Simplifying analysis of hierarchical HDF5 and NetCDF4 files with xarray-DataTree

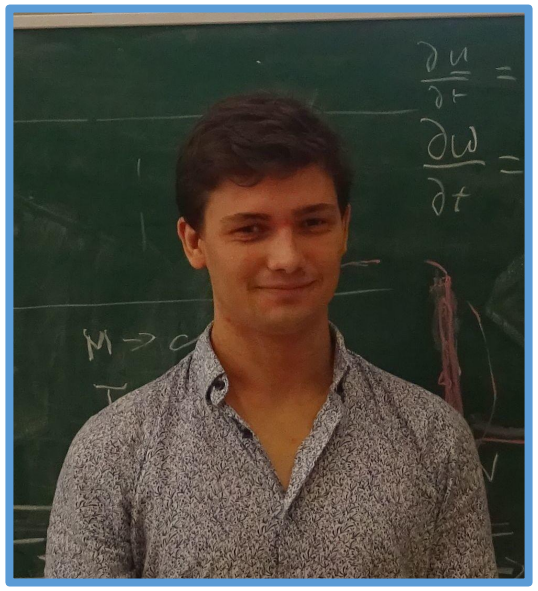
or *How Trees Can Help You* 

Eni Awowale [1], Tom Nicholas PhD[2],
Lucas Sterzinger PhD [1], and Nick Lenssen [1]

[1] NASA Goddard Earth Sciences (GES) Data and Information Services Center (DISC)
[2] [C]Worthy

Who we are? A ragtag team of scientists and engineers

Eni Awowale
Earth scientist and software engineer at NASA GES DISC
new xarray core developer!



Tom Nicholas, PhD
Staff Scientist at [C]Worthy LLC.
Xarray core developer
Original author of xarray-DataTree

Lucas Sterzinger, PhD
Atmospheric scientist and software engineer at NASA GES DISC

Also thanks to Nicholas Lenssen, Owen Littlejohns, Matt Savoie, and Stephan Hoyer

