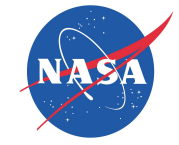


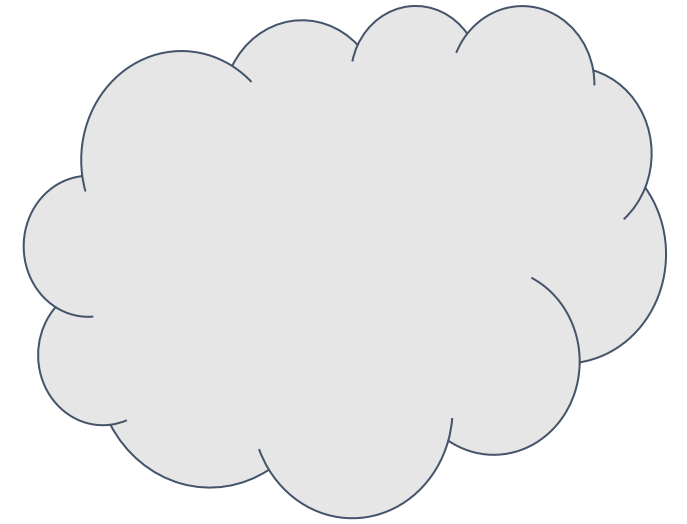
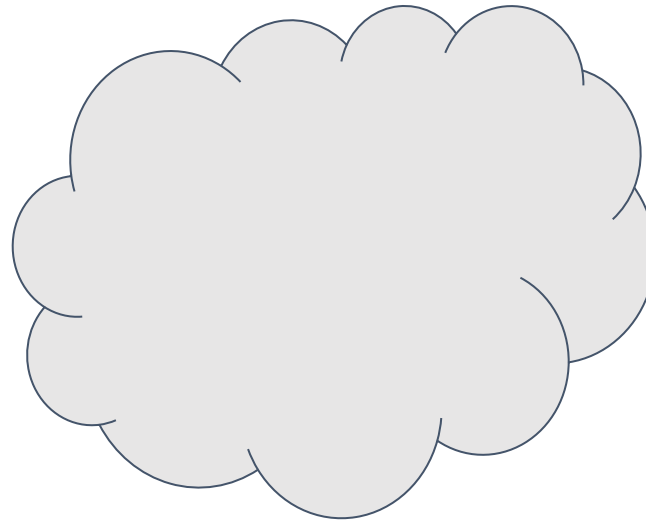
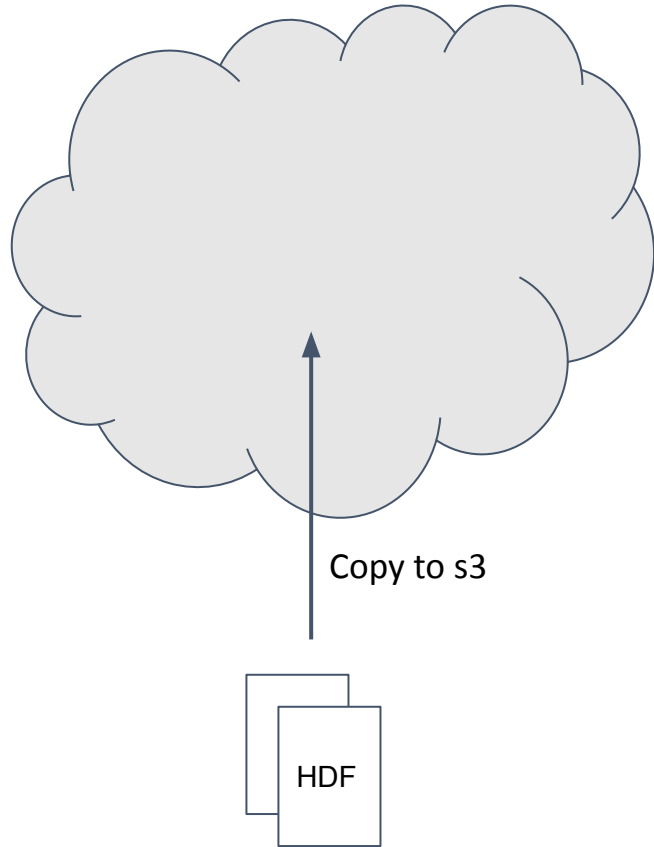
Metadata for geospatial, multi-dimensional Zarr arrays in the Cloud

