DANRA (Harmonie output) storage and dissemination with zarr

Leif Denby, Weather Research Department, DMI

15/4/2024, ACCORD ASW

What is DANRA?

"The goal is to create a 70-year atmospheric reanalysis for Danish area with the Harmonie-2.5 km Numericical Weather Prediction model. A near term goal is to have a 30+ year reanalysis for 1990-2020 by the end of 2023."

https://dmidk.github.io/danra-docs/intro.html

How is DANRA stored?

```
/dmidata/projects/nckf/danra/archive
├── 1990
│   ├── 01
│   │   ├── 01
│   │   │   ├── [1.0G]  1990_01_01_ANALYSIS.tgz
│   │   │   ├── [817M]  1990_01_01_FC33hr.tgz
│   │   │   └── [3.7G]  1990_01_01_FC3hr.tgz
:   :   :
│   │   └── 31
│   │       ├── [1.0G]  1990_01_31_ANALYSIS.tgz
│   │       ├── [819M]  1990_01_31_FC33hr.tgz
│   │       └── [3.7G]  1990_01_31_FC3hr.tgz
│   └── 12
│       ├── 01
:       :
│       └── 31
│           ├── [1.0G]  1990_12_31_ANALYSIS.tgz
│           ├── [819M]  1990_12_31_FC33hr.tgz
│           └── [3.7G]  1990_12_31_FC3hr.tgz
├── 1991
│   ├── 01
│   │   ├── 01
:   :   :   :
:
└── 2020
    ├── 01
    :   :
    │
    └── 12
        ├── 01
        :   :
        :
        └── 31
            ├── [1.0G]  2020_12_31_ANALYSIS.tgz
            ├── [819M]  2020_12_31_FC33hr.tgz
            └── [3.7G]  2020_12_31_FC3hr.tgz
```

```
lcd@8fr95q2:/dmidata/projects/nckf/danra$ tar tvf archive/2020/12/31/2020_12_31_ANALYSIS.tgz
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_1_103_0_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_1_105_0_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_1000_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_100_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_200_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_250_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_300_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_400_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_500_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_600_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_700_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_800_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_850_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_900_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_925_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_950_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_0_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_100_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_150_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_200_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_2_0_ANALYSIS
```

- indidual GRIB files per variable, level-type, level
- packed in .tgz-file per data-kind (forecast, analysis) and date
- stored in directory structure, per day, month, year

What is the problem?