Who are we? GES DISC

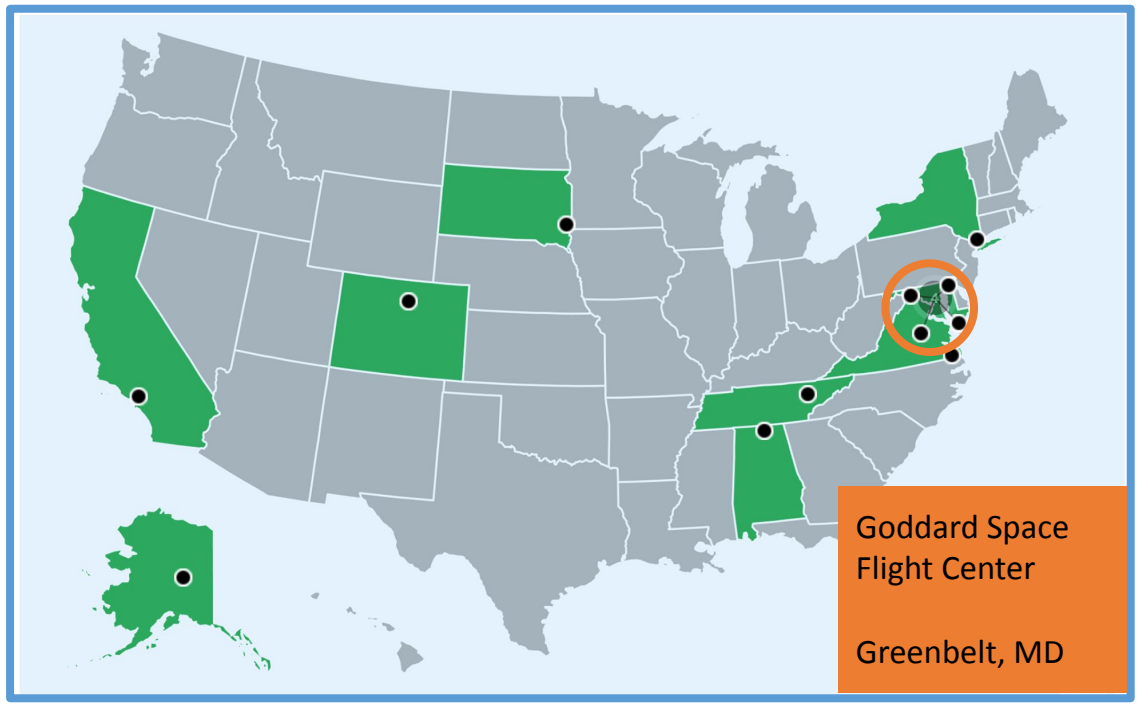


Image source: https://www.earthdata.nasa.gov/eosdis/daacs

- GES DISC:
  - Goddard Earth Sciences Data and Information Services Center
- One of 12 NASA Distributed Active Archive Centers (DAACs)
  - DAACs are organized by subject matter and tasked with archiving and distributing NASA's Earth science data
- GES DISC's primary datasets deal with:
  - Atmospheric composition
  - Atmospheric dynamics
  - Global precipitation
  - Solar irradiance
- Popular Datasets:
  - GPM (Global Precipitation Measurement)
  - MERRA2 Reanalysis
  - NLDAS/GLDAS/FLDAS (Land Surface Assimilation Models)
  - Aqua, SNPP, JPSS-1/2 Atmospheric Sounders

We've got lots of data!

- GES DISC is moving its online data archive to the cloud
  - >6 petabytes of data and only getting bigger
- Data are supported by cloud services under active development
- Need a robust way to test that these services are working for our cloud hosted data.

- Much of NASA Earth Science data is in a storage format called "HDF" or "Hierarchical Data Format"

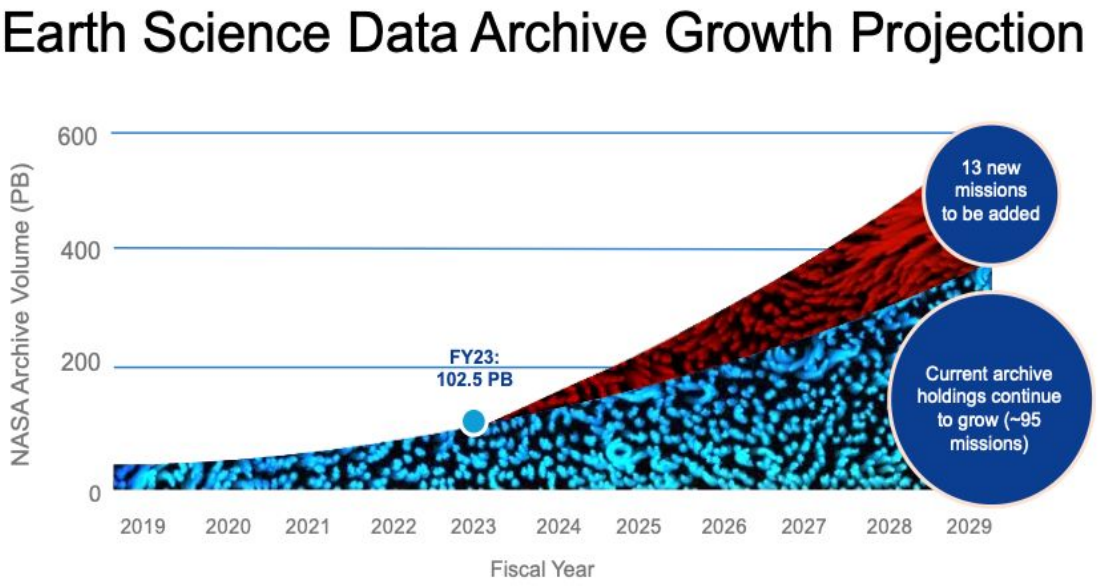


Image source: https://www.earthdata.nasa.gov/technology/open-science

What are hierarchical data formats?

HDF (or the Hierarchical Data Format) is a model for storing, managing, and describing data.

- HDF4 developed in the 1980s, HDF5 in 1990s
  - National Center for Supercomputing Applications, Univ of Illinois
- Current specification is HDF5, and is managed by the non-profit HDF Group.
- HDF5 is the storage specification used by the popular NetCDF4 file format. The vast majority of Earth science data is stored with HDF in one form or another.
  - In large part due to NASA, who selected HDF out of 15 data formats for use in the Earth Observing System (EOS) mission satellites

