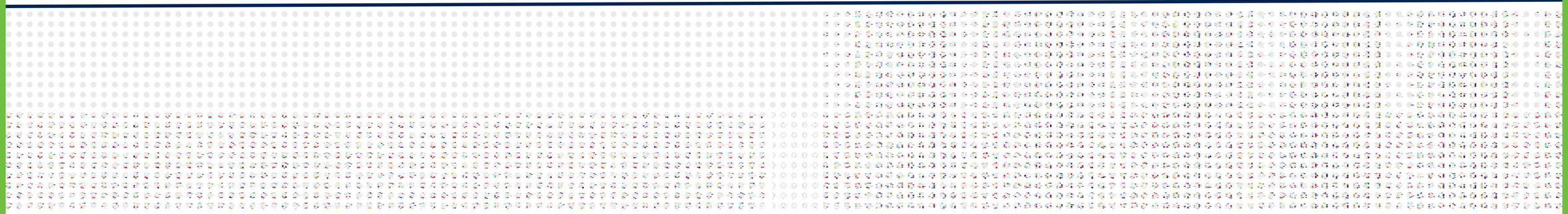
HDF5 ↔ Zarr

2020 ESIP Summer Meeting



The HDF Group

Aleksandar Jelenak

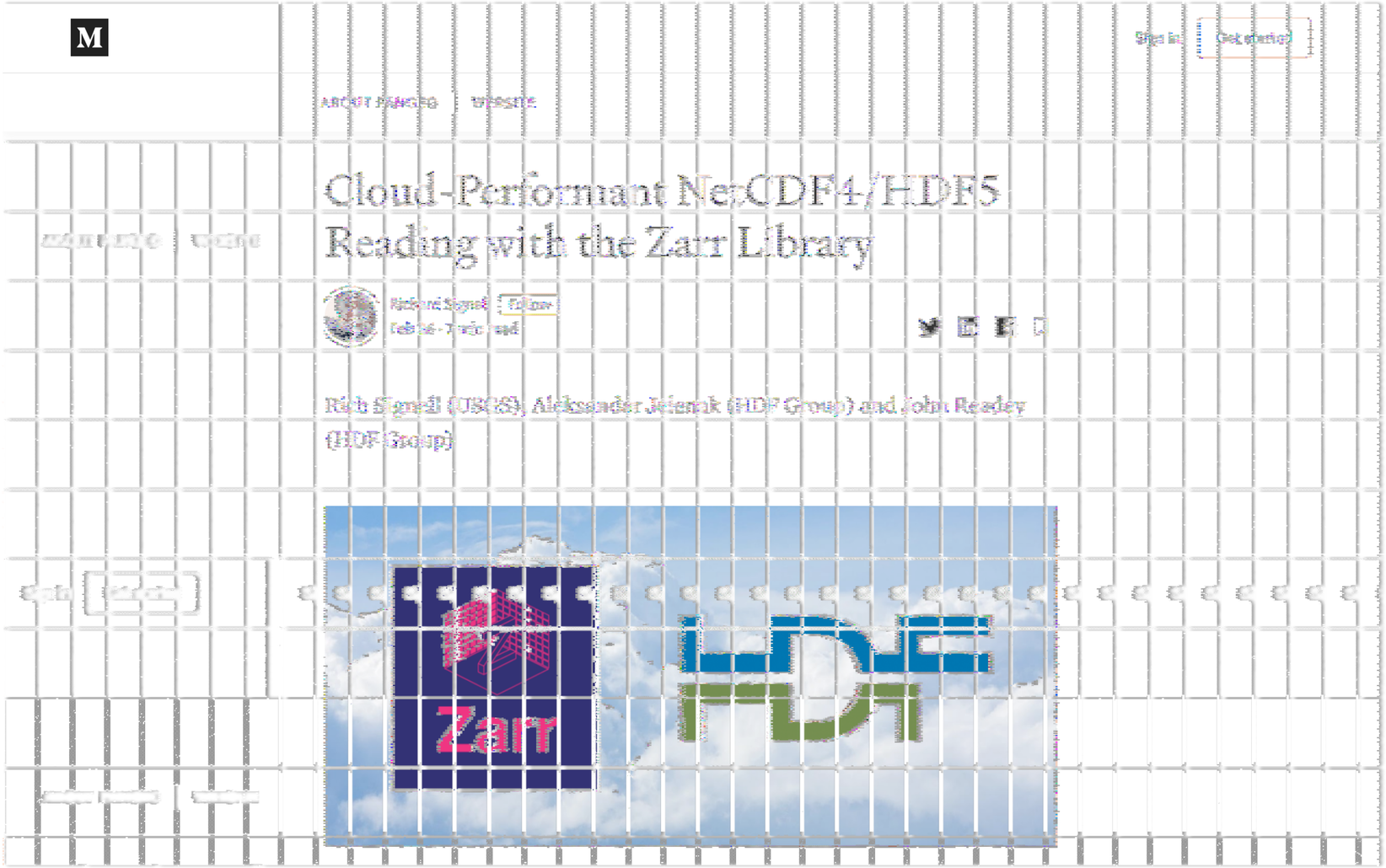
Overview

- Zarr ➔ HDF5*
- HDF5 ➔ Zarr

*
A portion of this work was supported by the United States Geological Survey (USGS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of United States Government.