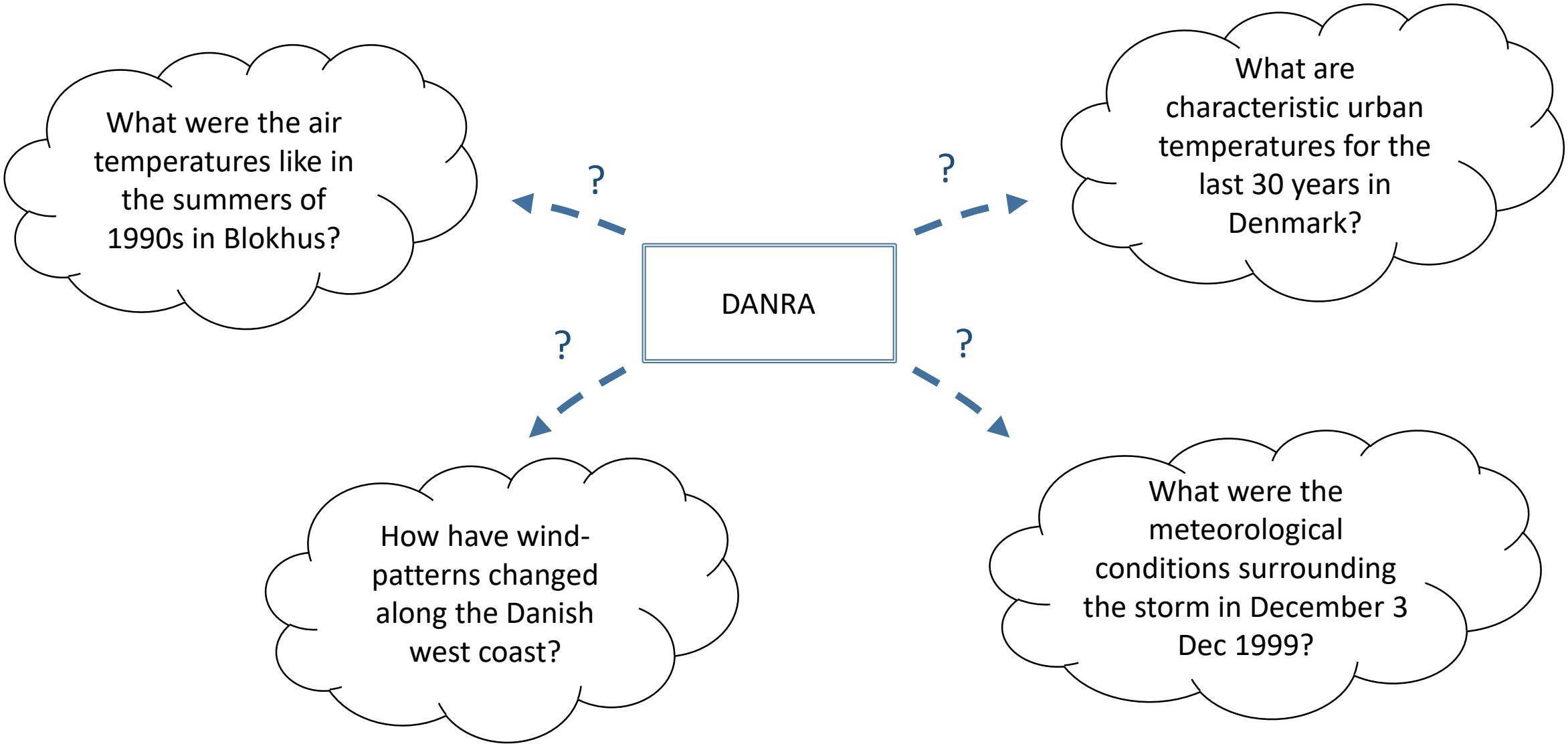
What were the air temperatures like in the summers of 1990s in Blokhus?

What are characteristic urban temperatures for the last 30 years in Denmark?

How have wind-patterns changed along the Danish west coast?

What were the meteorological conditions surrounding the storm in December 3 Dec 1999?

What is the problem?



How is DANRA (also) stored now?

```
lcd@8fr95q2:/dmidata/projects/cloudphysics/danra/data/v0.4.0$ du -h -d 1
568G    ./pressure_levels.zarr
342G    ./height_levels.zarr
1.9T    ./single_levels.zarr
2.8T    .
lcd@8fr95q2:/dmidata/projects/cloudphysics/danra/data/v0.4.0$
```

all variables* and levels** across entire

30-years in 3 three .zarr datasets

**: only prognostic variables for now, diagnostic variables (for example precipitation, which are derived from forecast files) are to come*

***: only near-surface (100hPa and 100m) levels processed so far*

So what is this “zarr” thing?

- Functionally like netCDF

```python
import xarray as xr
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
```

Python

```python
ds = xr.open_zarr("/dmidata/projects/cloudphysics/danra/data/v0.4.0/single_levels.zarr/")
ds
```