HDF5 - Pros and Cons

Benefits 👍

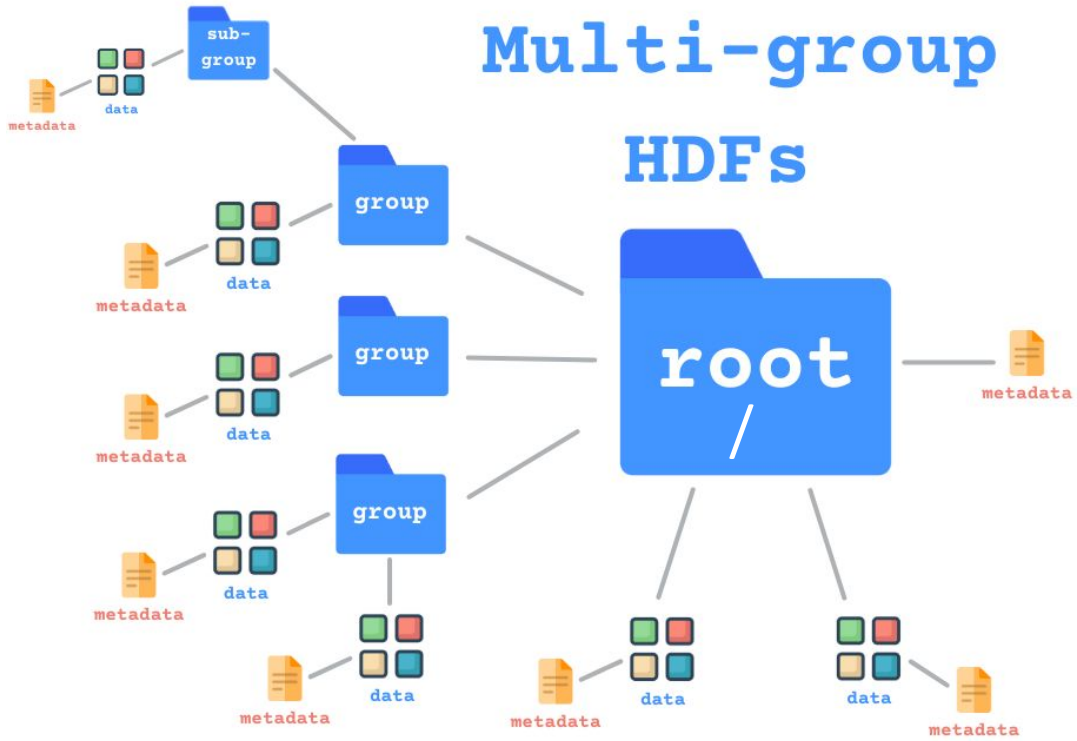
- Much simplified over HDF4
- Multi-dimensional
- Self Describing
  - Older storage models relied on external tables to describe file contents
- Support for heterogeneous data
- Open format
- Supports data slicing
  - Can extract a range of data without loading it all into memory
- Broad support from programming languages and toolkits
- Efficient, compressible binary storage

Downsides 👎

- Designed for on-disk access, difficult to optimize for cloud storage
- Complex, open-ended format
  - NASA-derived spinoff HDF-EOS and HDF-EOS5 added complexity
  - NetCDF4 is a popular, more simplified file format that is based on a restricted HDF5 storage layer

Groups within HDF5

- There are two main concepts in HDF: Datasets and Groups
  - Datasets
    - Single multi-dimensional array of data, with its own attributes and metadata
  - Groups
    - Collection of datasets, or other groups. Datasets and groups may belong to one or more groups.
    - Acts similarly to directories in a filesystem
    - Groups can have separate dimension variables
- Group and metadata structures can vary wildly between datasets (Quirky™ Data), makes building dataset-agnostic subsetting services difficult



The problem with groups

- One of our teams primary objectives at GES DISC is to provide subsetting services for all of our different datasets
  - This includes spatial, temporal and variable subsetting services and allows scientists to collect data on the exact region, time and, or phenomenon they are interested in
- Difficult because different datasets treat grouping differently
- Popular tools like xarray and netCDF4 python libraries can only open a single group at a time
- Writing code that supports different datasets can be difficult because each dataset may have it's own unique group hierarchy
  - For example, spatially subsetting a generic grouped HDF or NetCDF4 file is difficult as it involves traversing an unknown group hierarchy

How we subset grouped datasets

- Make a copy of the file and open it with `nc4.Dataset()`
- Loop through every group and subgroup for variables and dimensions

Flatten the dataset:

