# ECMWF point database: providing direct access to any model output grid-point values

Baudouin Raoult, Cihan Şahin, Sylvie Lamy-Thépaut
ECMWF

# Why a point database?

- ECMWF main models output
  - a 16 km high resolution global forecast to 10 days (HRES)
  - a 32 km ensemble forecast of 51 members to 15 days (ENS)
  - Two runs: 00Z and 12Z
  - Each run produces around 4 Tbytes of GRIB data (~3.5 million fields)
- It is impossible for our users to transfer the full model output on their computers for further use
  - The ensemble data are not used to their full potential
- We know from our users that there is a strong requirement for point forecasts:
  - Vertical profiles
  - Time series
- Providing direct access to point data will allow our users to develop their own ensemble based products
  - We want them to have access to any parameters/levels/steps produced

# Implementation issues

- The first approach would be read all the GRIB model data, unpack it and feed a database (e.g. Postgres, MySQL…), organised by points (e.g. all the steps/members/levels for one parameter for a given point). This entails:
  - Reading in 4TB from disk
  - Unpack the GRIB data
  - *(Interpolating spherical harmonics fields to grid point)*
  - "Transposing" all the data from ~3.5 millions 2D global gridded fields to ~2 millions point based matrices
  - Writing the result in the database
  - Build the spatial indexes
  - Create two copies (for redundancies), create more copies for test suites
- All that under 10-20 minutes!
  - That cannot work (test shows that hours are needed…)

# Solution explored

- Reading in 4TB from disk (Solution: don’t do it)
  - Access the GRIB directly. There are already indexed by the model in out fields database (FDB)
  - Using more efficient indexes (B-Trees) will speed up access
  - Extend index to contain grid description, and GRIB packing information (data offset, number of bits, scaling factor, reference value…)
- Unpack the GRIB data (Solution: don’t do it)
  - GRIB data values can be extracted directly (simple packing)
- *(Interpolating spherical harmonics fields to grid point)*
  - Have the model output grid point data directly (being studied)
- “Transposing” all the data from ~3.5 millions 2D global gridded fields to ~2 millions point based matrices (Solution: don’t do it)
  - Not needed.