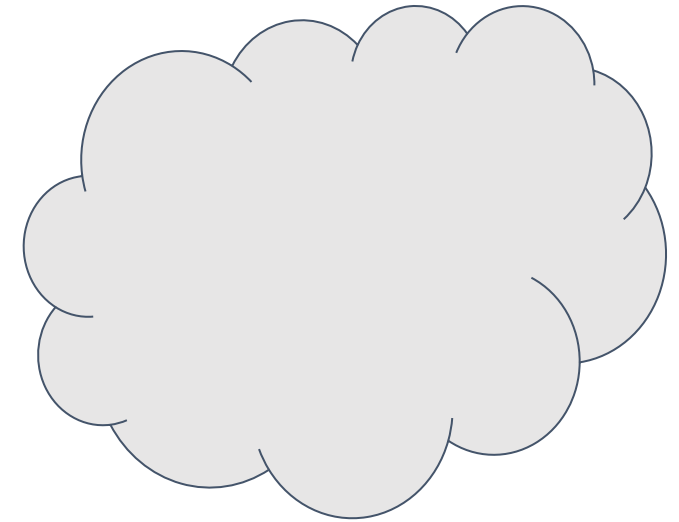
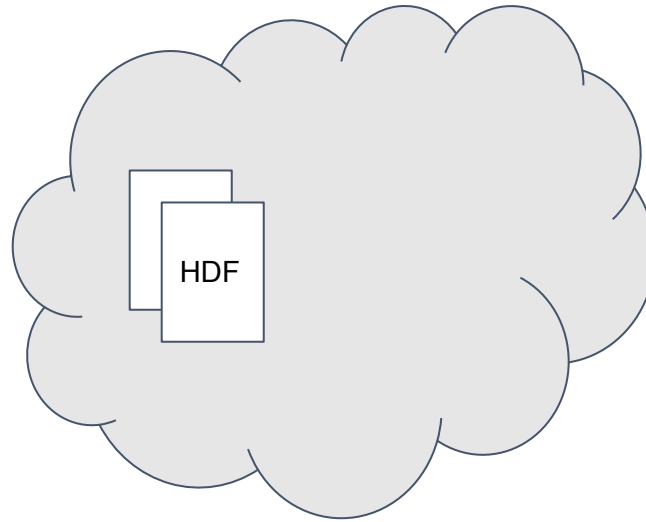
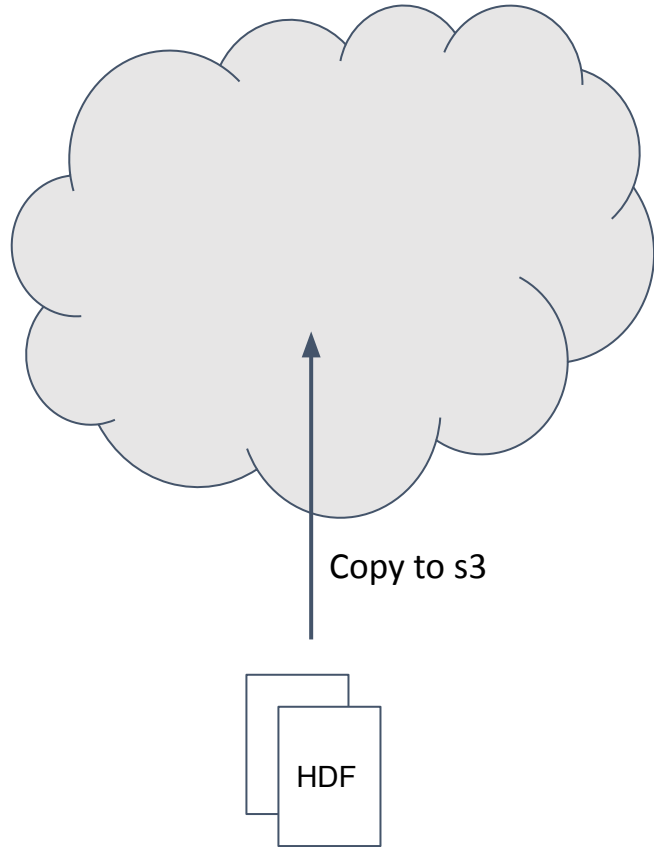
Christine Smit

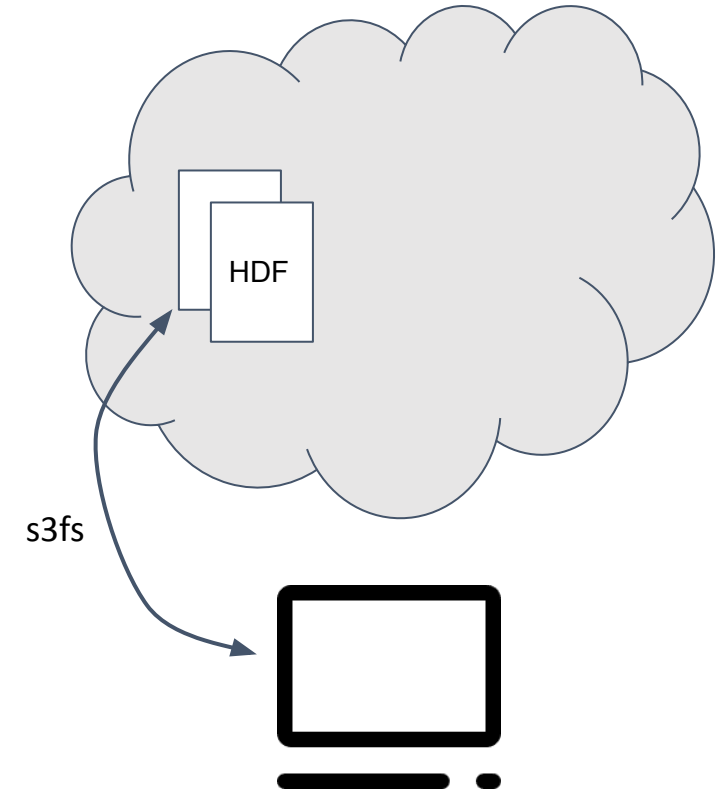
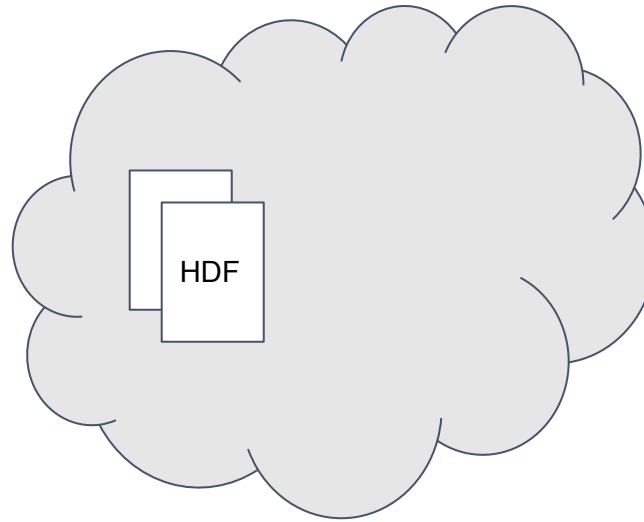
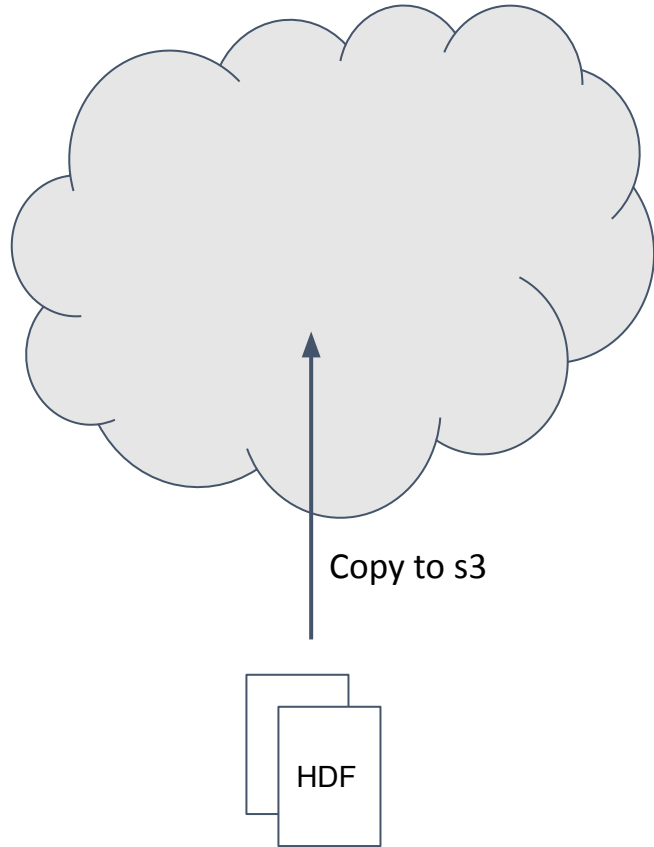
NASA GES DISC / Telophase

