OPeNDAP in the Cloud

Accelerating Scientific Discovery Through Exploratory Open Data Access

04/23/2025, 2pm EDT

Miguel A. Jimenez Urias, PhD

Computational Oceanographer | Sc. Community Director

James Gallagher

President of OPeNDAP | Principal Software Engineer

Agenda

- Intro to OPeNDAP in the Cloud
- Data Access Demos
  - → Requires Earthdata Login account
  - Web browser / traditional OPeNDAP
  - Python demos
    - → https://github.com/OPENDAP/NASA-tutorials
    - → How to find OPeNDAP URLS
    - → How to add parallelism / cloud-performance
- Q & A

OPeNDAP to Address NASA's Data Challenges

- Growing collection of Archival Datasets (~100 Pbs of data.[1])
- Data is heterogeneous, hierarchical
- Different file formats (Netcdf, HDF5, HDF4-EOS2, HDF4)
- Different levels of processing

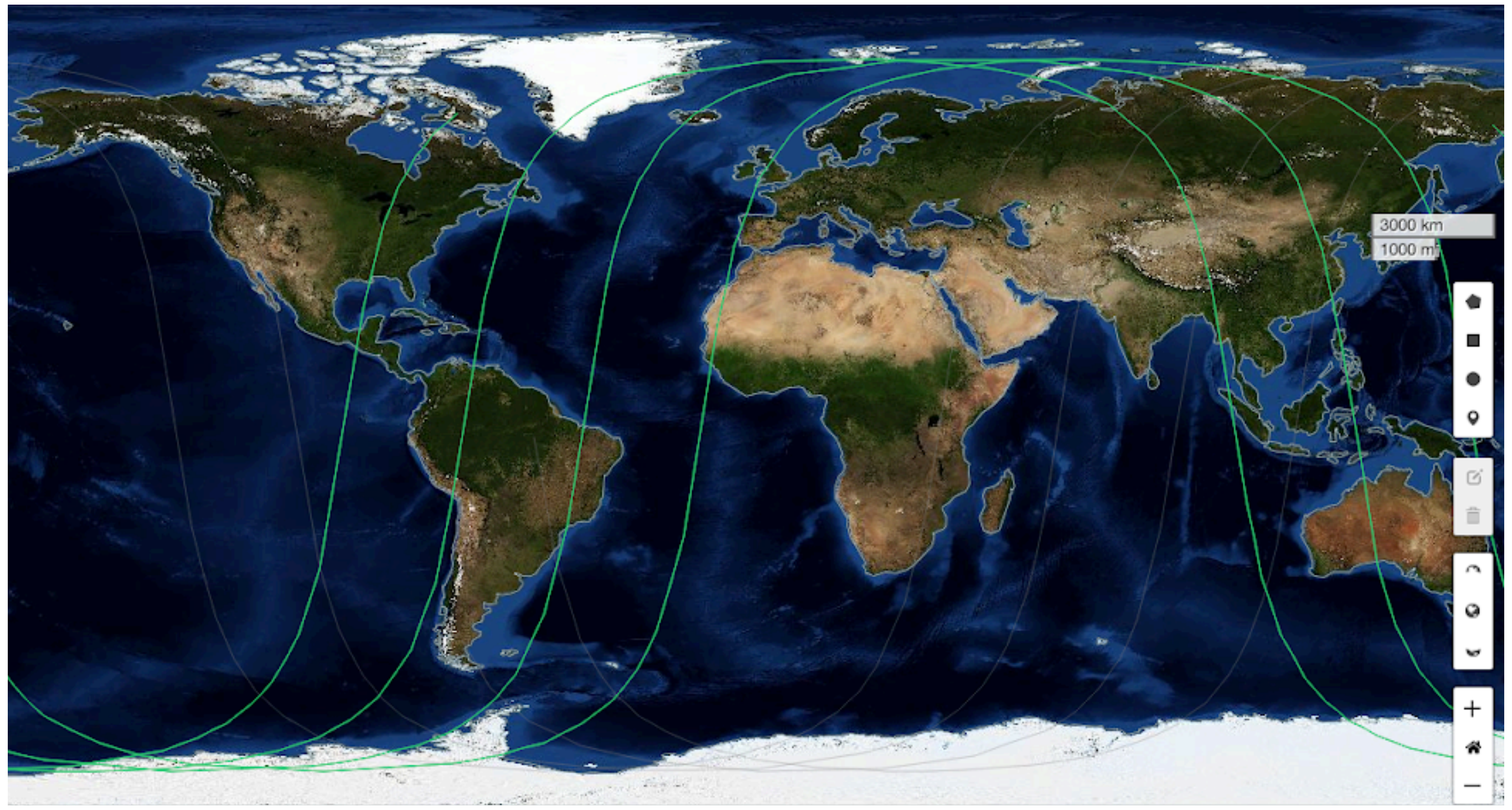

Figure 1. Swaths of Land Surface Temperature from ECOSTRESS (Level 2), available in hierarchical HDF file formats. Source: https://search.earthdata.nasa.gov/

More processed!

Table 1. Overview of different data processing levels used by NASA[2]

Data Level	Description
Level 0	Unprocessed instrument and payload data at full resolution
Level 1A	Same as Level 0, but time referenced and with ancillary information
Level 1B	Level 1A data processed to instrument units
Level 1C	Level 1B data with new variable to describe the spectra
Level 2	Derived geophysical variables at same resolution and locations L1
Level 2A	L2 data plus information derived from geolocated instrument data
Level 2B	L2A data that has been processed to instrument units
Level 3	Variables mapped to uniform space-time grid scales.
Level 3A	L3 data with periodic summaries (weekly, 10-day, monthly)
Level 4	Model output or results from analysis of lower level data

[1] Earth Science Data Systems 2024 Fiscal Year Highlights. From www.earthdata.nasa.gov

[2] https://www.earthdata.nasa.gov/learn/earth-observation-data-basics/data-processing-levels

What is OPeNDAP?

Open-source **P**roj**e**ct for a **N**etwork **D**ata **A**ccess **P**rotocol

URL ≈ Remote Dataset

- Designed jointly by scientists, developers and systems architects[3]
  - Share/download remote data
  - Data-proximate subsetting
  - Discipline-neutral
  - Interoperable