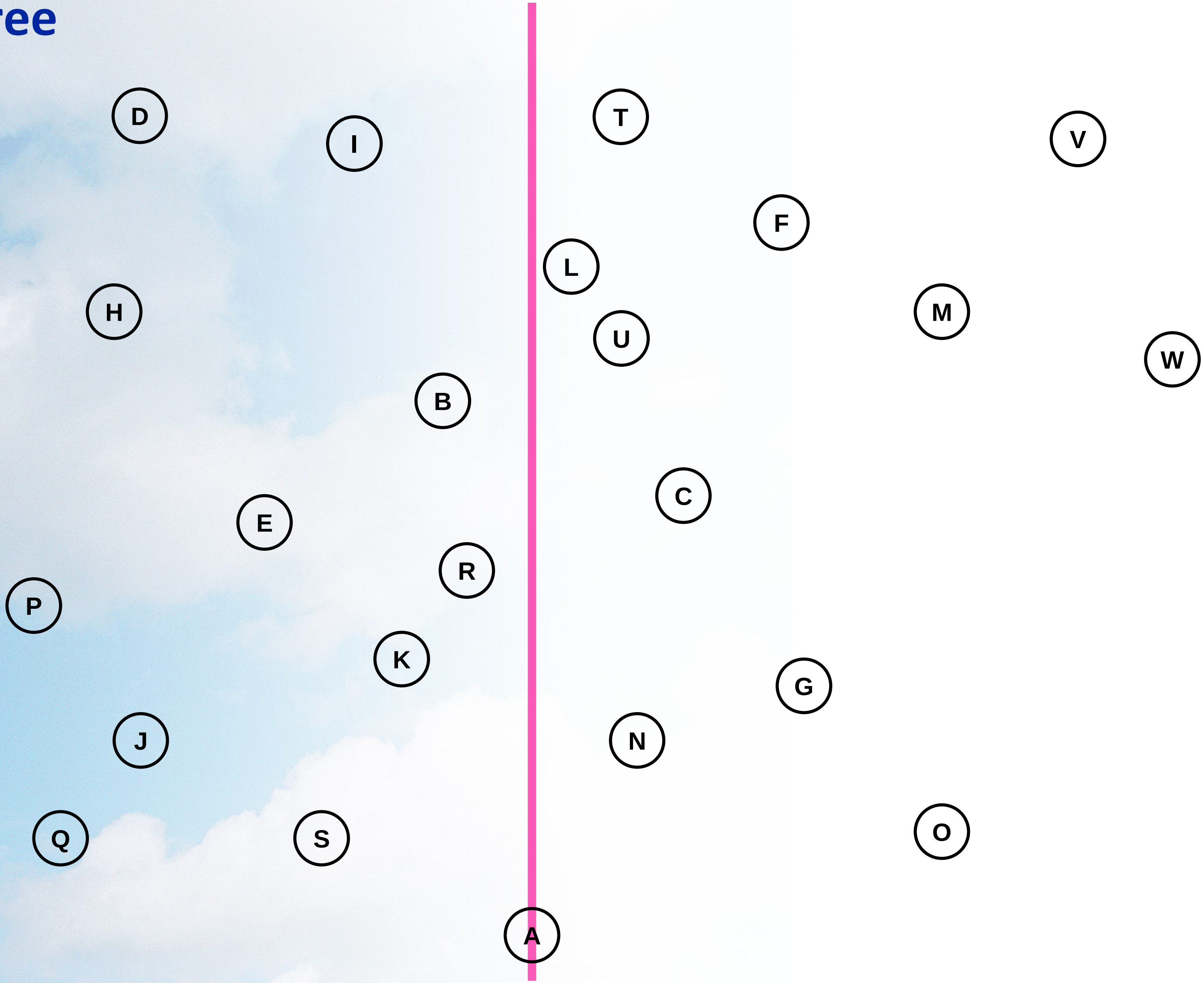
# Solution explored (cont.)

- Writing the result in the database (Solution: Don't do it)
  - Not needed.
- Build the spatial indexes
  - There are many spatial indexes Quadtrees, Octrees, R-trees, kd-trees…
  - We selected kd-tree: simple to implement, support many dimensions.
  - We only need one index per type of grids, i.e. 5: N640, N320, N160 and HRES wave and ENS wave.
- Create two copies (for redundancies), create more copies for test suites (Solution: Don't do it)
  - Not needed: we are already doing this for the FDB.