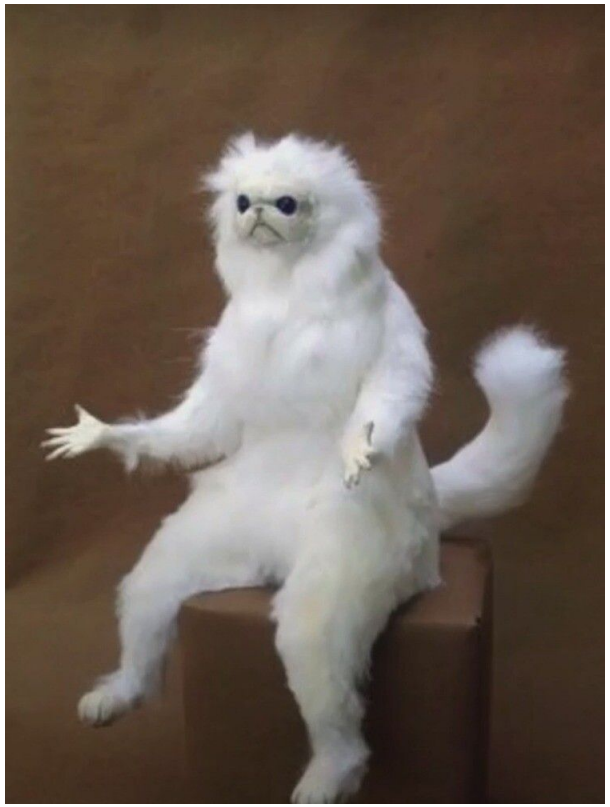
  - Copy the variables and dimension into a new variable in the root group
  - Change the variable name to include its group's path
  - Delete each variable from its respective group

- If there are subgroups, use the flattening function recursively 😵‍💫
- Do the actual variable or coordinate subset on the newly flattened dataset ✅

BUT THEN - to preserve the group hierarchy of the original file:

- Create a new netCDF4 dataset with the groups of the original dataset
  - Get this from the full path names of the variables in the subsetted and flattened dataset
- Copy variables into their groups and change variables back to their original names

Yes, we know this is kind of confusing



Why this method is imperfect

- You have to unpack and package a dataset
  - It's like opening a box, unpacking it, reorganizing, removing everything you don't want, and then getting another box to put everything in 😵
- Writing and supporting recursive code can be challenging
  - One bug can result in an infinite loop
- Not great for memory
  - Copy of original dataset is made for each subset request
- Makes code difficult to follow and visualize
- Slows down processing speed

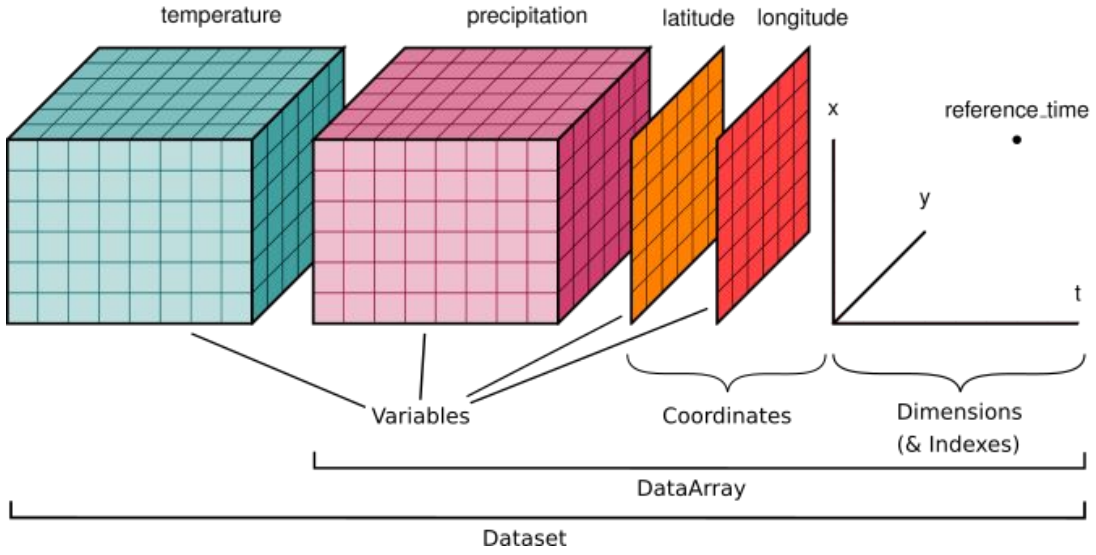
What do we want?

A simple(ish) way to open grouped HDF files

- Minimal code complexity
  - No RECURSIVE code!
- Fast(ish)
- Reduces the amount of duplication
  - No need for copying datasets
- Opens grouped datasets without having to specify each individual group
  - Understands the group hierarchy without any additional inputs from the user
- Works as simple as `open_datatree()` …

What is xarray?

- Python package providing N-D labelled arrays, datasets, and metadata
  - Flexible data model and toolkit for scientific data
- NumPy with labels
  - selection through labelled dimensions rather than numpy integer axes
- In-memory representation of a netCDF group




```
# xarray style
>>> ds.sel(time='2017-11-28').max(dim='station')

# numpy style
>>> array[[0, 1, 2, 3], :, :].max(axis=2)
```

Have you done this?