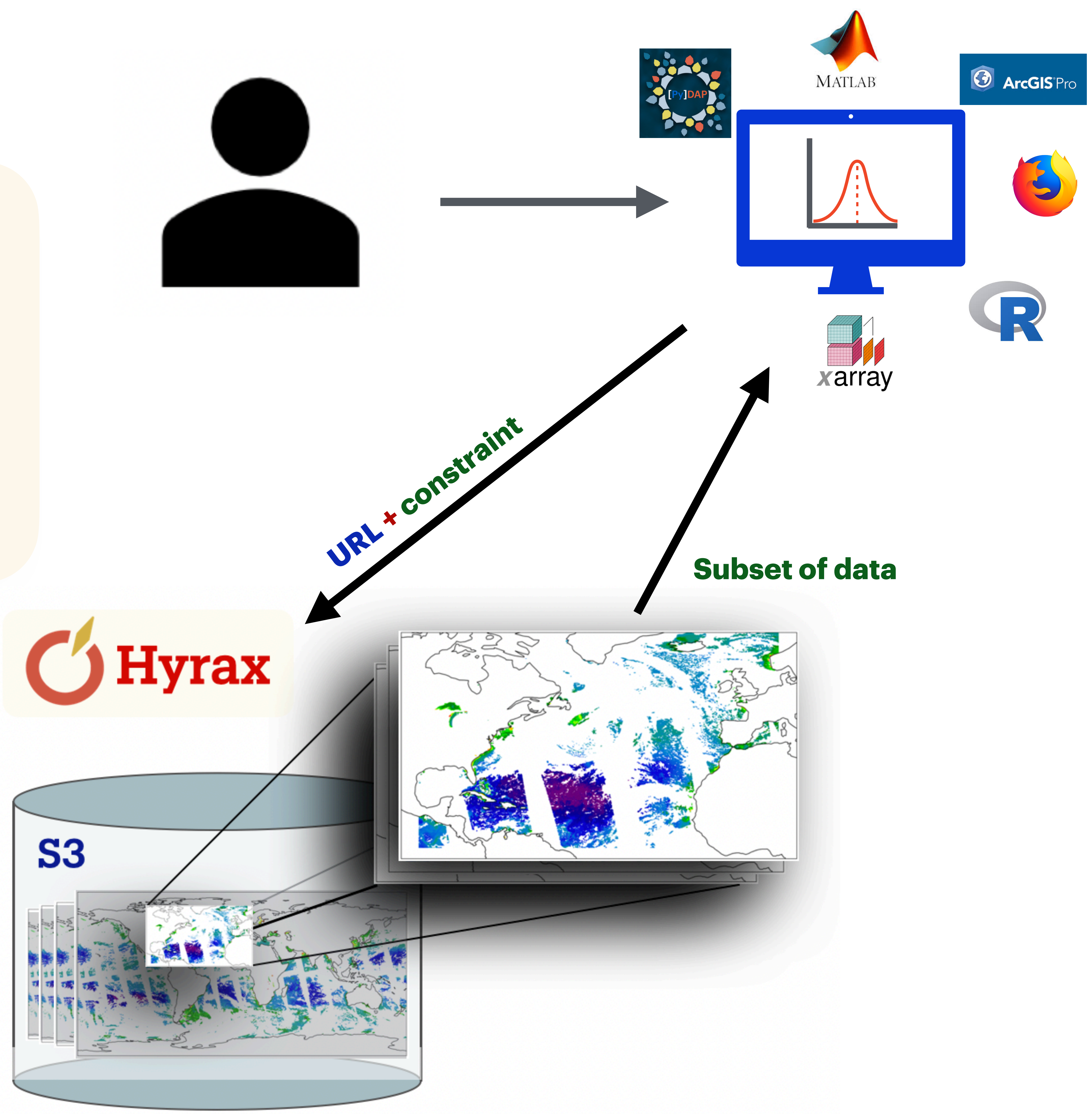

Figure 2. A data user workflow for accessing Plankton, Aerosol, Cloud ocean Ecosystem[5] (PACE) data via Hyrax data server.

What is Hyrax ?

OPeNDAP's flagship data server[4]

[3] https://www.opendap.org/about/history/
[4] https://www.opendap.org/software/hyrax-data-server/
[5] https://pace.oceansciences.org

What is OPeNDAP?

Open-source **P**roj**e**ct for a **N**etwork **D**ata **A**ccess **P**rotocol

- OPeNDAP implement the DAP Protocol
  - Covers complex, hierarchical datasets
  - Compatible with Climate and Forecast (CF) Metadata Conventions[6]