Python

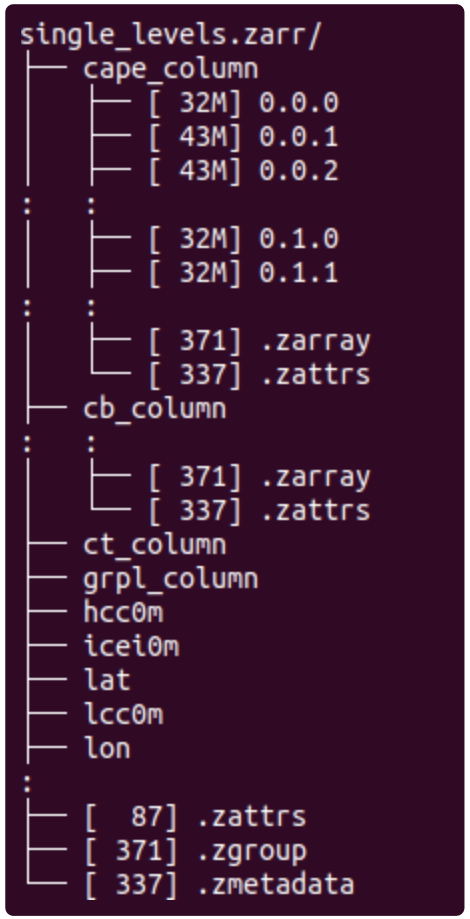
```python
g = ds.sel(time=slice("1999-12-3T00:00", "1999-12-4T00:00")).u10m.plot(
    col="time",
    col_wrap=3,
    x="lon",
    y="lat",
    transform=ccrs.PlateCarree(),
    subplot_kws=dict(projection=ccrs.PlateCarree()),
    aspect=2.0,
    size=2.0,
)

for ax in g.axes.flatten():
    ax.gridlines(draw_labels=["top", "left"])
    ax.coastlines()
```

Python

```python
# plot 2m temperature near Herning
ds.t2m.sel(x=-1.0e6, y=50e3, method="nearest").plot()
```

Python

So what is this “zarr” thing *actually*?



- functionally like netCDF for the end-user
- a specification for how to store multi-dimensional array data, in
  - predefined directory structure (each subdir is a separate variable)
  - data is chunked by dimensions of array (e.g. [time,x,y] for CAPE, but [time,x,y,level] for moisture on pressure levels)
  - meta-information stored separately
- Storing by-variable, chunked and with separate meta-data allows for very access:
  - Only required chunks are fetched and opened
  - No server/API needed (!) the specfication is the API

How do convert GRIB to zarr?

```
lcd@8fr95q2:/dmidata/projects/nckf/danra$ tar tvf archive/2020/12/31/2020_12_31_ANALYSIS.tgz
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_1_103_0_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_1_105_0_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_1000_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_100_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_200_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_250_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_300_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_400_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_500_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_600_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_700_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_800_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_850_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_900_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_925_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_100_950_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_0_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_100_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_150_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_200_0_ANALYSIS
-rw-r----- nhd/dk      7440512 2021-04-28 19:39 DKREA_PRODYEAR_2020_12_31_11_105_2_0_ANALYSIS
```

?

dataset.zarr

gribscan + rechunker

How do convert GRIB to zarr?

gribscan + *rechunker*

```
lcd@volta:/dmidata/cache/dini/sf$ tree 2024041500/
2024041500/
├── 000
├── 001
├── 002
├── 003
├── 004
├── 005
:
└── 060
```

Create index file for each GRIB source file

```
gribscan-index 000 001 002 ...
```

```
lcd@volta:/dmidata/grib-indecies/dini/sf$ tree 2024041500
2024041500
├── 000.index.json
├── 001.index.json
├── 002.index.json
├── 003.index.json
├── 004.index.json
├── 005.index.json
:
└── 060.index.json
```

Assemble dataset from index files

```
gribscan-build *.index -o dataset.zarr.json
```