- Who here uses many separate `xarray.Dataset` objects to open all the different groups of ONE dataset?
- You may start by doing doing a `ncdump` or `nc4.Dataset().groups` to get all of the groups

```
list_of_groups = list(nc4.Dataset(datset_name).groups)
print(list_of_groups)
```

✓ 0.0s

```
['Cloud', 'Geolocation', 'Metadata', 'Meteo', 'Offset', 'Science', 'Sequences']
```

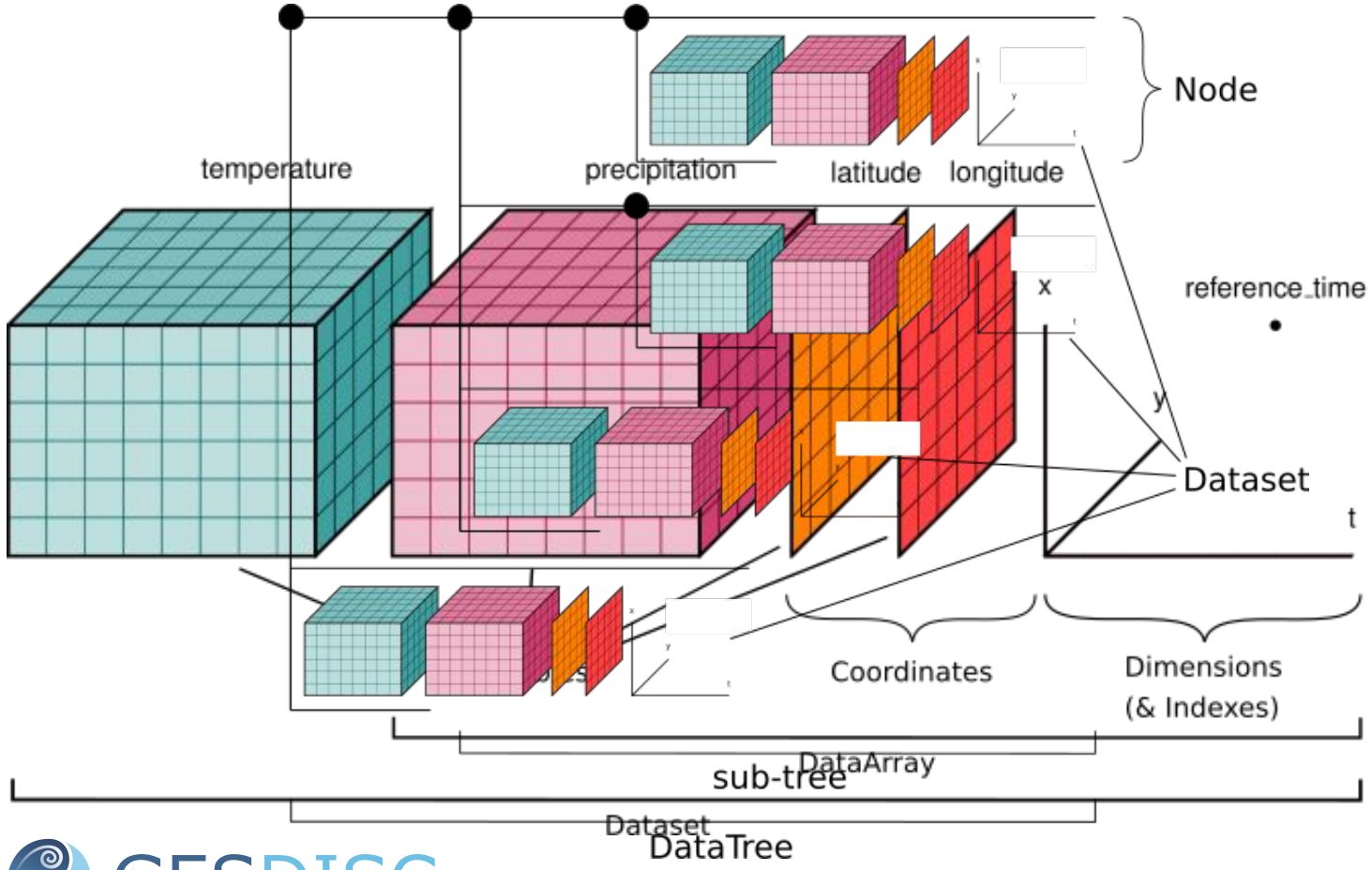
- Then you open each group with `xarray`

