Fake it 'till you make it

Zarr-like Access of Existing NetCDF4 Datasets

Lucas Sterzinger

Mentors: Chelle Gentemann (Farallon Institute), Julia Kent (NCAR), Kevin Paul (NCAR)

July 28, 2021

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

The Problem

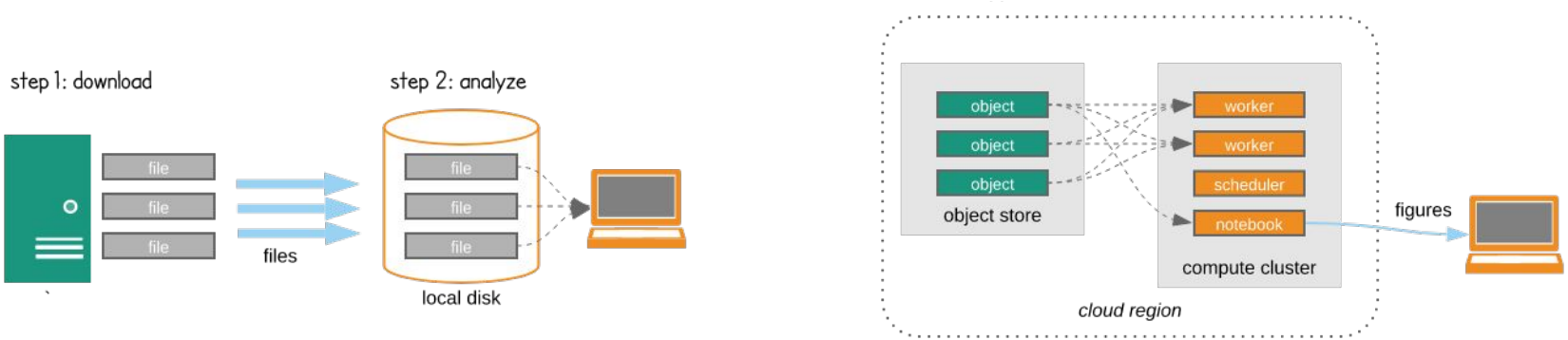
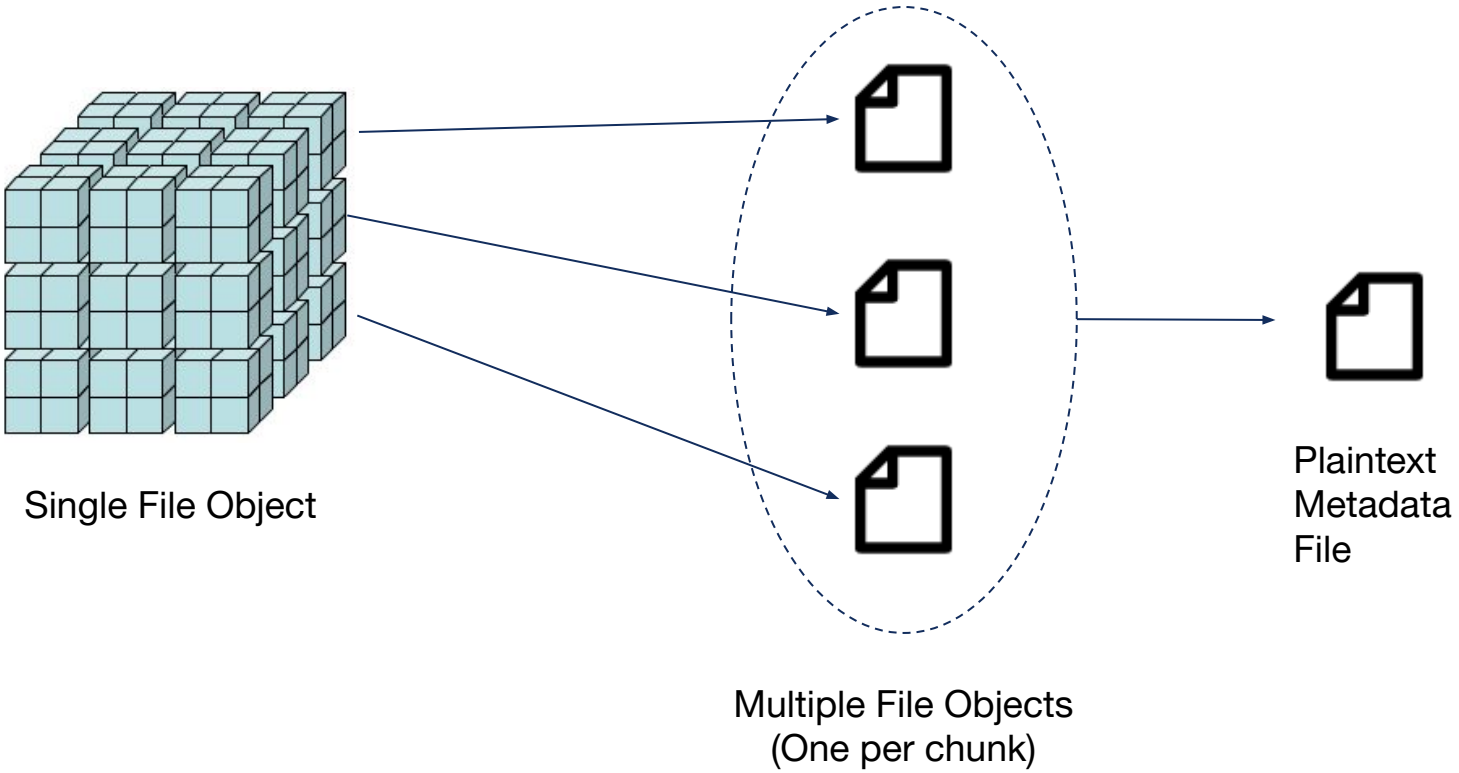