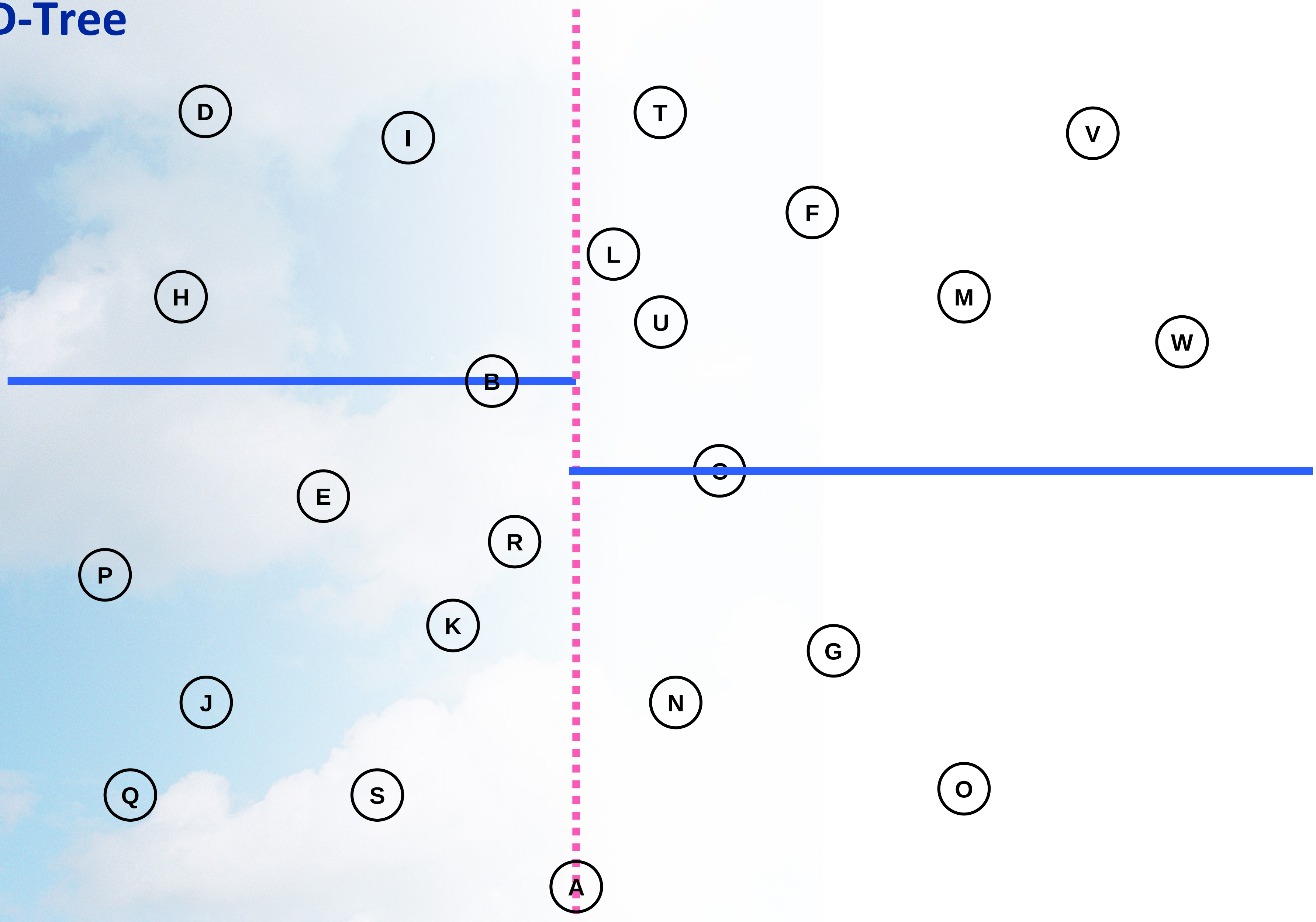
- Solution: do everything on demand from the raw model output, and cache as much as possible.