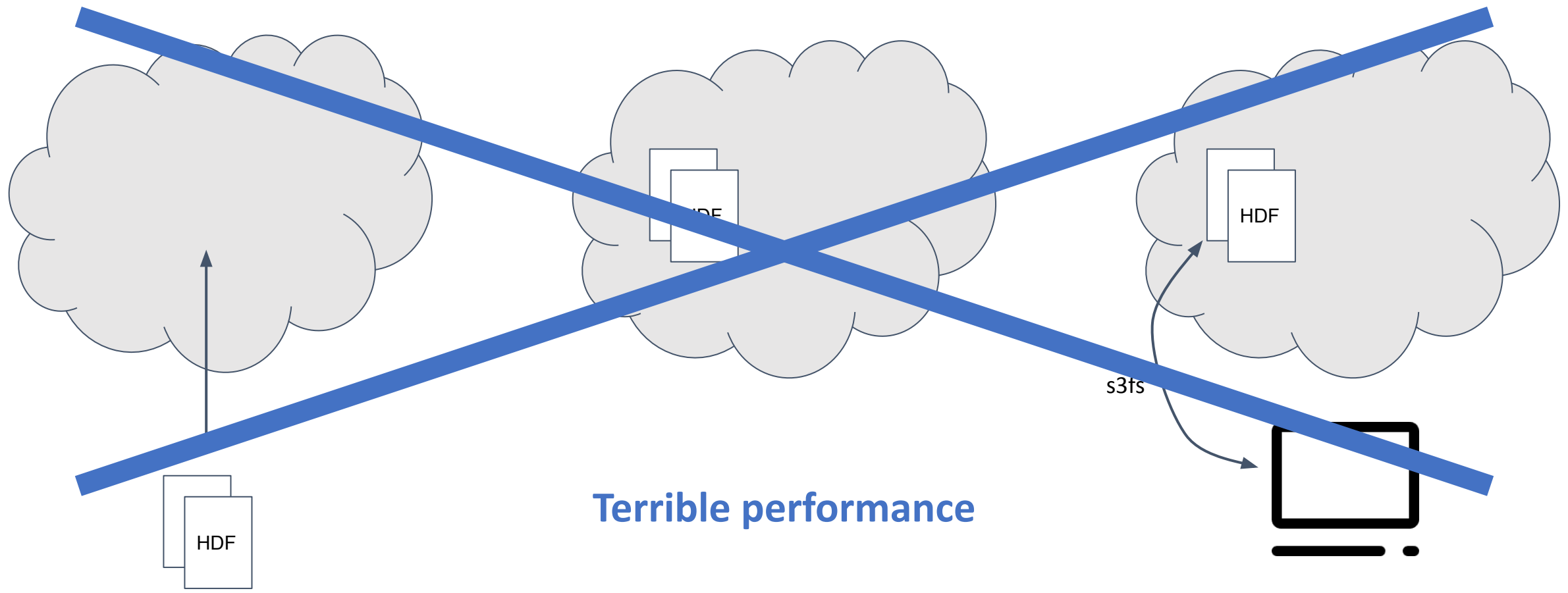


But why Zarr?