Image Source - Pangeo/Ryan Abernathy

- Peta(Exa?)bytes of geophysical data now available in the cloud
- Much of the data are in NetCDF4/HDF5 format
  - Not efficient for cloud computing

NetCDF vs Zarr

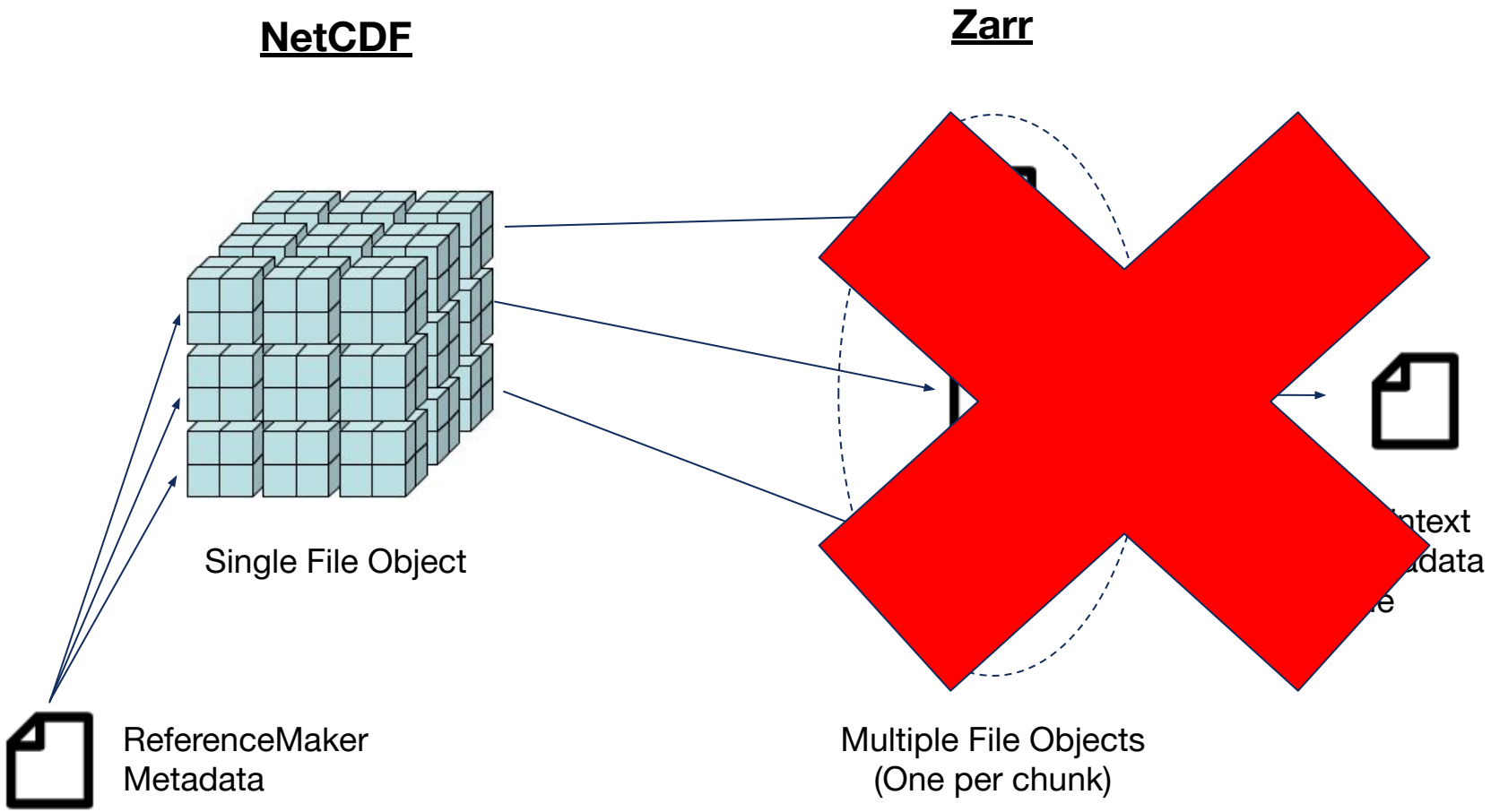


ReferenceFileSystem

ReferenceMaker/ReferenceFileSystem:

- Part of Intake group's **`fsspec`** project
- Access existing NetCDF/HDF files ***as if they are Zarr***
- Zarr-like Metadata files (few MB each)
  - Remote file address (Azure Blob, AWS S3, etc)
  - Data variable and chunking info
- Extracts byte-ranges for individual variables/chunks