# KD-Tree

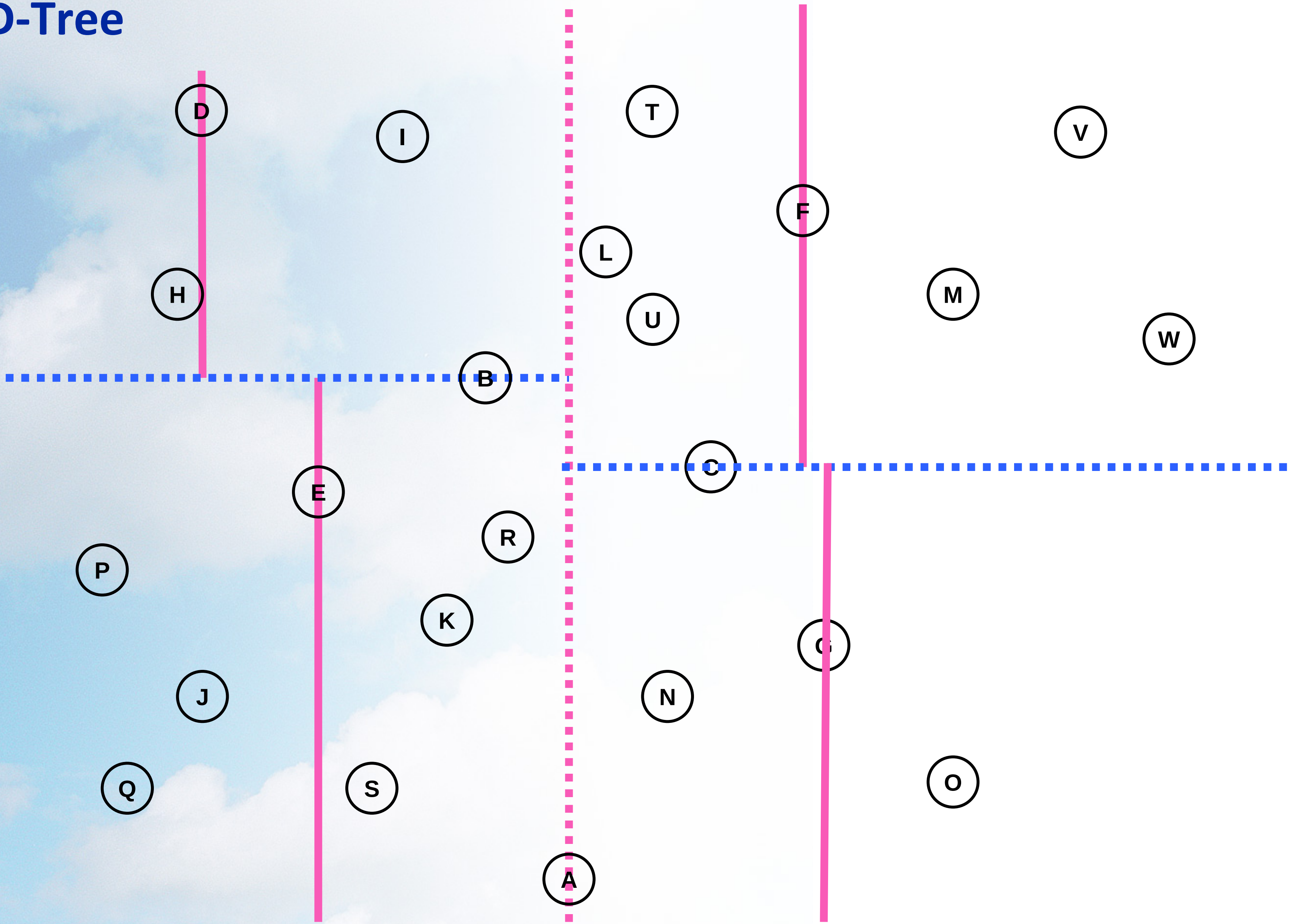
# KD-Tree



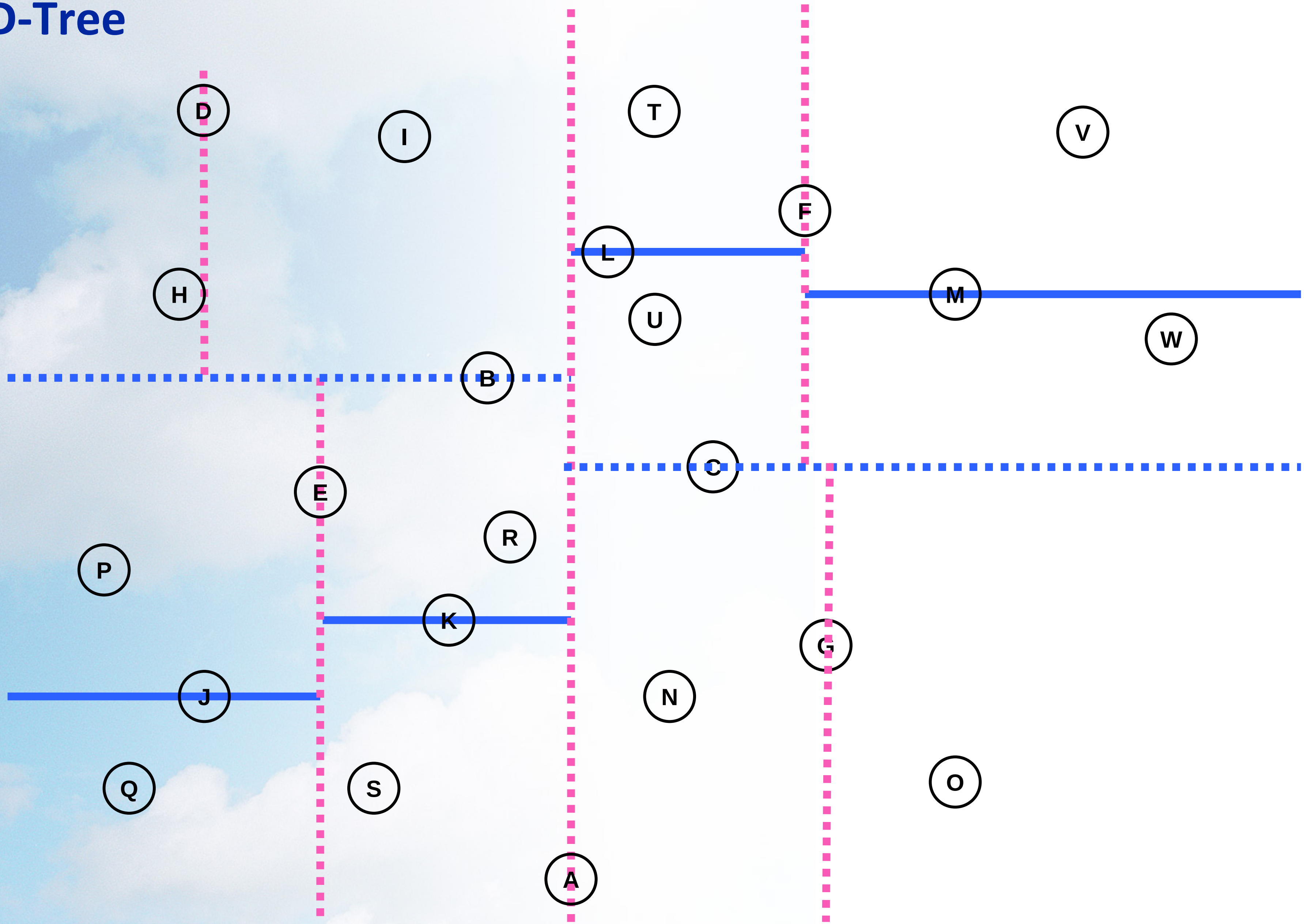
# KD-Tree



# KD-Tree