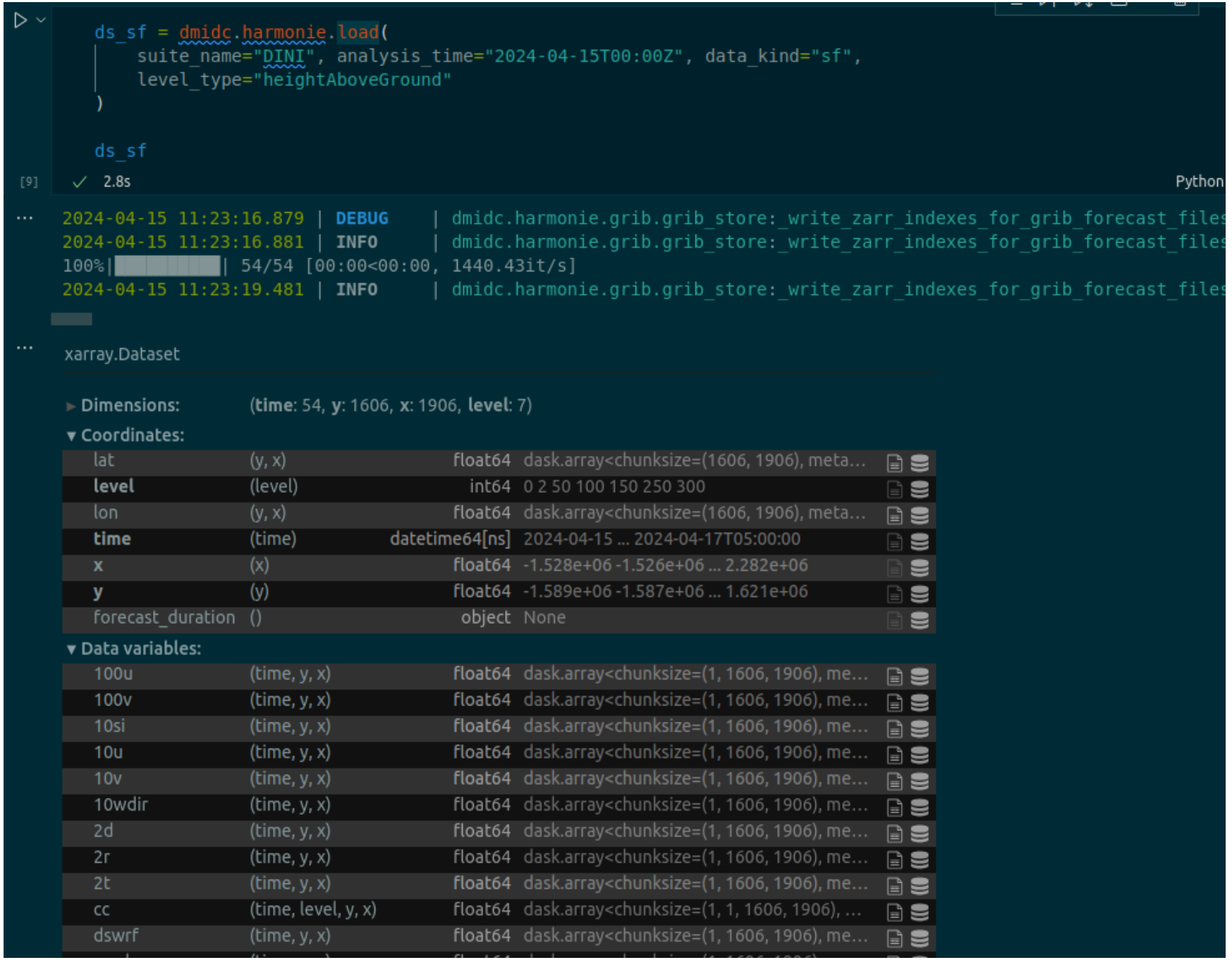
Open dataset

```
analysis_time_20240415T000000+0000__suite_name_DINI__data_kind_sf__level_type_heightAboveGround.zarr.json
```

```
import gribscan
import xarray as xr
ds = xr.open_zarr("reference::dataset.zarr.json", consolidated=False)
ds
```

https://github.com/leifdenby/gribscan/tree/harmonie-rotated-and-lambert-grids

How does this look from a user perspective?



How do convert GRIB to zarr?

gribscan + *rechunker*

But, what is it with all these json files? I thought you were making a zarr dataset?