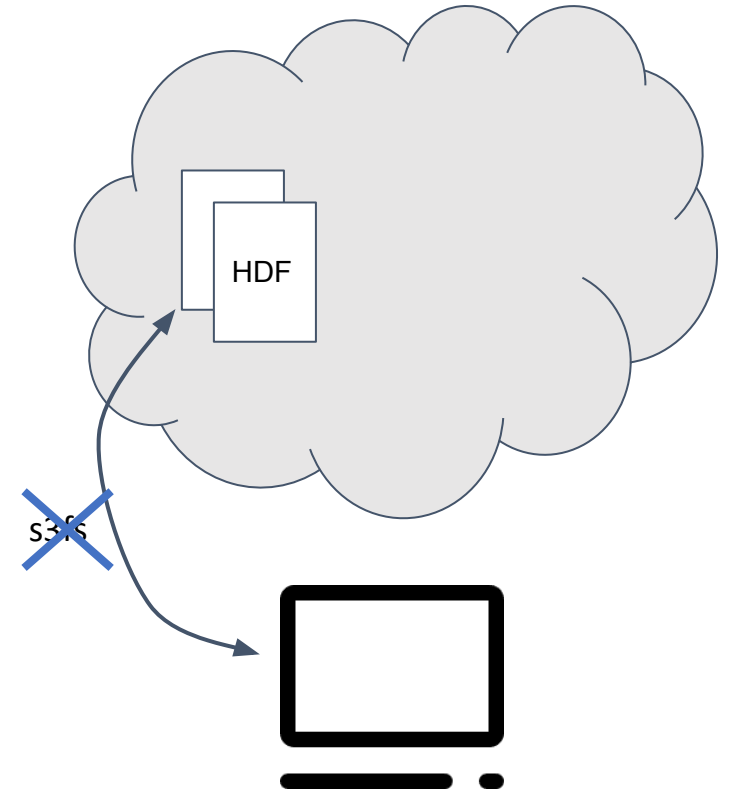
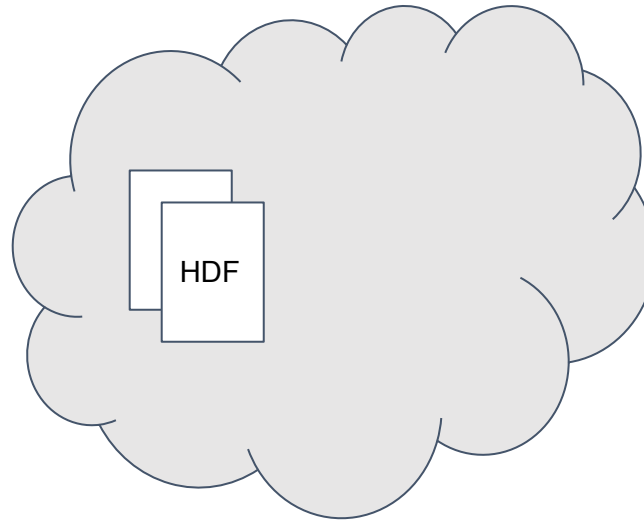
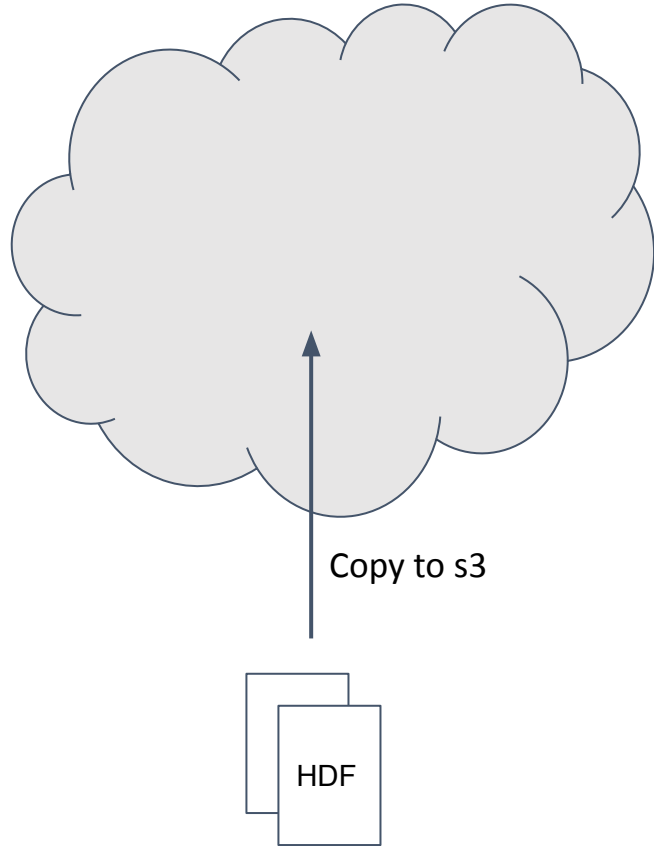






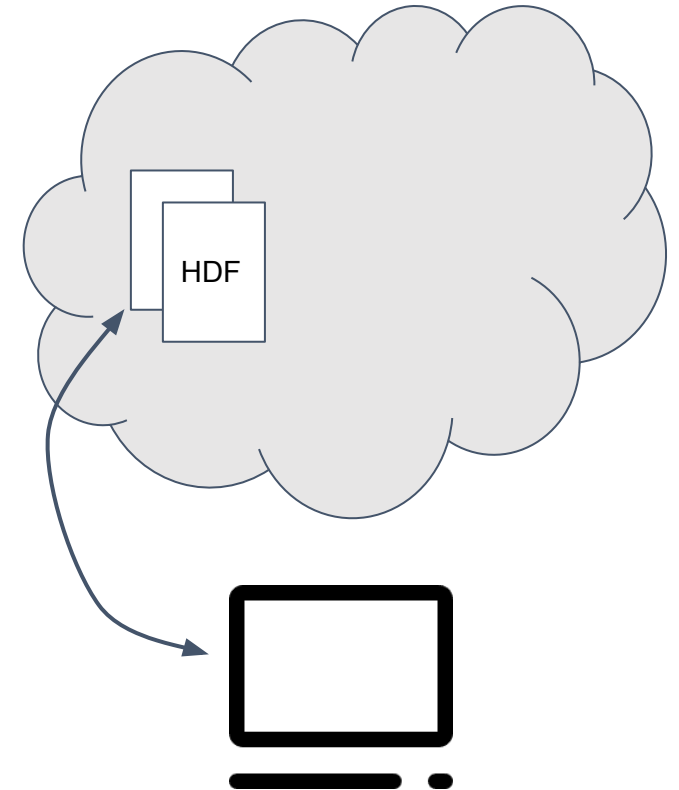
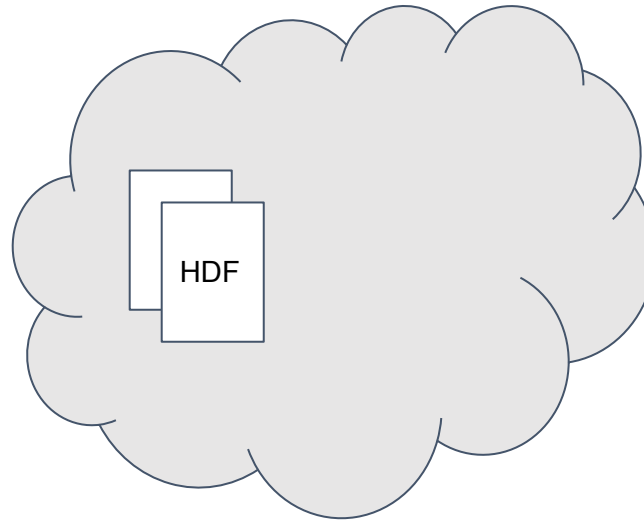
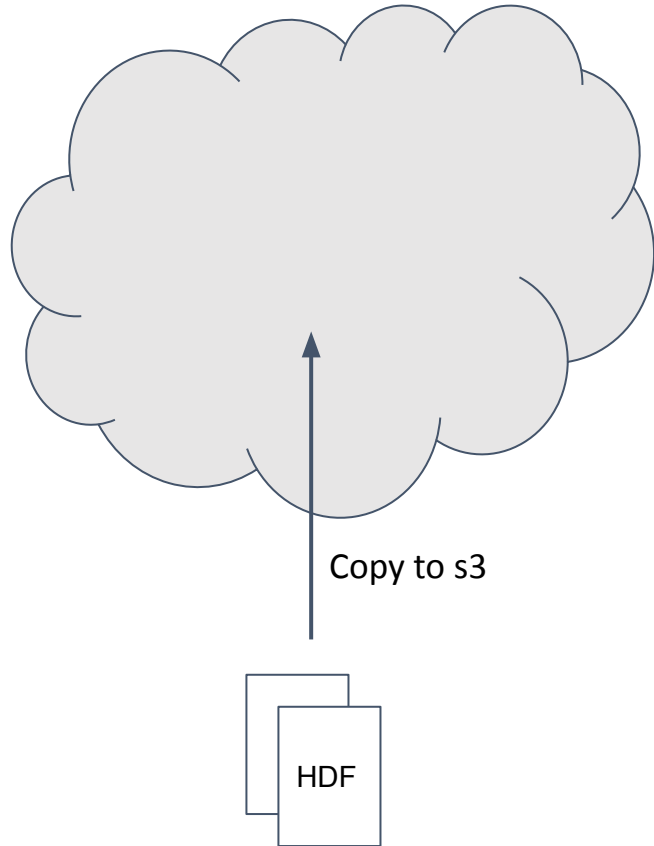