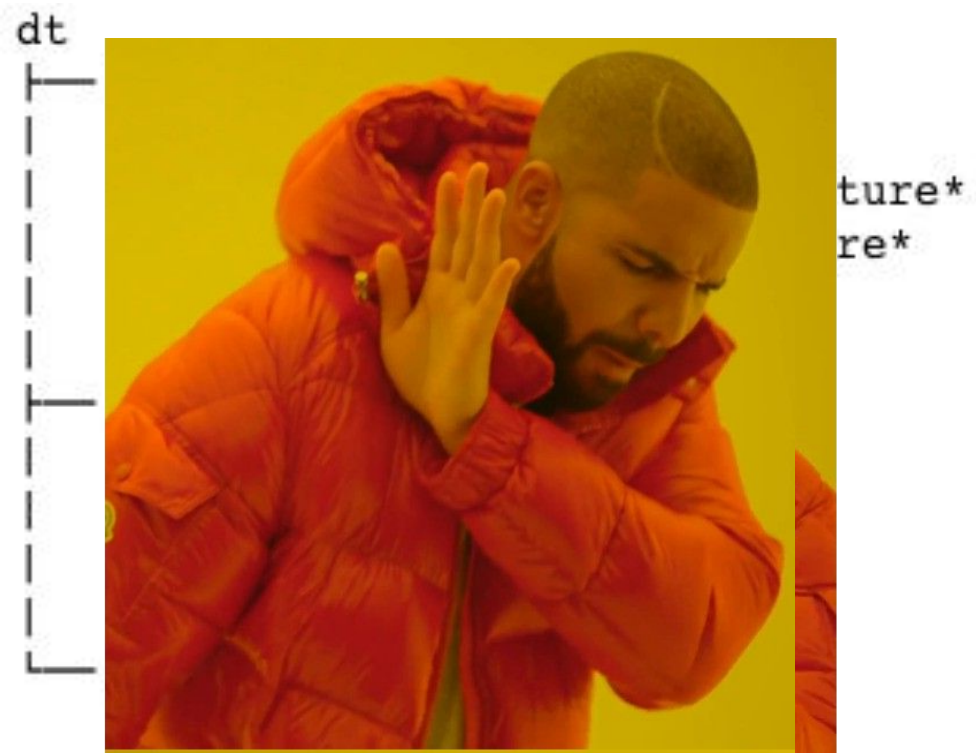
```
for group in list_of_groups:
    xr.open_dataset(datset_name, group=group)
```

- Problem: You have to open each group like it's a separate dataset!

Enter xarray-DataTree

- A “DataTree” is a hierarchical tree of xarray Datasets





Features 1: I/O

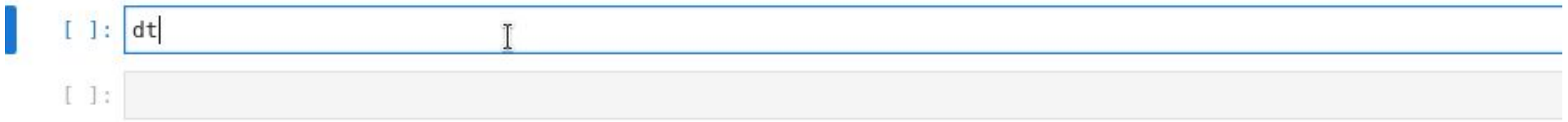
- Open a netCDF file (/ Zarr store) containing multiple groups as a nested tree
- (Can save back out as file with multiple groups too)

```
[4]  oco2_tree = open_datatree('./downloads/OCO2_L2_Lite_SIF.11r/oco2_LtSIF_220101_B11012Ar_220627180315s.nc4')
     ✓ 0.2s

[26] print(oco2_tree)
     ✓ 0.0s

...  DataTree('None', parent=None)
     │   Dimensions:                (sounding_dim: 188677, vertex_dim: 4)
     │   Dimensions without coordinates: sounding_dim, vertex_dim
     │   Data variables: (12/15)
     │       Delta_Time             (sounding_dim) float64 2MB ...
     │       SZA                    (sounding_dim) float32 755kB ...
     │       VZA                    (sounding_dim) float32 755kB ...
     │       SAz                    (sounding_dim) float32 755kB ...
     │       VAz                    (sounding_dim) float32 755kB ...
     │       Longitude              (sounding_dim) float32 755kB ...
     │       ...                     ...
     │       SIF_740nm              (sounding_dim) float32 755kB ...
     │       SIF_Uncertainty_740nm  (sounding_dim) float32 755kB ...
     │       Daily_SIF_740nm        (sounding_dim) float32 755kB ...
     │       Daily_SIF_757nm        (sounding_dim) float32 755kB ...
     │       Daily_SIF_771nm        (sounding_dim) float32 755kB ...
     │       Quality_Flag           (sounding_dim) float64 2MB ...
     │   Attributes: (12/32)
     │       References:                       ['Sun, Y. et al., Remote Sensing of En...
     │       conventions:                      CF-1.6
     │       product_version:                  B11012Ar
     │       summary:                          Fraunhofer-line based SIF retrievals
     │       keywords:                         ISS, OCO-2, Solar Induced Fluorescence...
     │       keywords_vocabulary:              NASA Global Change Master Directory (G...
     │       ...                               ...
     │       InputBuildId:                     B11.0.06
     │       InputPointers:                    oco2_L2MetGL_39883a_211231_B11006r_220...
     │       CoordSysBuilder:                  ucar.nc2.dataset.conv.CF1Convention
     │       identifier_product_doi_authority: http://dx.doi.org/
     │       gesdisc_collection:               11r
```