  - Instead of individual chunk files in Zarr

ReferenceMaker



Creating JSONS

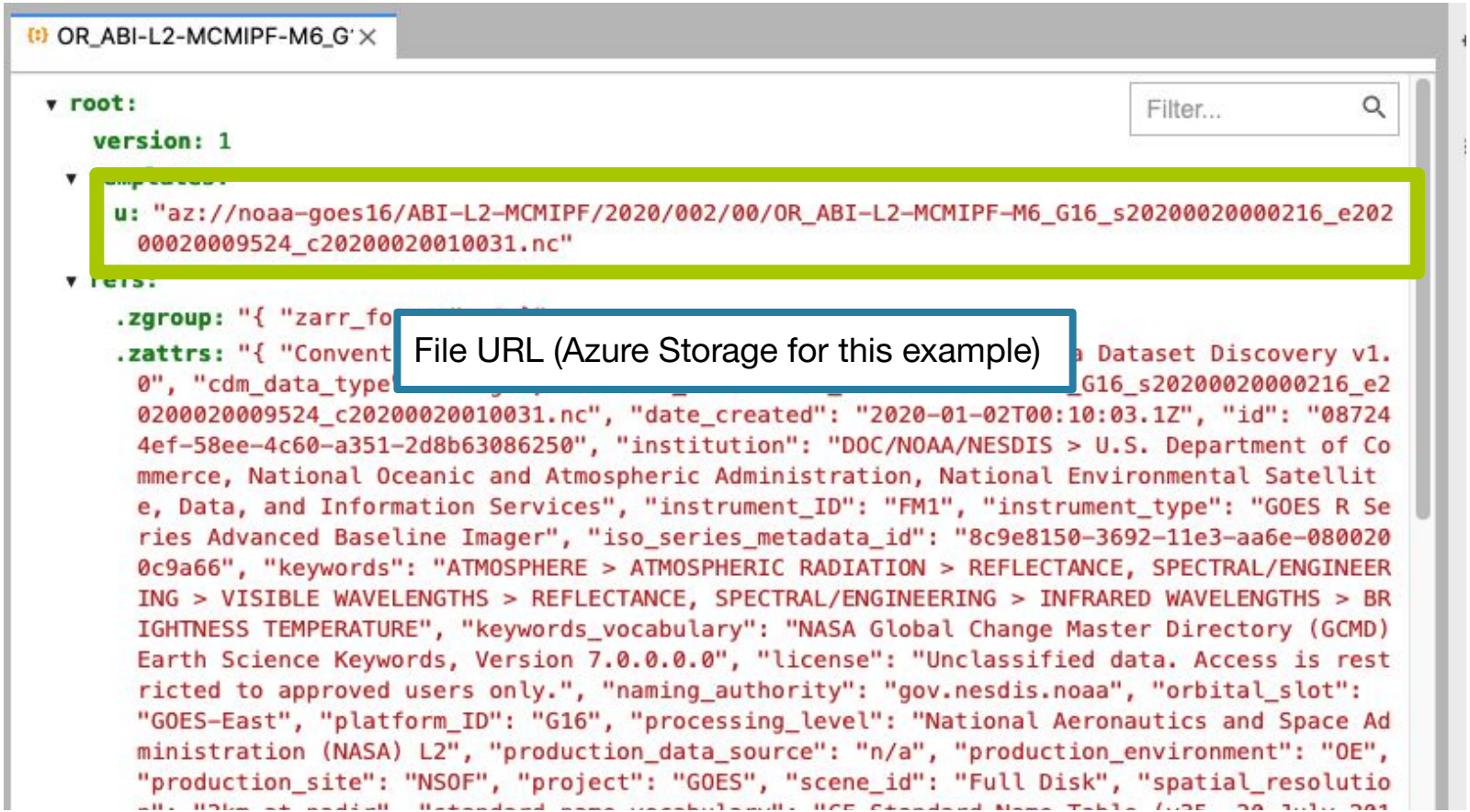
```python
@dask.delayed
def gen_json(url):
    so = dict(
        mode="rb", anon=True, default_fill_cache=False, default_cache_type="none"
    )

    outpath = './jsons/' + url.split('/')[-1].split('.nc')[0] + '.json'

    with fsspec.open(url, **so, account_name='goeseuwest') as inf:
        h5chunks = SingleHdf5ToZarr(inf, u, inline_threshold=300)

        with open("./reference.json", 'wb') as outf:
            outf.write(ujson.dumps(h5chunks.translate()).encode())
```