s3fs is the problem!



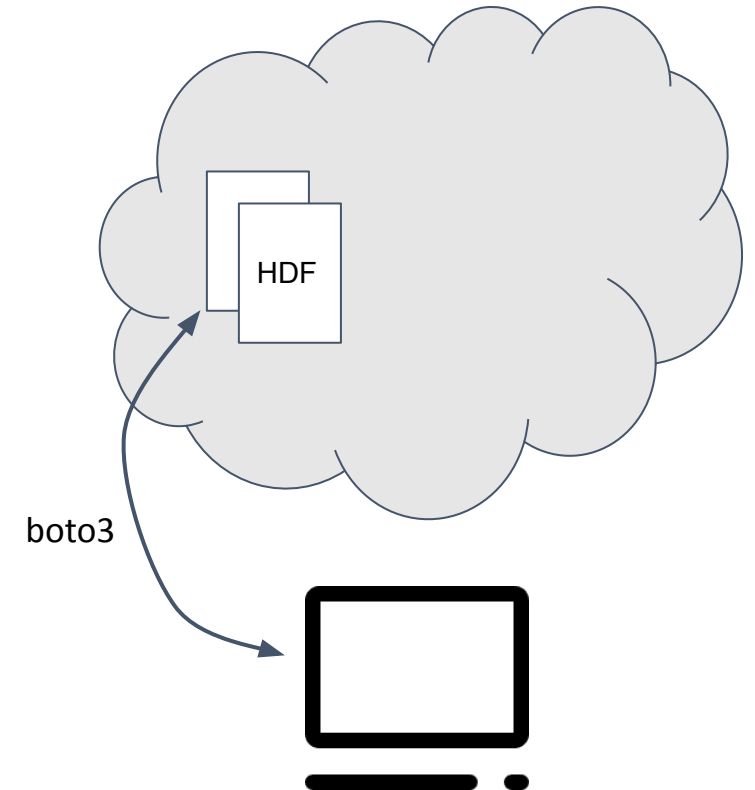
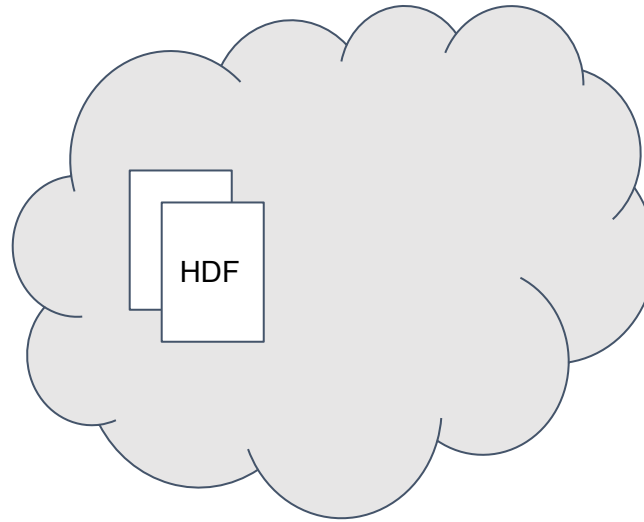
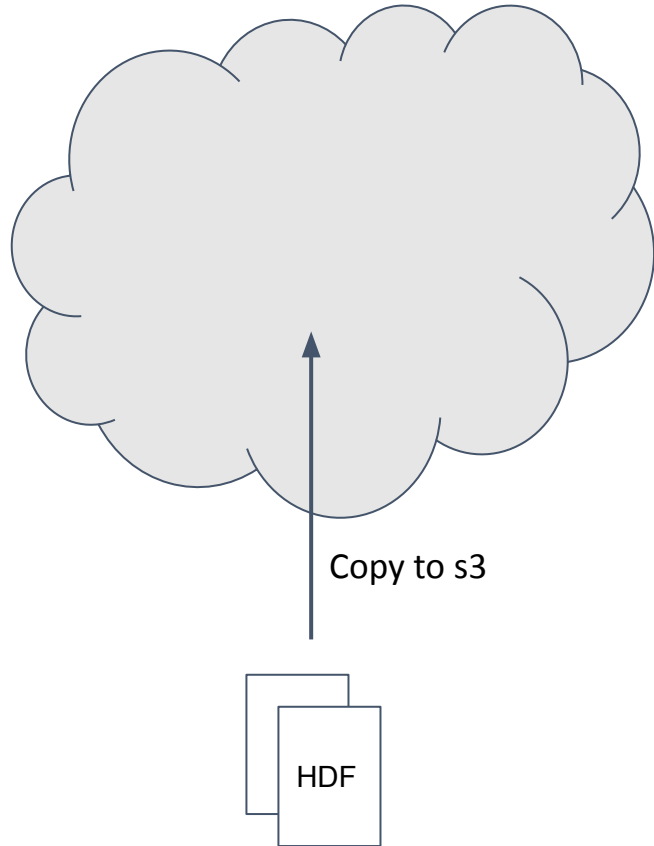