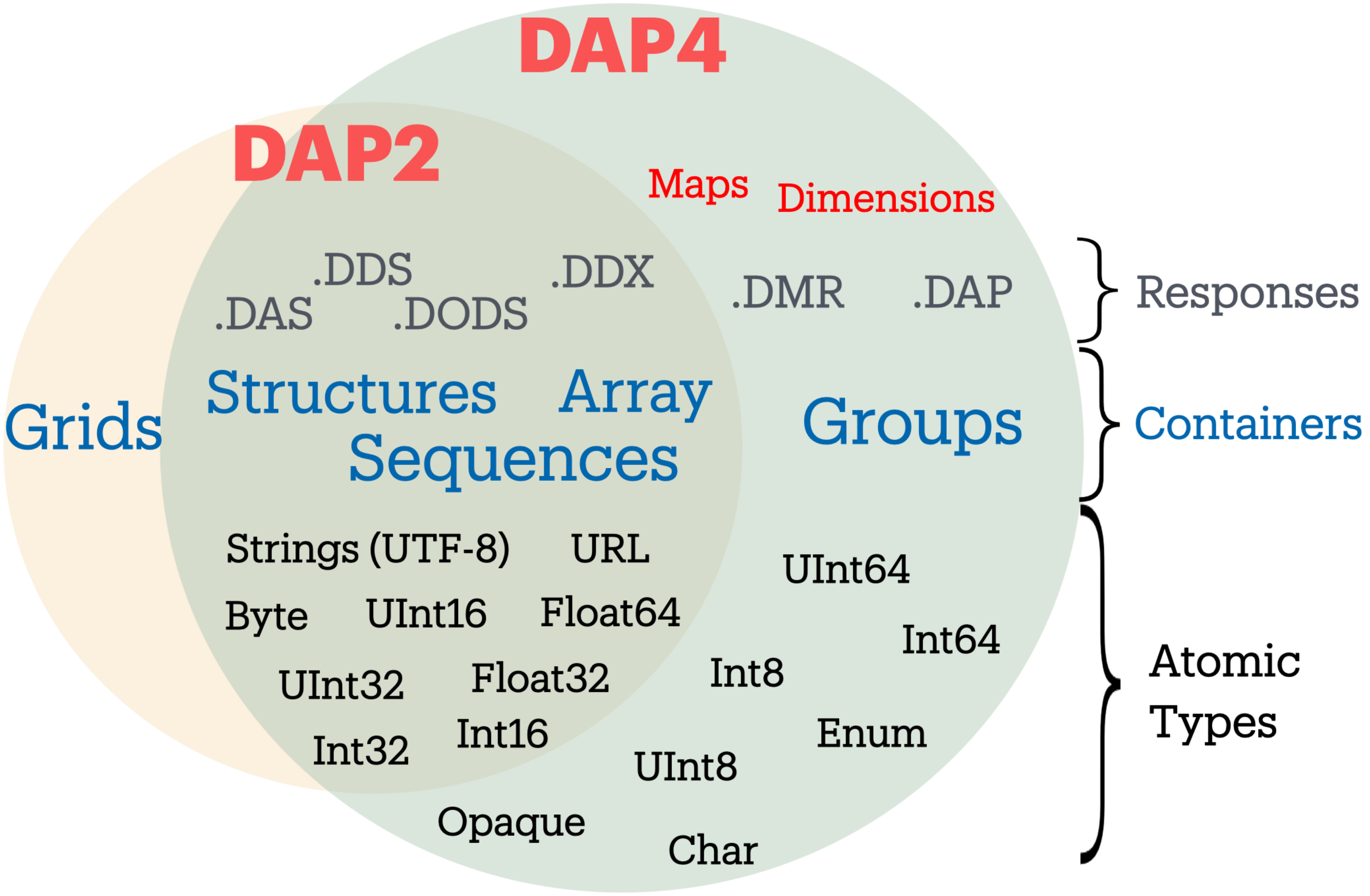

Figure 3. DAP2 and **DAP4** protocols are implemented by OPeNDAP's Hyrax Data Server. It describes complex, hierarchical data with different levels of processing, and is widely used by NASA.

[6] https://cfconventions.org/

What is OPeNDAP?

Open-source **P**roj**e**ct for a **N**etwork **D**ata **A**ccess **P**rotocol

- Provides a separated view of the Metadata
  - Allows for lazy inspection / evaluation
  - Performant cloud access and subsetting via [7] **DMR++**!