Easily distributed with Dask!

Read remote file and process chunks

Write metadata to file

Reference JSON Example



Reference JSON Example


```
ministration (NASA) L2", "production_data_source": "n/a", "production_environment": "OE",
"production_site": "NSOF", "project": "GOES", "scene_id": "Full Disk", "spatial_resolutio
n": "2km at nadir", "st                                    ame Table (v35, 20 July 201
6)", "summary": "Multip                                    ud and Moisture Imagery Pro
ducts are digital maps                                     ndows at visible, near-IR,
and IR bands.", "time_c                                    "time_coverage_start": "20
20-01-02T00:00:21.6Z", "timeline_id": "ABI Mode 6", "title": "ABI L2 Cloud and Moisture I
magery" }"
CMI_C01/.zarray: "{ "chunks": [ 226, 226 ], "compressor": { "id": "zlib", "level": 1 }, "dt
  ype": "<i2", "fill_value": -1, "filters": null, "order": "C", "shape": [ 5424, 5424 ], "z
  arr_format": 2 }"
CMI_C01/.zattrs: "{ "_ARRAY_DIMENSIONS": [ "y", "x" ], "_FillValue": -1, "_Netcdf4Dimid":
  0, "_Unsigned": "true", "add_offset": 0.0, "ancillary_variables": "DQF_C01", "cell_method
  s": "t: point area: sum (interval: 0.000028 rad)", "coordinates": "band_id_C01 band_wavel
  ength_C01 t y x", "downsampling_method": "average", "grid_mapping": "goes_imager_projecti
  on", "long_name": "ABI Cloud and Moisture Imagery reflectance factor", "resolution": "y:
  0.000056 rad x: 0.000056 rad", "scale_factor": 0.00031746001332066953, "sensor_band_bit_d
  epth": 10, "standard_name": "toa_lambertian_equivalent_albedo_multiplied_by_cosine_solar_
  zenith_angle", "units": "1", "valid_range": [ 0, 4095 ] }"
▾ CMI_C01/0.4: [] 3 items
    0: "{{u}}"
    1: 253643
    2: 468
```


Variable (CMI_C01) chunk info, compression, shape, etc

Single Chunk Byte Range

Reference JSON Example

Set “reference” filesystem and specify JSON file

```
fs = fsspec.filesystem('reference',
fo='reference.json',
remote_protocol='az',
remote_options={'account_name':'goeseuwest'},
simple_templates=True)
```

Specify remote file protocol (Azure here) and authentication

```
m = fs.get_mapper("")

ds = xr.open_dataset(m, engine='zarr')
```

Pass mapper to xarray and specify Zarr engine

GOES Workflow

Microsoft Azure

Data stored in same region as compute

4 CPU, 32 GB Memory,
8 Dask Workers

- Created a test workflow on Microsoft Planetary Computer
- Analysis of 24 hours of GOES data
- Test access to data stored in Azure in 3 formats:
  - NetCDF4
  - NetCDF4 + ReferenceMaker
  - Zarr