Drop s3fs, use boto3



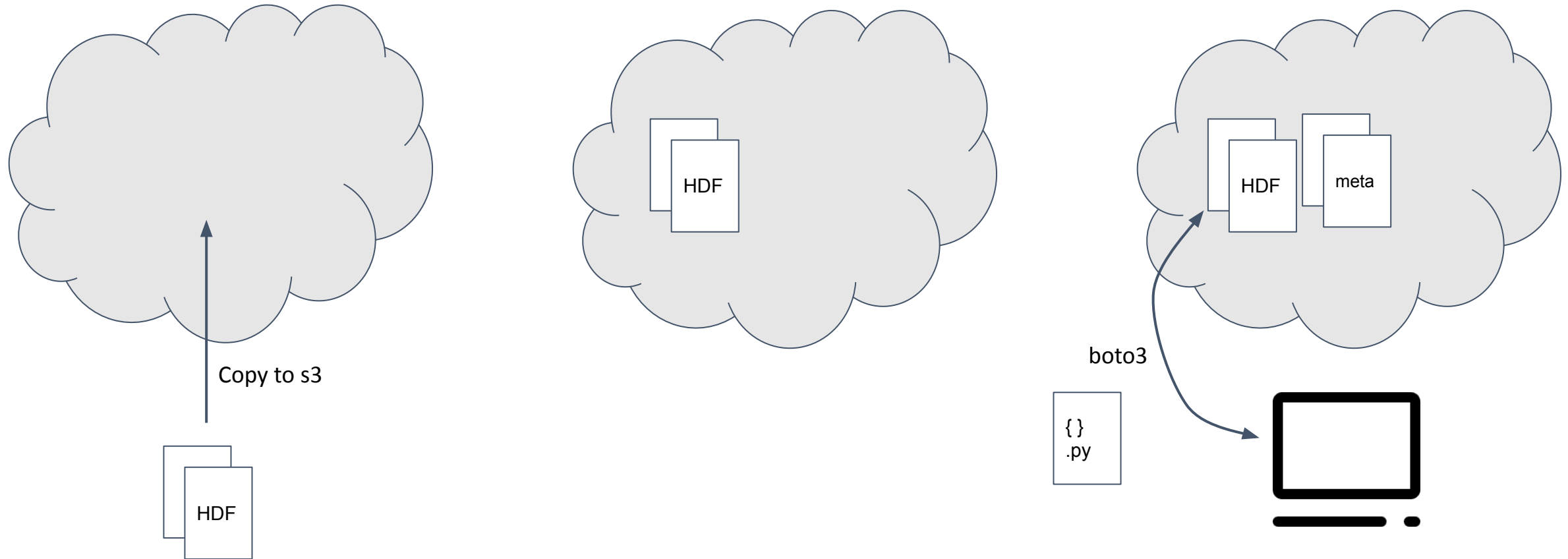


Drop s3fs, use boto3



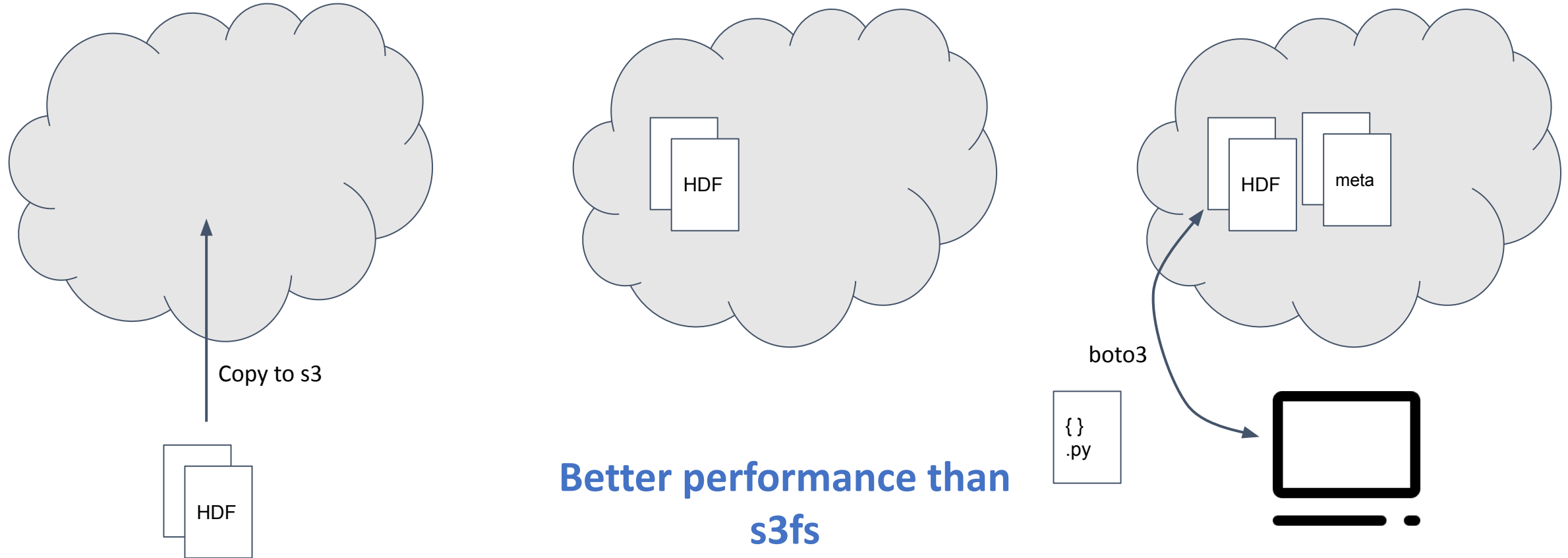


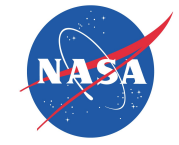
Drop s3fs, use boto3, write custom code, add separate metadata