How do convert GRIB to zarr?

gribscan + ***rechunker***

But, what is it with all these json files? I thought you were making a zarr dataset?

How do convert GRIB to zarr?

gribscan + ***rechunker***

```
import gribscan
import xarray as xr
ds = xr.open_zarr("reference::dataset.zarr.json", consolidated=False)
ds
```

Rechunker is a Python package which enables efficient and scalable manipulation of the chunk structure of chunked array formats such as Zarr and TileDB. Rechunker takes an input array (or group of arrays) stored in a persistent storage device (such as a filesystem or a cloud storage bucket) and writes out an array (or group of arrays) with the same data, but different chunking scheme, to a new location. Rechunker is designed to be used within a parallel execution framework such as Dask.

*Create **actual** .zarr dataset using rechunker, setting chunk-sizes to match access pattern*

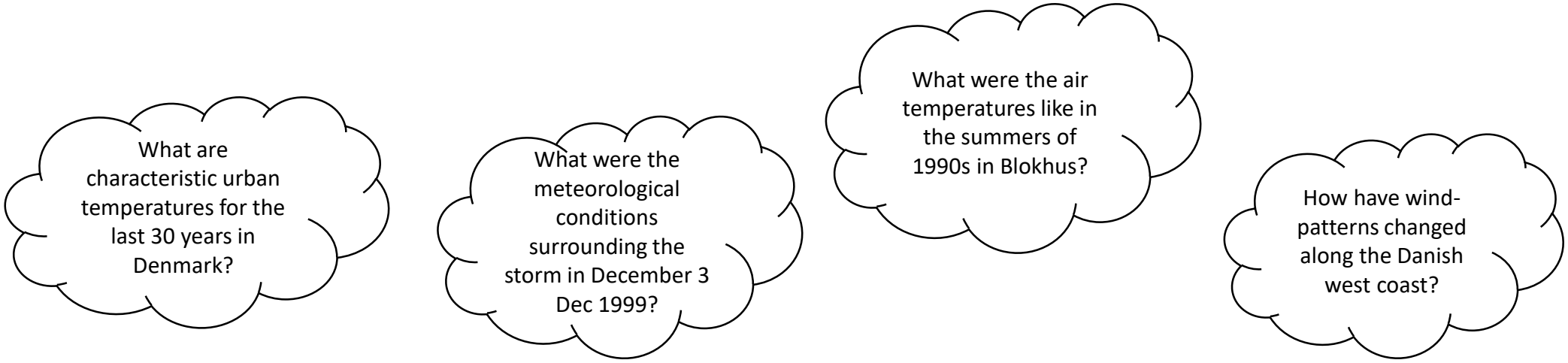
```
from rechunker import rechunk

target_chunks = dict(time=24)
max_mem = "2GB"

target_store = "height_levels.zarr"
temp_store = "/tmp/danratemp.zarr"

rechunk_plan = rechunk(
    ds, target_chunks, max_mem, target_store, temp_store=temp_store
)
rechunk_plan.execute()
```

https://github.com/pangeo-data/rechunker

Why am I doing this?