Zarr → HDF5

The HDF Group



https://medium.com/pangeo/cloud-performant-reading-of-netcdf4-hdf5-data-using-the-zarr-library-1a95c5c92314

Zarr → HDF5: Implementation

The HDF Group

- **Created a new Zarr store: *FileChunkStore***
- **Developed Zarr JSON metadata for (HDF5) chunk file location: `.zchunkstore`**

```
"zeta/9.38": {
        "offset": 3883848970,
        "size": 6907818
    },
"zeta/9.39": {
        "offset": 3890756788,
        "size": 7879493
    },
"zeta/9.4": {
        "offset": 3648355033,
        "size": 6525250
    },
"zeta/9.40": {
        "offset": 3898636281,
        "size": 7132453
    }
```

- **Small fixes in the *zarr* and *xarray* Python packages to support a separate Zarr store for file chunks**

Zarr → HDF5: Performance comparison

The HDF Group

Zarr reading Zarr

```
%%time
ds = xr.open_zarr(fsspec.get_mapper('s3://pangeo-data-uswest2/esip/adcirc/adcirc_01d',
                                    anon=False, requester_pays=True))
```

```
CPU times: user 999 ms, sys: 61.4 ms, total: 1.06 s
Wall time: 3.6 s
```

Compute the max water level at each grid cell (reads all the data):

```
%%time
max_var1 = ds['zeta'].max(dim='time').compute()
```

```
CPU times: user 10.3 s, sys: 721 ms, total: 11.1 s
Wall time: 24.1 s
```

Zarr reading HDF5

```
%%time
ncfile = fsspec.open('s3://pangeo-data-uswest2/esip/adcirc/adcirc_01d.nc',
                     anon=False, requester_pays=True)
```

```
CPU times: user 70 µs, sys: 8 µs, total: 78 µs
Wall time: 79.2 µs
```

```
%%time
store = fsspec.get_mapper('s3://hdf5-zarr/adcirc_01d.nc.chunkstore', anon=True)
chunk_store = FileChunkStore(store, chunk_source=ncfile.open())

ds2 = xr.open_zarr(store, consolidated=True, chunk_store=chunk_store)
```

```
CPU times: user 100 ms, sys: 20.4 ms, total: 121 ms
Wall time: 341 ms
```

```
%%time
max_var2 = ds2['zeta'].max(dim='time').compute()
```

```
CPU times: user 11 s, sys: 1.01 s, total: 12 s
Wall time: 22.1 s
```

Zarr → HDF5: How to try out?

The HDF Group

- Python >= 3.6
- HDF5-1.10.6 library
- *pip install git+https://github.com/h5py/h5py.git*
- *pip install git+https://github.com/ajelenak/xarray.git@zarr-chunkstore*
- *pip install git+https://github.com/HDFGroup/zarr-python.git@hdf5*
- *pip install fsspec*
- HDF5-to-Zarr translator:
  https://gist.github.com/ajelenak/80354a95b449cedea5cca508004f97a9

Zarr → HDF5: Limitations

The HDF Group

- HDF5 dataset compact layout not supported
- HDF5 dataset data may be written by compressor/filter not supported by Zarr
- Storage system hosting the HDF5 file must allow partial file reading

HDF5 ➔ Zarr

The HDF Group

- HDF5 API access to Zarr data is provided by the HDF's Highly Scalabale Data Service (HSDS)
- Only for Zarr data in AWS S3
- HSDS object store schema is similar to Zarr: a combination of JSON and binary objects
- HSDS JSON is a superset of HDF5/JSON
- Still work in progress

HDF5 → Zarr: Implementation

The HDF Group

- **Using special HSDS schema chunking layout:** `H5D_CHUNKED_REF_INDIRECT`
- **This chunking layout is not supported by the HDF5 library**
- **Developed to enable HSDS access to chunks in HDF5 files in object storage**
- **Chunk information for one Zarr array is stored as an anonymous HDF5 compound dataset**
- **The compound datatype has 3 fields for: byte offset (always 0), chunk object size, and chunk object URI**
- **The HDF5 dataset representing the Zarr array has the** `H5D_CHUNKED_REF_INDIRECT` **layout and its value points to the anonymous HDF5 dataset with chunk location information**

HDF5 ➔ Zarr: Data Wrangling

The HDF Group

- **Because HSDS does not (yet) support the Blosc compressor, the original Zarr dataset was copied with the Zlib compressor instead:**

```
from sys import stdout
import zarr
import fsspec
from numcodecs import Zlib


src_root = zarr.open(
    fsspec.get_mapper('s3://pangeo-data-uswest2/esip/adcirc/adcirc_01d',
                      anon=False,
                      requester_pays=True),
    mode='r')

dest_root = zarr.open(
    fsspec.get_mapper('s3://hdf5-zarr/adcirc_01d.zarr', anon=False),
    mode='w')

zarr.copy_all(src_root, dest_root, shallow=False, without_attrs=False,
              log=stdout, if_exists='replace', dry_run=True,
              compressor=Zlib(level=6), filters=None)
```

HDF5 ➔ Zarr: Example translation

The HDF Group

Zarr array

```
Name               : /zeta
Type               : zarr.core.Array
Data type          : float64
Shape              : (720, 9228245)
Chunk shape        : (10, 141973)
Compressor         : Zlib(level=6)
No. bytes          : 53154691200 (49.5G)
Chunks initialized : 4680/4680
```

HDF5 dataset with chunk info

```
Type               : h5pyd.Dataset
Data type          : compound
Shape              : (72, 65)

Value:
[[(0, 1949049, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.0')
  (0, 2911533, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.1')
  (0, 2506163, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.2') ...
  (0, 4344724, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.62')
  (0, 5696617, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.63')
  (0, 4275725, 's3://hdf5-zarr/adcirc_01d.zarr/zeta/0.64')]
```

HDF5 ➔ Zarr: Limitations

The HDF Group

- Zarr array data may be written by compressor/filter not supported by HSDS

- Since both Zarr and HSDS are written in Python, it should be possible to add Zarr's *numcodecs* package to HSDS to resolve this limitation

THANK YOU!

ajelenak@hdfgroup.org