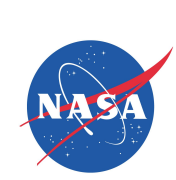


Drop s3fs, use boto3, write custom code, add separate metadata

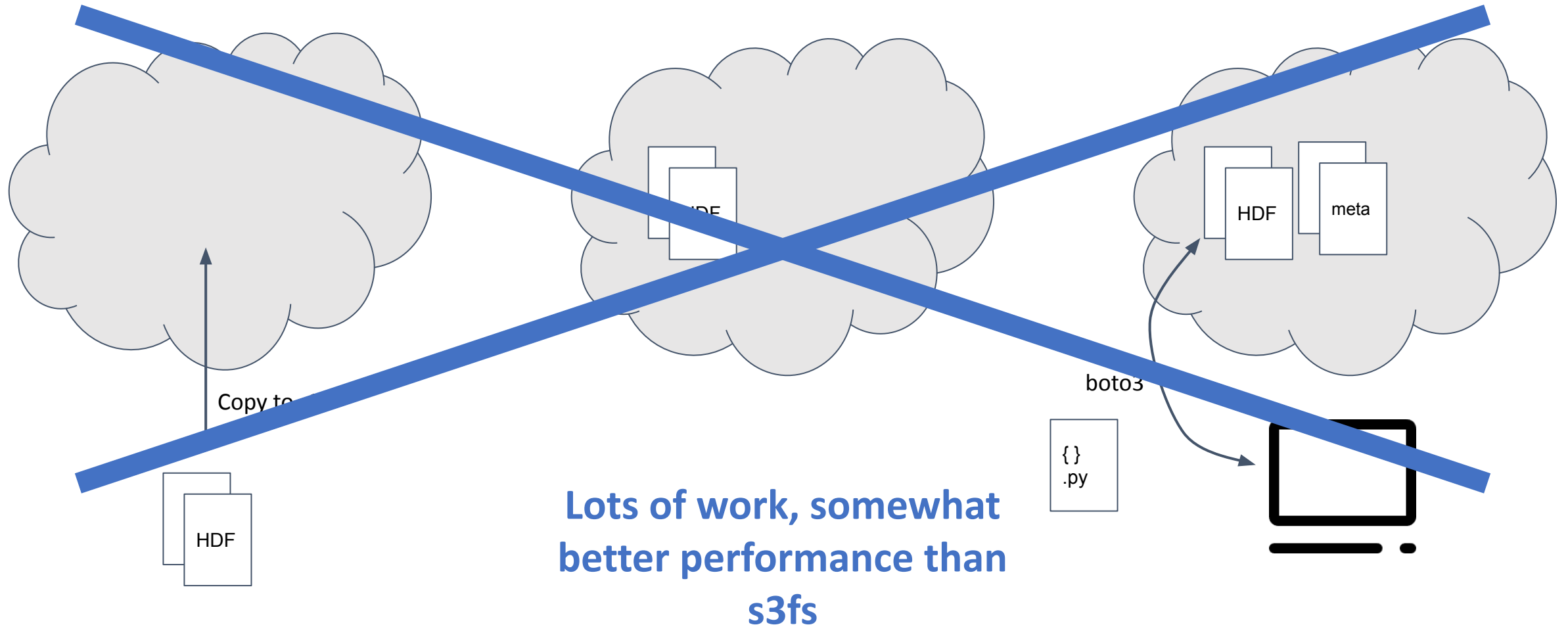


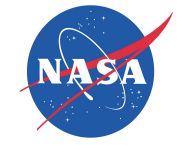


... but you're still fighting to use
formats designed for disks, not
s3

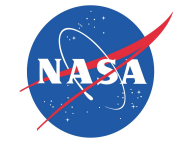


Drop s3fs, use boto3, write custom code, add separate metadata

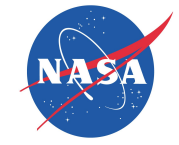




??



data model vs. storage format



data model vs. storage format

- inspired by the HDF data model
- stores multidimensional arrays organized in groups with metadata
- designed from the ground up to work well with object storage



Related technologies

xarray : added metadata to add NetCDF data model concepts

Unidata : adding (work in progress) zarr support in other languages

- NetCDF-C
- NetCDF-Java



OGC Draft standard: Zarr Storage Specification 2.0 Community Standard

2022-04-15: <https://portal.ogc.org/files/100727>

- Includes data type, array size, chunking, etc.
- Includes `_ARRAY_DIMENSIONS`
- Includes Coordinate Reference Systems (CRS)
 - CF-1, section 5.6 : longitude and latitude, `grid_mapping/crs_wkt`
 - GDAL: `_CRS/url/wkt/projjson`

"The forthcoming GeoZarr Spec should seek to reconcile these discrepancies in CRS encoding in Zarr."



So what does this metadata look like?

GLDAS_CLSM025_D_2_0_Rainf_f_tavg.zarr/Rainf_f_tavg/.zarray:

```
{
  "chunks": [
    180,
    720,
    1
  ],
  "compressor": null,
  "dtype": "<f4",
  "fill_value": -9999.0,
  "filters": null,
  "order": "C",
  "shape": [
    720,
    1440,
    2
  ],
  "zarr_format": 2
}
```



So what does this metadata look like?