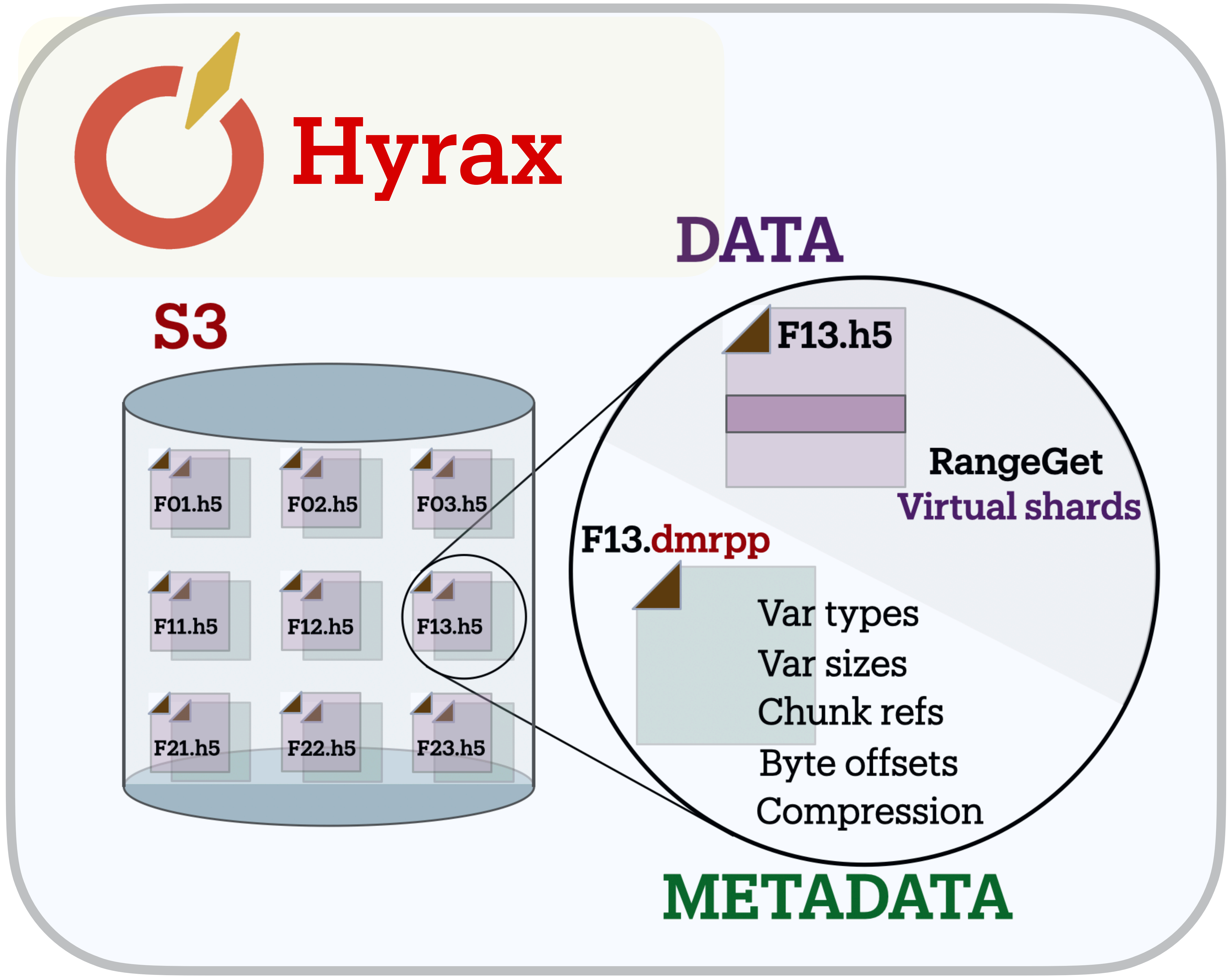

Figure 4. OPeNDAP produces a rich metadata view of the files, which maps the bytes of variables within the file, compression, without even opening it! This results in a huge performance gain!!!

[7] https://www.opendap.org/documentation/

How to OPeNDAP in the Cloud?

- Search using the **C**ommon **M**etadata **R**epository (CMR) or Earthdata Search to find
  - collection_id and granule_id
  - OPeNDAP urls

- Hyrax uses DMR++ for efficient access and cloud-performant subsetting of
  - (NetCDF-4, HDF5, HDF4-EOS2, HDF4)

- When using Hyrax directly, you can specify one of the following download option:
  - NetCDF-4
  - Binary dap (OPeNDAP native)
  - CSV