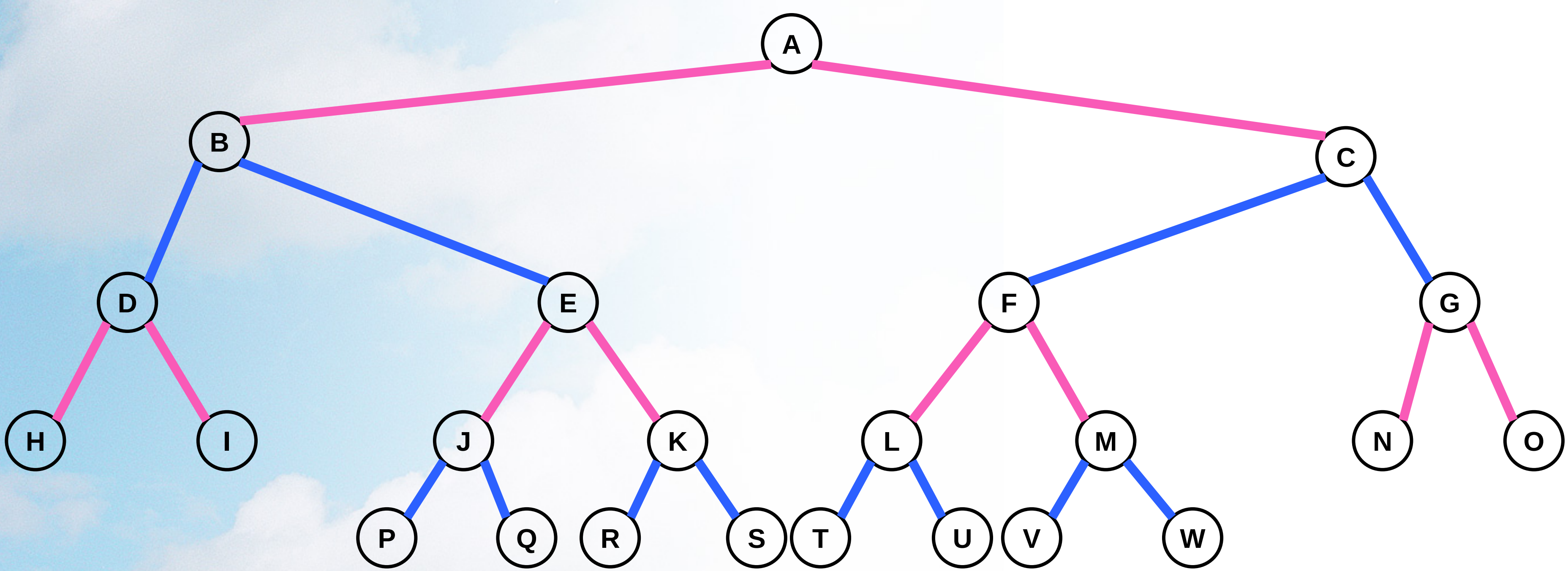


# KD-Tree



# KD-Tree

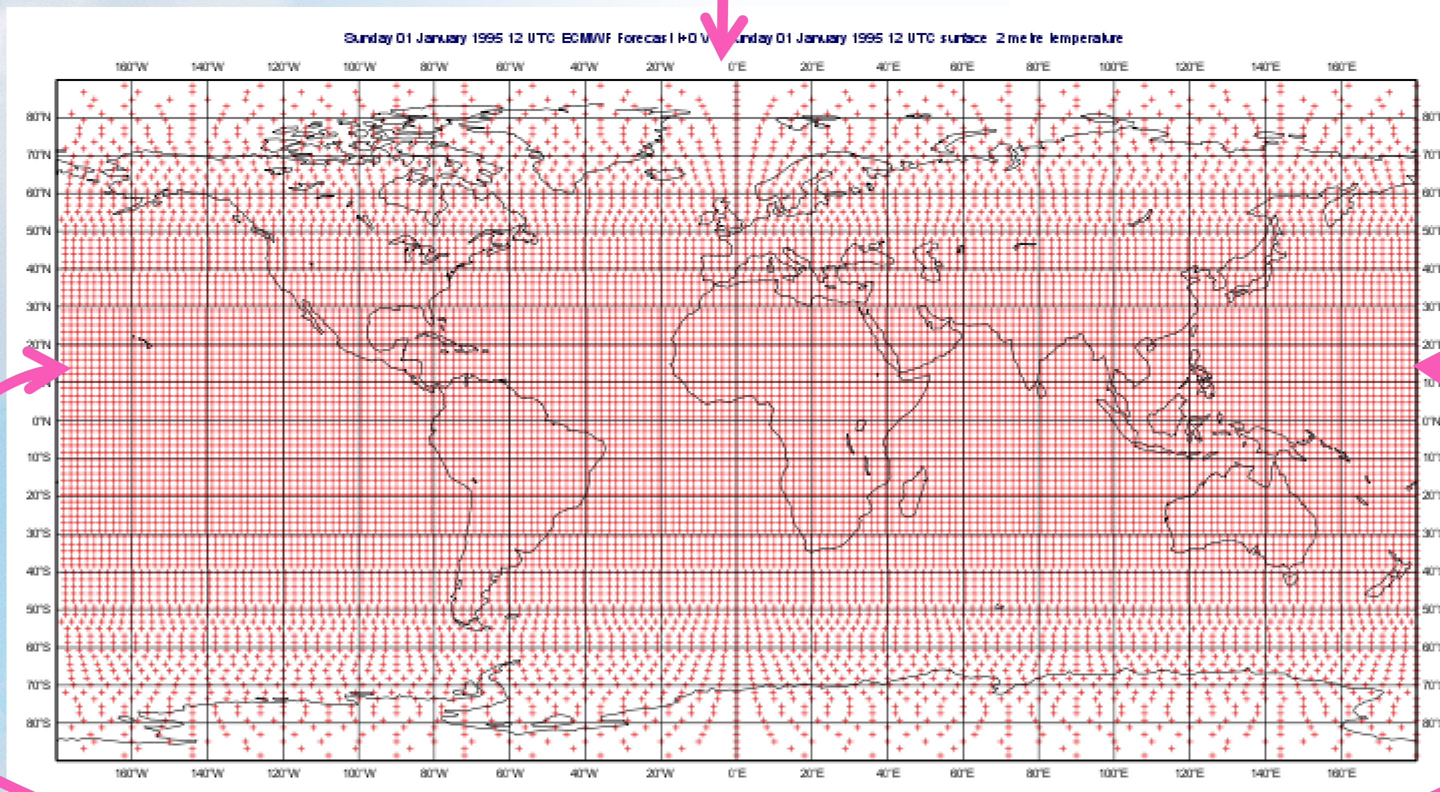
- Finding the nearest point is a matter of visiting the tree
- For a N640 (2,140,702 points) ~ 21 comparisons needed
- Trees only depends of the grid, not the data, so they are cached