GLDAS_CLSM025_D_2_0_Rainf_f_tavg.zarr/Rainf_f_tavg/.zattrs:

```
{
  "_ARRAY_DIMENSIONS": [
    "time",
    "latitude",
    "longitude"
  ],
  "_FillValue": -9999.0,
  "coordinates": "time latitude longitude",
  "long_name": "Total precipitation rate",
  "product_short_name": "GLDAS_CLSM025_D",
  "product_version": "2.0",
  "standard_name": "precipitation_flux",
  "units": "kg m-2 s-1",
  "variable_short_name": "Rainf_f_tavg"
}
```



```
$ ncdump -h "file:///path/to/GLDAS_CLSM025_D_2_0_Rainf_f_tavg.zarr#mode=nczarr,zarr"
```

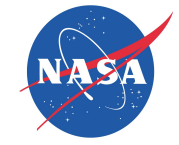
```
netcdf GLDAS_CLSM025_D_2_0_Rainf_f_tavg {
  dimensions:
    time = 2 ;
    latitude = 720 ;
    longitude = 1440 ;
    nv = 2 ;
  variables:
    float Rainf_f_tavg(time, latitude, longitude) ;
      Rainf_f_tavg:_FillValue = -9999. ;
      Rainf_f_tavg:coordinates = "time latitude longitude" ;
      Rainf_f_tavg:long_name = "Total precipitation rate" ;
      Rainf_f_tavg:product_short_name = "GLDAS_CLSM025_D" ;
      Rainf_f_tavg:product_version = "2.0" ;
      Rainf_f_tavg:standard_name = "precipitation_flux" ;
      Rainf_f_tavg:units = "kg m-2 s-1" ;
      Rainf_f_tavg:variable_short_name = "Rainf_f_tavg" ;
    float latitude(latitude) ;
      latitude:bounds = "latitude_bnds" ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
    float latitude_bounds(latitude, nv) ;
      latitude_bounds:units = "degrees_north" ;
    float longitude(longitude) ;
      longitude:bounds = "longitude_bnds" ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
    float longitude_bounds(longitude, nv) ;
      longitude_bounds:units = "degrees_east" ;
    int time(time) ;
      time:bounds = "time_bnds" ;
      time:calendar = "proleptic_gregorian" ;
      time:long_name = "time" ;
      time:standard_name = "time" ;
      time:units = "seconds since 1970-01-01 00:00:00" ;
    int time_bounds(time, nv) ;
      time_bounds:calendar = "proleptic_gregorian" ;
      time_bounds:units = "seconds since 1970-01-01 00:00:00" ;
}
```



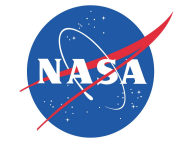
```
$ ncdump -h "file:///path/to/GLDAS_CLSM025_D_2_0_Rainf_f_tavg.zarr#mode=nczarr,zarr"
```

```
netcdf GLDAS_CLSM025_D_2_0_Rainf_f_tavg {
  dimensions:
    time = 2 ;
    latitude = 720 ;
    longitude = 1440 ;
    nv = 2 ;
  variables:
    float Rainf_f_tavg(time, latitude, longitude) ;
      Rainf_f_tavg:_FillValue = -9999. ;
      Rainf_f_tavg:coordinates = "time latitude longitude" ;
      Rainf_f_tavg:long_name = "Total precipitation rate" ;
      Rainf_f_tavg:product_short_name = "GLDAS_CLSM025_D" ;
      Rainf_f_tavg:product_version = "2.0" ;
      Rainf_f_tavg:standard_name = "precipitation_flux" ;
      Rainf_f_tavg:units = "kg m-2 s-1" ;
      Rainf_f_tavg:variable_short_name = "Rainf_f_tavg" ;
    float latitude(latitude) ;
      latitude:bounds = "latitude_bnds" ;
      latitude:standard_name = "latitude" ;
      latitude:units = "degrees_north" ;
    float latitude_bnds(latitude, nv) ;
      latitude_bnds:units = "degrees_north" ;
    float longitude(longitude) ;
      longitude:bounds = "longitude_bnds" ;
      longitude:standard_name = "longitude" ;
      longitude:units = "degrees_east" ;
    float longitude_bnds(longitude, nv) ;
      longitude_bnds:units = "degrees_east" ;
    int time(time) ;
      time:bounds = "time_bnds" ;
      time:calendar = "proleptic_gregorian" ;
      time:long_name = "time" ;
      time:standard_name = "time" ;
      time:units = "seconds since 1970-01-01 00:00:00" ;
    int time_bnds(time, nv) ;
      time_bnds:calendar = "proleptic_gregorian" ;
      time_bnds:units = "seconds since 1970-01-01 00:00:00" ;
}
```

data model >>
storage format



Questions?



Extra Slides



Using python-netCDF4 with zarr

```
In [1]: import os
```

```
In [2]: from netCDF4 import Dataset
```

```
In [3]: destination_dir = os.path.abspath('./GLDAS_CLSM025_D_2_0_Rainf_f_tavg.zarr')
```

```
In [4]: Dataset(f"file://{destination_dir}#mode=nczarr,zarr")
```

```
Out[4]:
```

```
<class 'netCDF4._netCDF4.Dataset'>
```

```
root group (NETCDF4 data model, file format None):
```

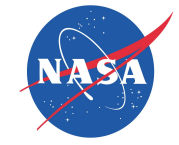
```
    dimensions(sizes): time(2), latitude(720), longitude(1440), nv(2)
```

```
    variables(dimensions): float32 Rainf_f_tavg(time, latitude, longitude), float32
```

```
latitude(latitude), float32 latitude_bounds(latitude, nv), float32 longitude(longitude),
```

```
float32 longitude_bounds(longitude, nv), int32 time(time), int32 time_bounds(time, nv)
```

```
    groups:
```



NetCDF-C / NetCDF-Java as foundation for additional zarr adoption

- nco
- Panoply
- ...



Familiar HDF concepts in metadata

GLDAS_CLSM025_D_2_0_Rainf_f_tavg.zarr/Rainf_f_tavg/.zarray:

```
{
  "chunks": [
    180,
    720,
    1
  ],
  "compressor": null,
  "dtype": "<f4",
  "fill_value": -9999.0,
  "filters": null,
  "order": "C",
  "shape": [
    720,
    1440,
    2
  ],
  "zarr_format": 2
}
```

← chunking

← compression

← data type

← array size