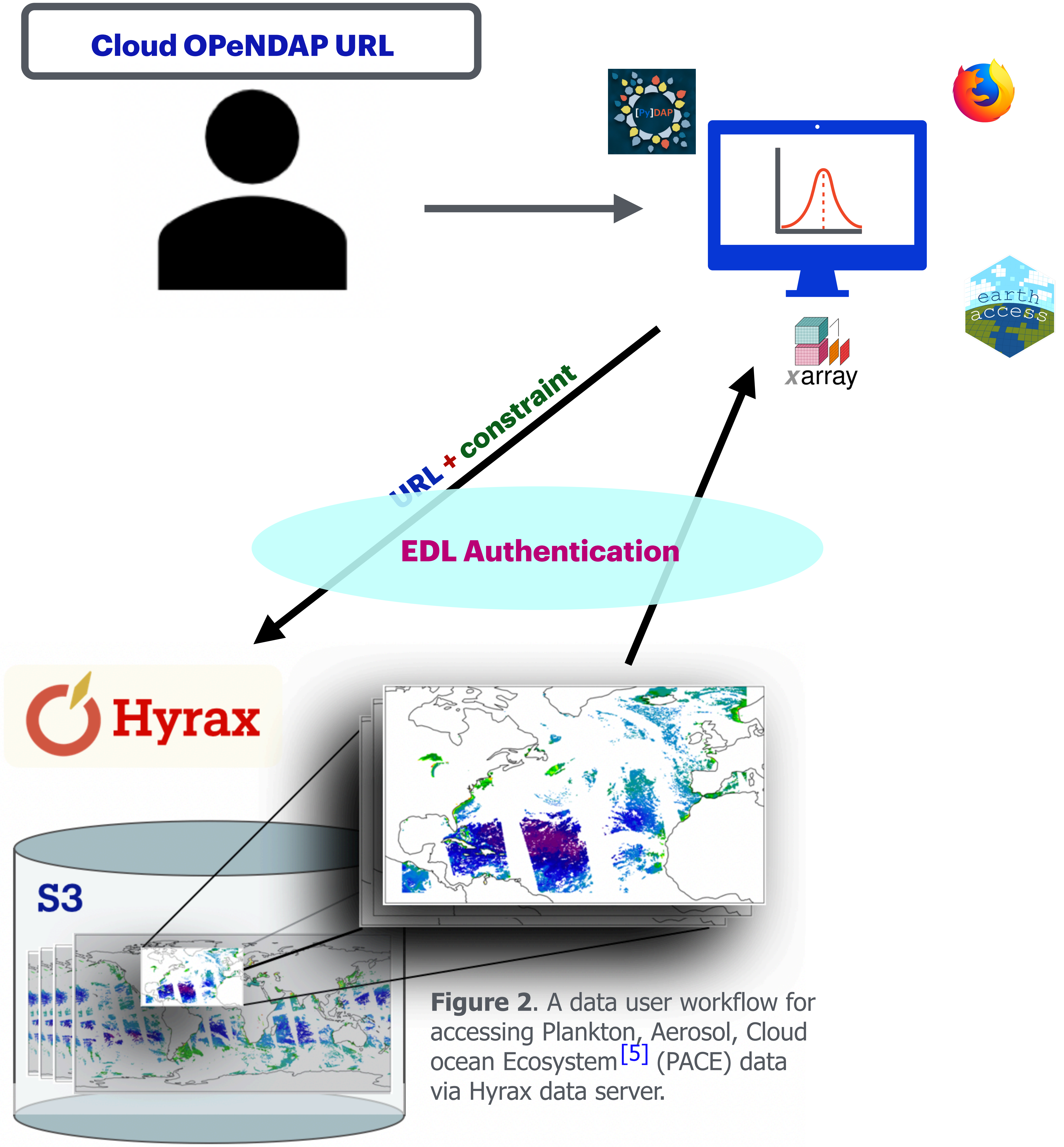

Figure 2. A data user workflow for accessing Plankton, Aerosol, Cloud ocean Ecosystem[5] (PACE) data via Hyrax data server.

Cloud OPeNDAP URL?

www.opendap.earthdata.nasa.gov/collections/collection_id/granules/granule_id

[5] https://pace.oceansciences.org

Data Access Demos!

Example Datasets

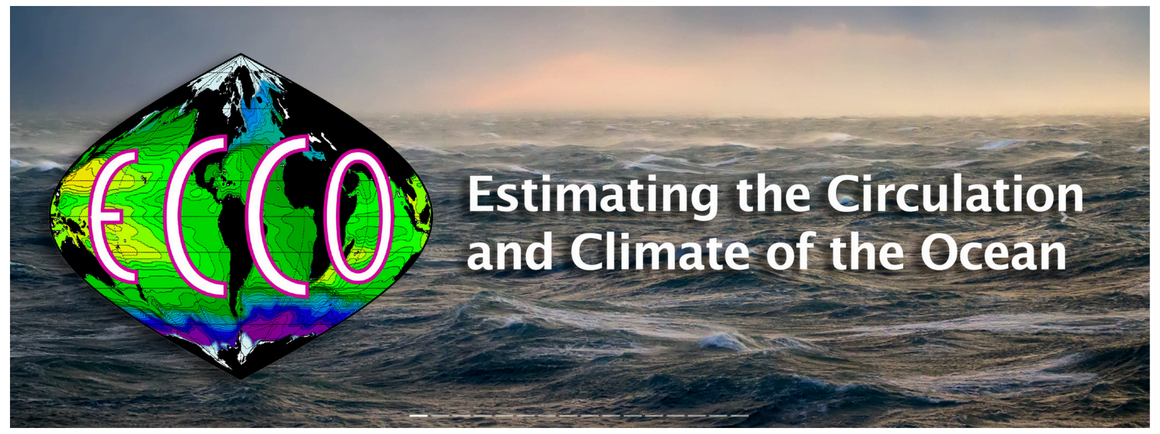

Figure 5. ECCO data product.[8]

Figure 6. ECOSTRESS[9] data product.

and more...