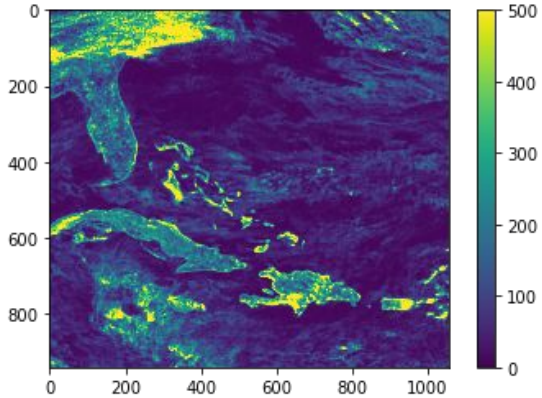
GOES Workflow

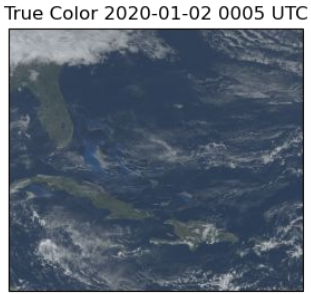
Microsoft Azure

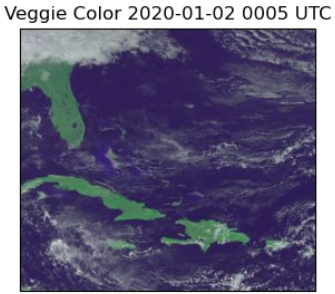
Data stored in same region as compute

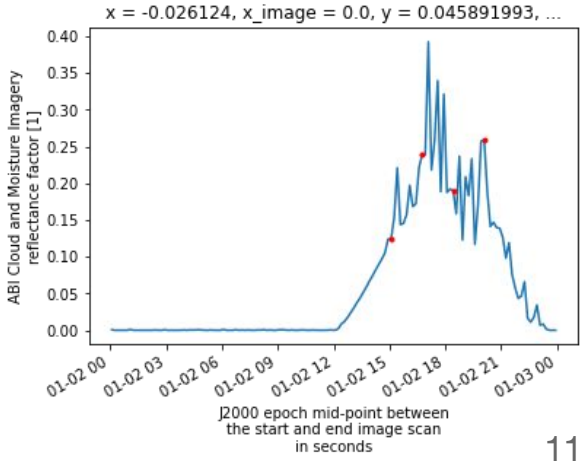
Microsoft Planetary Computer

4 CPU, 32 GB Memory,
8 Dask Workers









GOES Processing Times

For 24 hours of GOES-16 “ABI-L2-MCMIPF” product data

Format	**Preprocess Time**	**Data Open Time**	**Workflow Time**	**Extra Storage**
Remote NetCDF4	0 min	10 minutes	40 min	0 GB
Zarr	1 h 38 min	30 seconds	4 min	52 GB
ReferenceFileSystem	1 h 25 min	35 seconds	5 min 30 s	***416 MB***

- NetCDF has no upfront preprocess time cost
  - But very slow open and workflow times
- Zarr comes at cost of high storage costs (full data duplication)
- ReferenceFileSystem has speed of Zarr but ***<1% storage cost***
  - Reference file can be compressed further

The Implications

- ReferenceMaker/FileSystem harnesses the cloud-optimization of Zarr without converting any data
- Allows current institutions to cheaply make existing cloud-hosted data more optimized
- Reference files can be also created, hosted, and shared by third parties
- Automatable
  - e.g. Pangeo Forge

Future Work Needed

- Project is still in early development
- Test compatibility across different datasets
- Scale to larger datasets and compare performance to Zarr
- Test performance on a local parallel filesystem (like GLADE)

Acknowledgements

Thank you to the ReferenceFileSystem team!

- Special thanks to Martin Durant (Anaconda) and Rich Signell (USGS) for their help learning and contributing to this project
- https://github.com/intake/fsspec-reference-maker

Thank you to Kevin Paul (NCAR), Julia Kent (NCAR), and Chelle Gentemann (Farallon Institute) for their amazing mentorship through the SIParCS program

Thanks to the SIParCS team for working so hard to make a fully-online internship experience so successful

Thanks to Microsoft for providing credits for Planetary Computer

Link to site with interactive code examples, tutorials, resources, and contact info

https://lucassterzinger.com/2021-siparcs-poster/

Please feel free to reach out with questions lsterzinger@ucdavis.edu or twitter @lucassterzinger

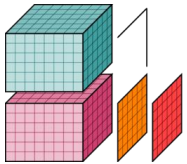
Community Matters

If you are a scientist that works with these tools

- Join the Pangeo community
- Talk to project maintainers about your workflow needs
- Open issues/bug reports on GitHub
- Submit pull requests, even if your code isn't 100% ready

PANGEO

Microsoft Planetary Computer



xarray

jupyter

Iris

DASK