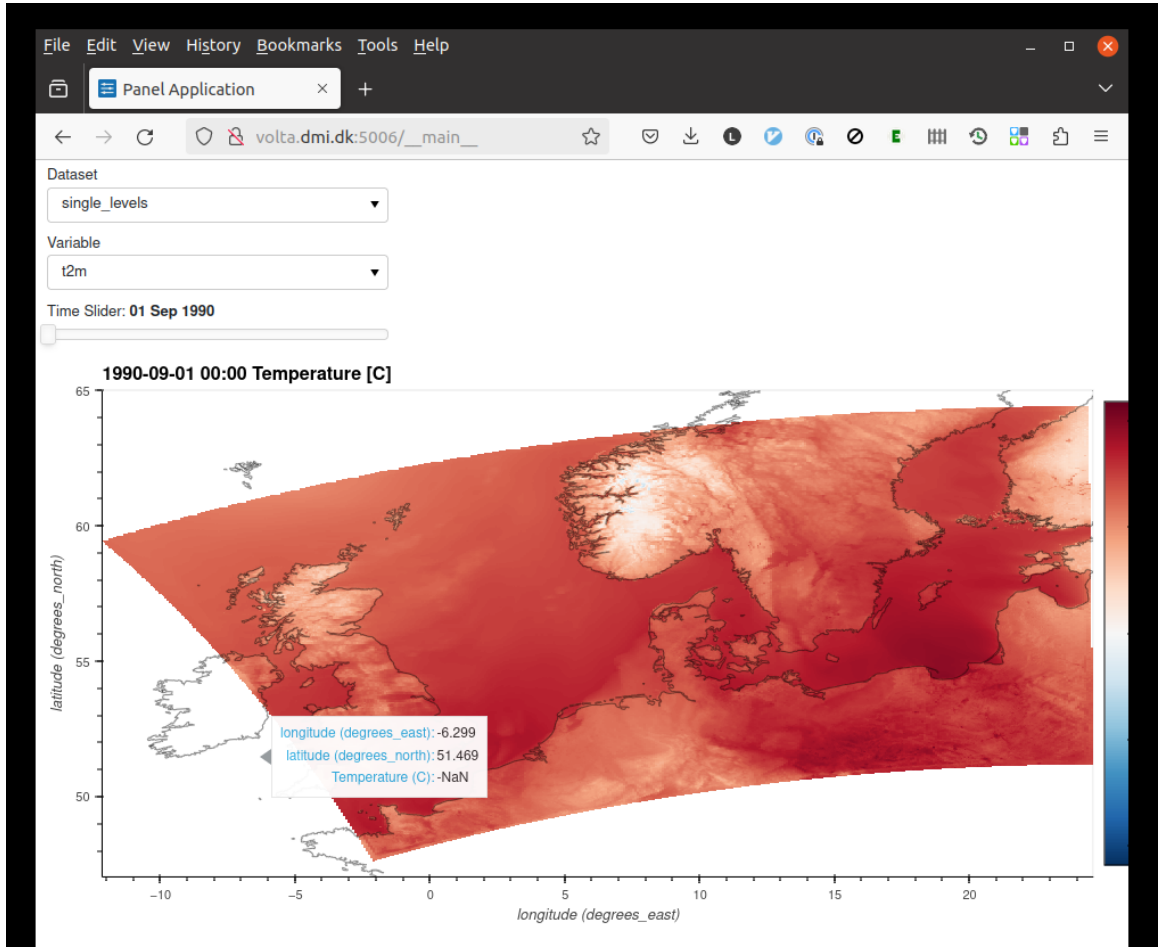
Converting DANRA to zarr makes it easy to

- construct machine learning training datasets from it (and encourage ML LAM community to convert their datasets to zarr)
- allow other downstream applications to easily work with DANRA



~~Demo time~~

interactive visualiser reading directly from zarr stored on scale



“running” on: http://volta.dmi.dk:5006/

code on: https://gitlab.dmi.dk/lcd/danra-viz

What's next?

European Weather Cloud or
Amazon S3 object-store

Accesses entire dataset directly fetching only needed parts without latency

Happy end-user / researcher

- DANRA in zarr copied to object-store (European Weather cloud or Amazon S3)
- Create Jupyter notebooks (ideally Jupyter book) to showcase how to a) access and b) use DANRA to stored in zarr
- End-users can read any part of DANRA they want directly from object store (!) with no request latency

code repo: https://gitlab.dmi.dk/nwp/danra/danra_to_zarr

What's next for zarr at DMI?

- We could (operationally) convert DINI and other suites to zarr
  - Or at least automatically create the gribscan-indexes to allow reading
- We could convert other raster datasets to zarr to improve ease of access and use (project to convert CARRA already in pipeline)

I am happy to help out!

code repo: https://gitlab.dmi.dk/nwp/danra/danra_to_zarr