- Web browser / traditional OPeNDAP approach
  → https://search.earthdata.nasa.gov
- Dask-enabled parallelism:
  → https://github.com/OPENDAP/NASA-tutorials
  → Xarray[10] with pydap[11] as an engine
  → earthaccess[12] + VirtualiZarr[13] & DMR++

Requirements

- Active Earthdata Login (EDL) authentication
  → https://urs.earthdata.nasa.gov

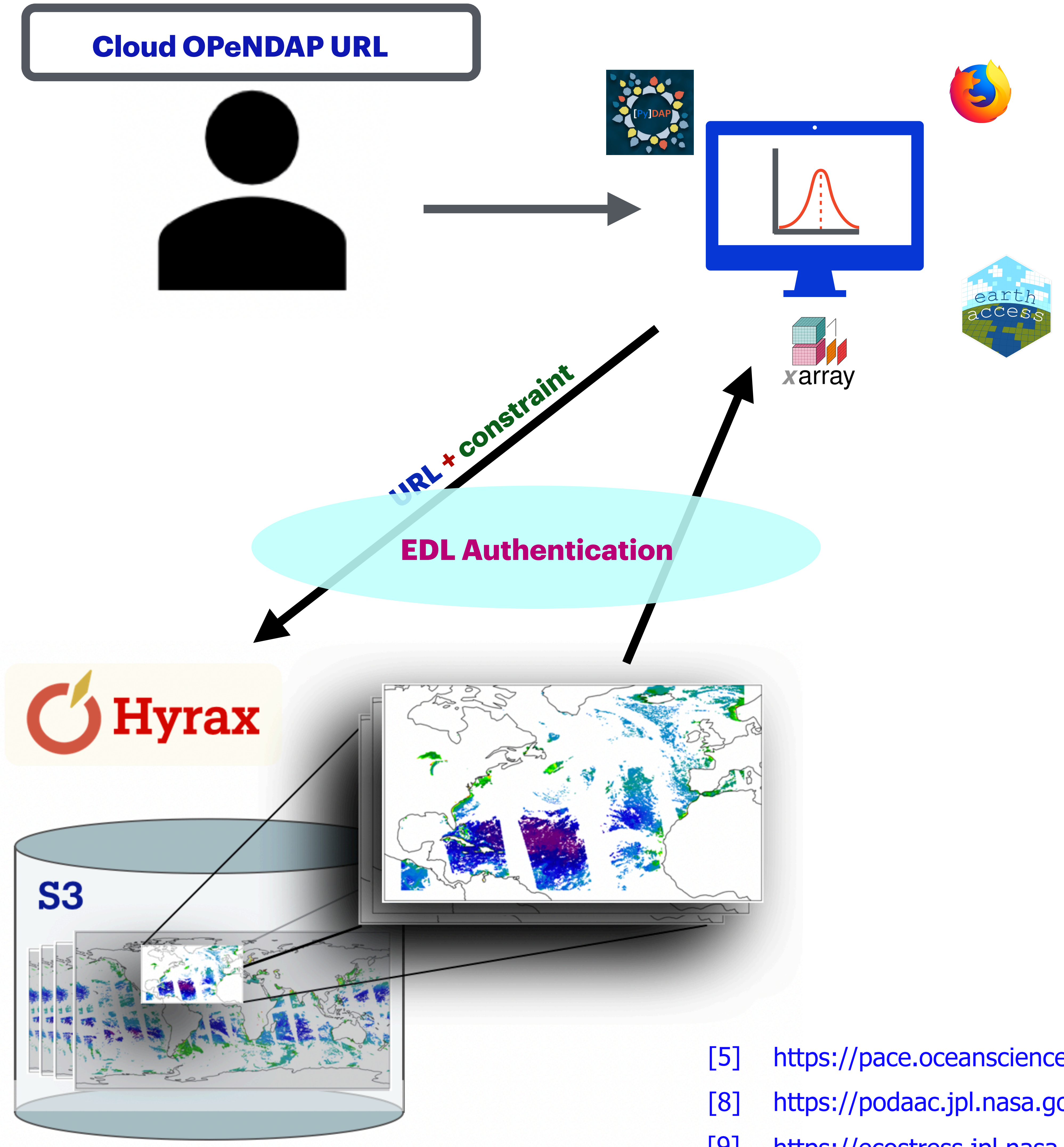

Figure 2. A data user workflow for accessing Plankton, Aerosol, Cloud ocean Ecosystem[5] (PACE) data via Hyrax data server.

[5] https://pace.oceansciences.org
[8] https://podaac.jpl.nasa.gov/ECCO
[9] https://ecostress.jpl.nasa.gov/
[10] https://xarray.dev/
[11] https://pydap.github.io/pydap
[12] https://earthaccess.readthedocs.io
[13] https://virtualizarr.readthedocs.io

Thank You

Keep me posted

Lets connect!