Features 2: Interactive HTML representation



Features 3: Node Relationships

- Groups are connected as parent/children (& siblings/ancestors etc...)

```
homer.children = {"Bart": bart, "Lisa": lisa, "Maggie": maggie}
```

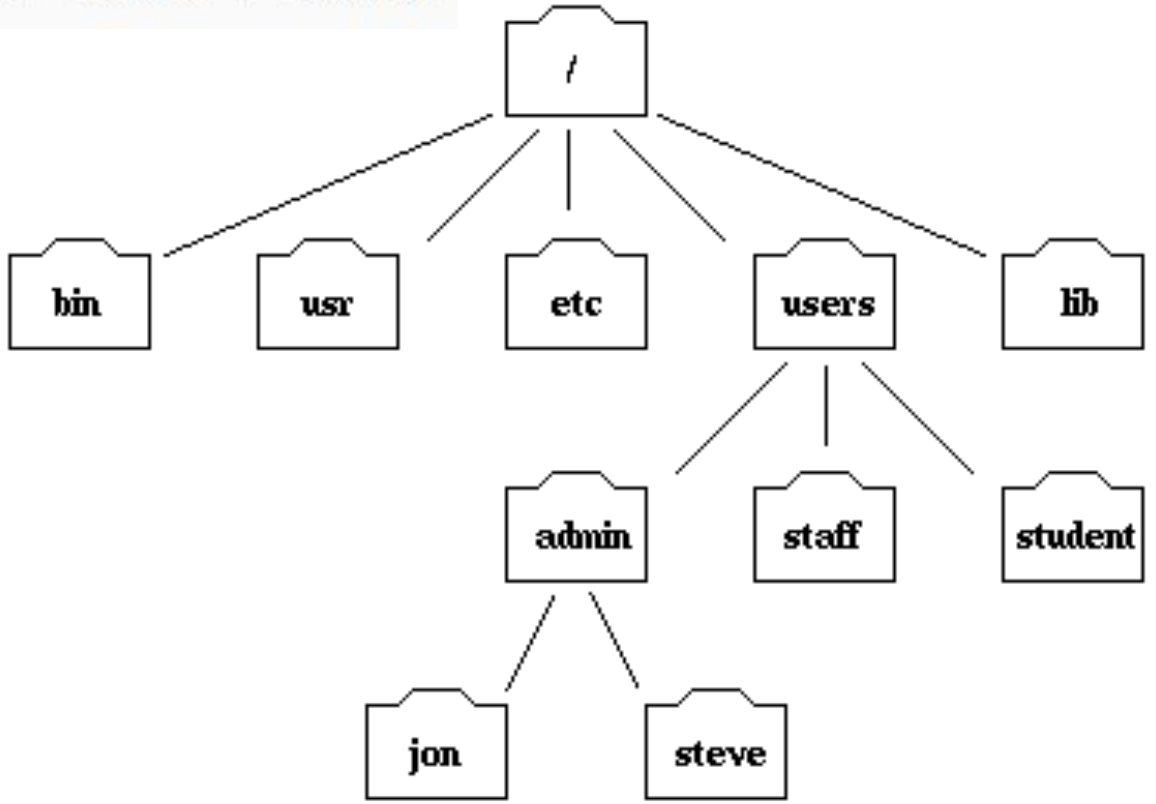
```
DataTree('Abe', parent=None)
├── DataTree('Homer')
│   ├── DataTree('Bart')
│   ├── DataTree('Lisa')
│   └── DataTree('Maggie')
└── DataTree('Herbert')
```

```
In [9]: maggie.parent.name
Out[9]: 'Homer'
```

- Access via file path-like syntax

```
In [39]: bart.relative_to(lisa)
Out[39]: '../Bart'
```

  - Or via attributes
    - i.e. dt.model.experiment_a



Part of the filesystem tree

Features 4: Map computations over tree

- Xarray's computation methods are automatically mapped over entire tree below

```
dt.mean(dim="time")
```

- Can also map custom computation

```
def mean_over_space(ds):
    return ds.mean(dim=["x", "y"])

dt.map_over_subtree(mean_over_space)
```

How trees have helped us enable Open Science

- Testing of cloud services against our server (on-prem) hosted services through
  - Before we can offer cloud subsetting services we have to test that the services are consistent with our on-prem services
- GES DISC has fully onboarded 48,000 granules to cloud services (that's about 86 terabytes worth of data)!
  - With more in the works!
- Plans to use DataTree for our cloud subsetting service!