# Issues with latitude/longitude coordinate system

- kd-trees only work with an Euclidean metric

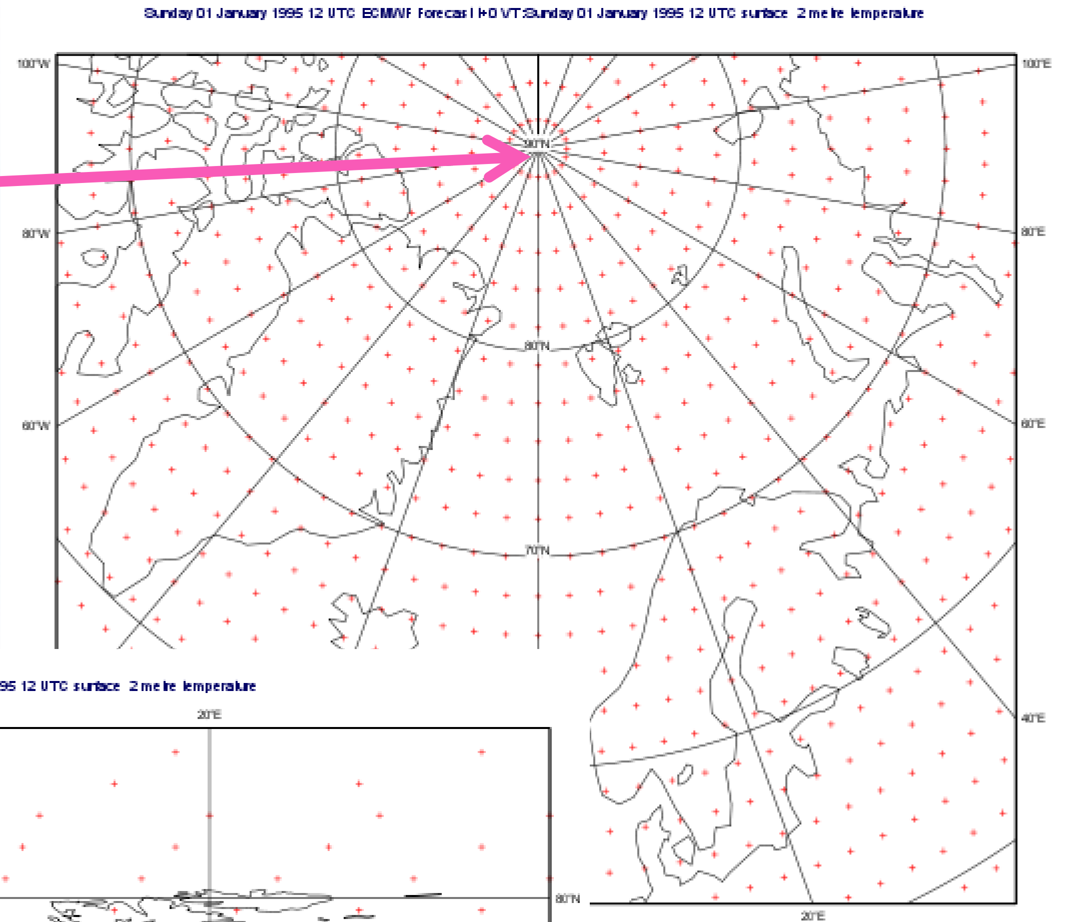
What about the poles?