Collaborate!

Questions?

Xarray + Pydap Approach

Features

Consolidates 100s of OPeNDAP URLS

Parallel access / analysis with xarray

Consolidate metadata

A single object that references O(100)s of OPeNDAP URLS

```
%%time
consolidate_metadata(URLs_CE, cached_session)
```

```
datacube has dimensions {'tile[0:1:12]', 'k[0:1:49]', 'time[0:1:0]', 'j[0:1:89]', 'i[0:1:89]'}
CPU times: user 1.74 s, sys: 591 ms, total: 2.33 s
Wall time: 19.7 s
```

Virtually aggregate all URLS

```
%%time
pyds = xr.open_mfdataset(URLs_CE, engine='pydap', session=cached_session, parallel=True, combine='nested', concat_dim='time')
pyds
```

```
CPU times: user 3.23 s, sys: 799 ms, total: 4.03 s
Wall time: 3.69 s
```

xarray.Dataset

▸ Dimensions: (**time**: 216, **k**: 50, **tile**: 13, **j**: 90, **i**: 90)

▾ Coordinates:

i	(i)	int32	0 1 2 3 4 5 6 ... 84 85 86 87 88 89
j	(j)	int32	0 1 2 3 4 5 6 ... 84 85 86 87 88 89
k	(k)	int32	0 1 2 3 4 5 6 ... 44 45 46 47 48 49
tile	(tile)	int32	0 1 2 3 4 5 6 7 8 9 10 11 12
time	(time)	datetime64[ns]	2000-01-16T12:00:00 ... 2000-01-...

▾ Data variables:

SALT	(time, k, tile, j, i)	float32	dask.array<chunksize=(1, 50, 13, 90, 90)...
THETA	(time, k, tile, j, i)	float32	dask.array<chunksize=(1, 50, 13, 90, 90)...

▸ Indexes: (5)

[11] https://pydap.github.io/pydap

Earthaccess approach (exploiting Virtualizarr and DMR++)

Parses DMR++ to achieve Zarr-like performance

- Query data, load, analyze

  Earthaccess uses CMR

- Uses Dark for parallel AWS reads of DMR++ files

- Read, subset, aggregate via xarray

[3]:
```
%%time
results = earthaccess.search_data(
    count=100,
    temporal=("2017-12-13", "2023-12-13"),
    short_name="MUR-JPL-L4-GLOB-v4.1"
)
```

```
CPU times: user 108 ms, sys: 18.8 ms, total: 127 ms
Wall time: 1.7 s
```

[4]:
```
%%time
from earthaccess.virtualizarr import open_virtual_dataset
vds = open_virtual_dataset(results, access="direct", concat_dim="time", coords='minimal', compat='override', combine_attrs="drop_conflicts")
vds
```

```
CPU times: user 9.45 s, sys: 783 ms, total: 10.2 s
Wall time: 18.7 s
```

[4]: xarray.Dataset

▶ Dimensions: (**time**: 100, **lat**: 17999, **lon**: 36000)

▼ Coordinates:

lat	(lat)	float32	-89.99 -89.98 ... 89.98 89.99
lon	(lon)	float32	-180.0 -180.0 ... 180.0 180.0
time	(time)	datetime64[ns]	2017-12-13T09:00:00 ... 2018-03-...

▼ Data variables:

analysed_sst	(time, lat, lon)	float32	dask.array<chunksize=(1, 1023, 2047), me...
analysis_error	(time, lat, lon)	float32	dask.array<chunksize=(1, 1023, 2047), me...
dt_1km_data	(time, lat, lon)	timedelta64[ns]	dask.array<chunksize=(1, 1447, 2895), me...
mask	(time, lat, lon)	float32	dask.array<chunksize=(1, 1447, 2895), me...
sea_ice_fraction	(time, lat, lon)	float32	dask.array<chunksize=(1, 1447, 2895), me...

▶ Indexes: (3)

▶ Attributes: (0)

[6]:
```
print(f"{vds.nbytes / 1e12} TB")
```

```
1.555113816796 TB
```

Gallagher, J. and Ayush Nag, (2024)

Earthaccess approach (exploiting Virtualizarr and DMR++)

ESIP (2024)

Gallagher, J. and Ayush Nag, (2024)