Image source: https://www.earthdata.nasa.gov/eosdis/cloud-evolution

NASA Open Science Initiative!

committed to ..

- Open sharing of software, data, information and development of software that adds value to Earth science data products

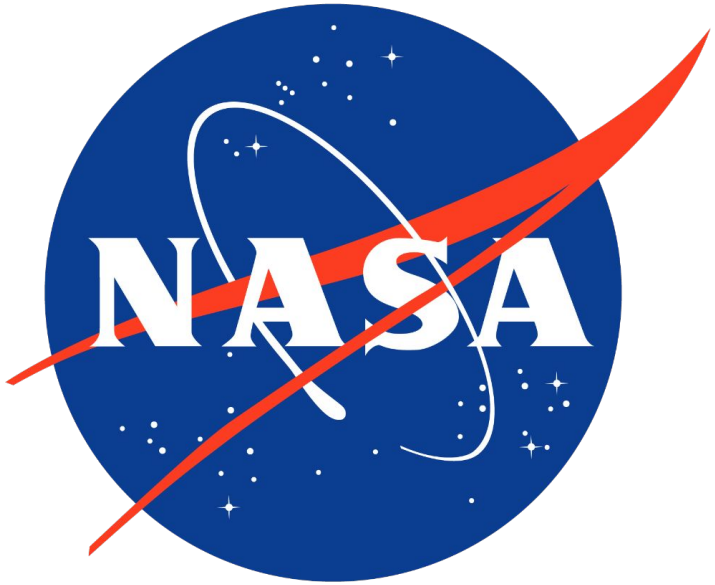
- That commitment is shown through support of NASA scientists and engineers as developers of open-source software!

Image source: https://www.earthdata.nasa.gov/technology/open-science

A story of a successful partnership

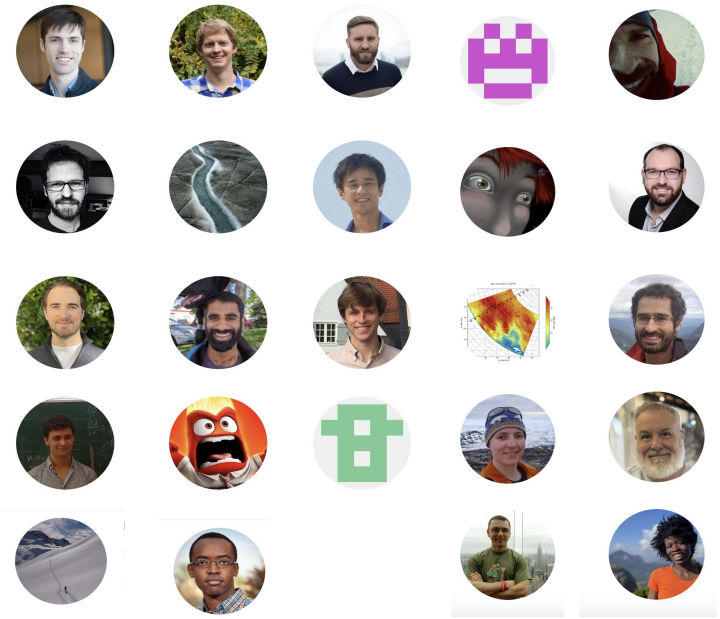
- Tom prototyped xarra-DataTree while working at Columbia Uni.
- It’s a semi-official prototype for a couple years
- NASA EOSDIS engineers are interested in integrating DataTree into internal tools
- NASA engineers are re-tasked to help integrate DataTree into xarray upstream (inc. Owen Littlejohns and Matt Savoie)
- Work is done by NASA folks with regular supervision from xarray team (Tom Nicholas + Stephan Hoyer)
- Plan to make xarray.DataTree public in next xarray release!

“You shouldn’t have to download the whole earth to do earth science”



Thank you!

- Everyone at GES DISC
  - Special thanks to Nicholas Lenssen
- Our colleagues at EOSDIS and NSIDC: Owen Littlejohns and Matt Savoie
- The whole xarray and DataTree team!



Sources

All images are licensed under the Adobe Creative commons license unless otherwise noted