17396262`.

World Meteorological Organization (WMO). *Manual on Codes, Volume I.2 – International Codes: Annex II to the WMO Technical Regulations, Part B – Binary Codes, Part C – Common Features to Binary and Alphanumeric Codes*.

WMO, Geneva, 2025 edition edition, 2025. ISBN 978-92-63-10306-2. URL https://library.wmo.int/idurl/4/35625. WMO-No. 306, Basic Documents No. 2.

Jonathan J. Helmus and Scott M. Collis. The python arm radar toolkit (py-art), a library for working with weather radar data in the python programming language. *Journal of Open Research Software*, 4(1):e25, 2016. doi:10.5334/jors.119.

Maxwell Grover, Kai Mühlbauer, Edouard Goudenhoofdt, H.A. Syed, Alfonso Ladino, Ryan Jackson, and David Wolfensberger. openradar/xradar: xradar v0.10.0, 2025. URL https://github.com/openradar/xradar.

Ryan Abernathey et al. icechunk: Transactional storage for cloud-optimized data, 2024. URL https://github.com/earth-mover/icechunk.

Maik Heistermann, Scott Collis, Michael J. Dixon, Scott Giangrande, Jonathan J. Helmus, Brian Kelley, Jarmo Koistinen, Daniel B. Michelson, Markus Peura, Thilo Pfaff, and David B. Wolff. The emergence of open-source software for the weather radar community. *Bulletin of the American Meteorological Society*, 96(1):117–128, 2015. doi:10.1175/BAMS-D-13-00240.1.

Alexander Ryzhkov, Pengfei Zhang, Heather Reeves, Matthew Kumjian, Thierry Tschallener, Silke Trömel, and Clemens Simmer. Quasi-vertical profiles—a new way to look at polarimetric radar data. *Journal of Atmospheric and Oceanic Technology*, 33(3):551–562, 2016. doi:10.1175/JTECH-D-15-0020.1.

J. S. Marshall and W. McK. Palmer. The distribution of raindrops with size. *Journal of Meteorology*, 5(4):165–166, 1948.