These points are close to each other

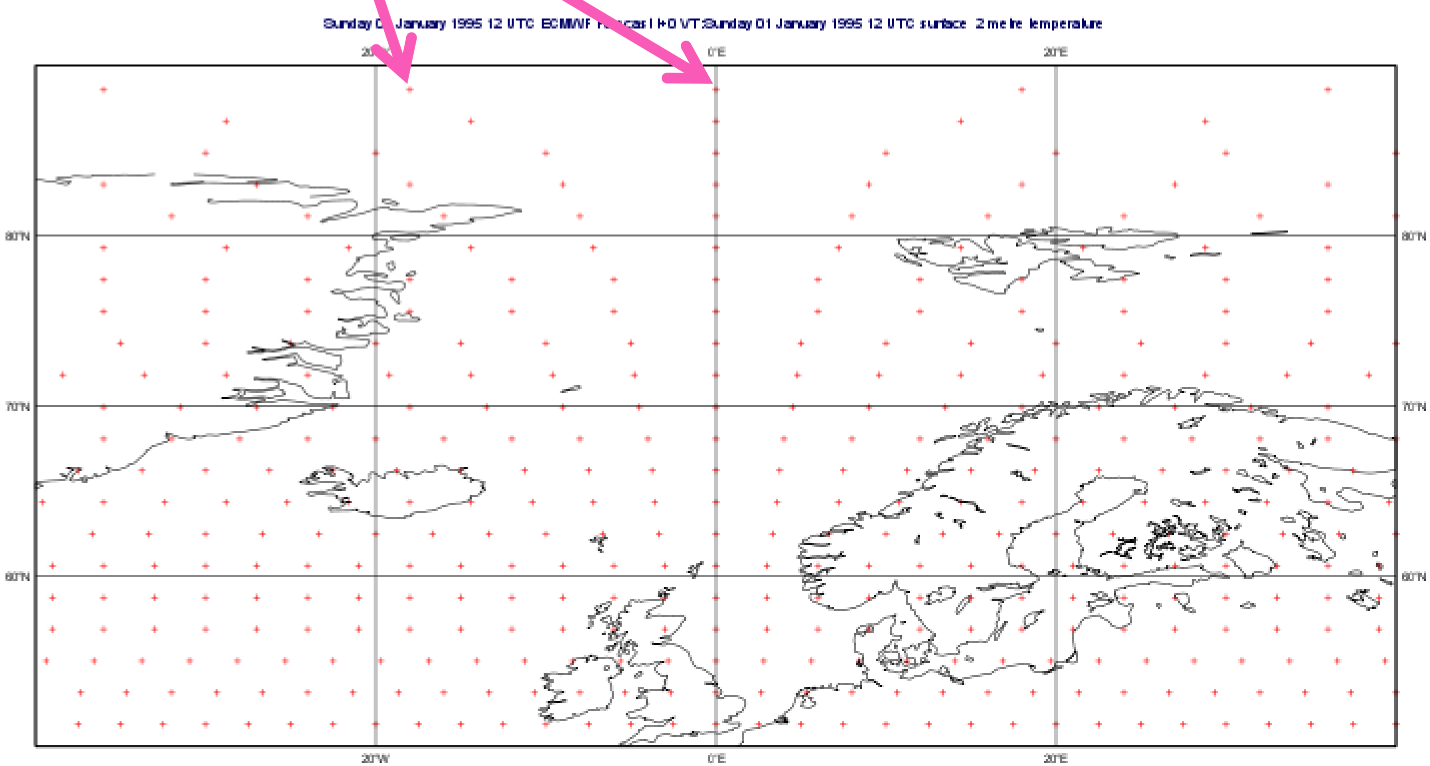

ECMWF

# What about the poles?




They are very close in this projection

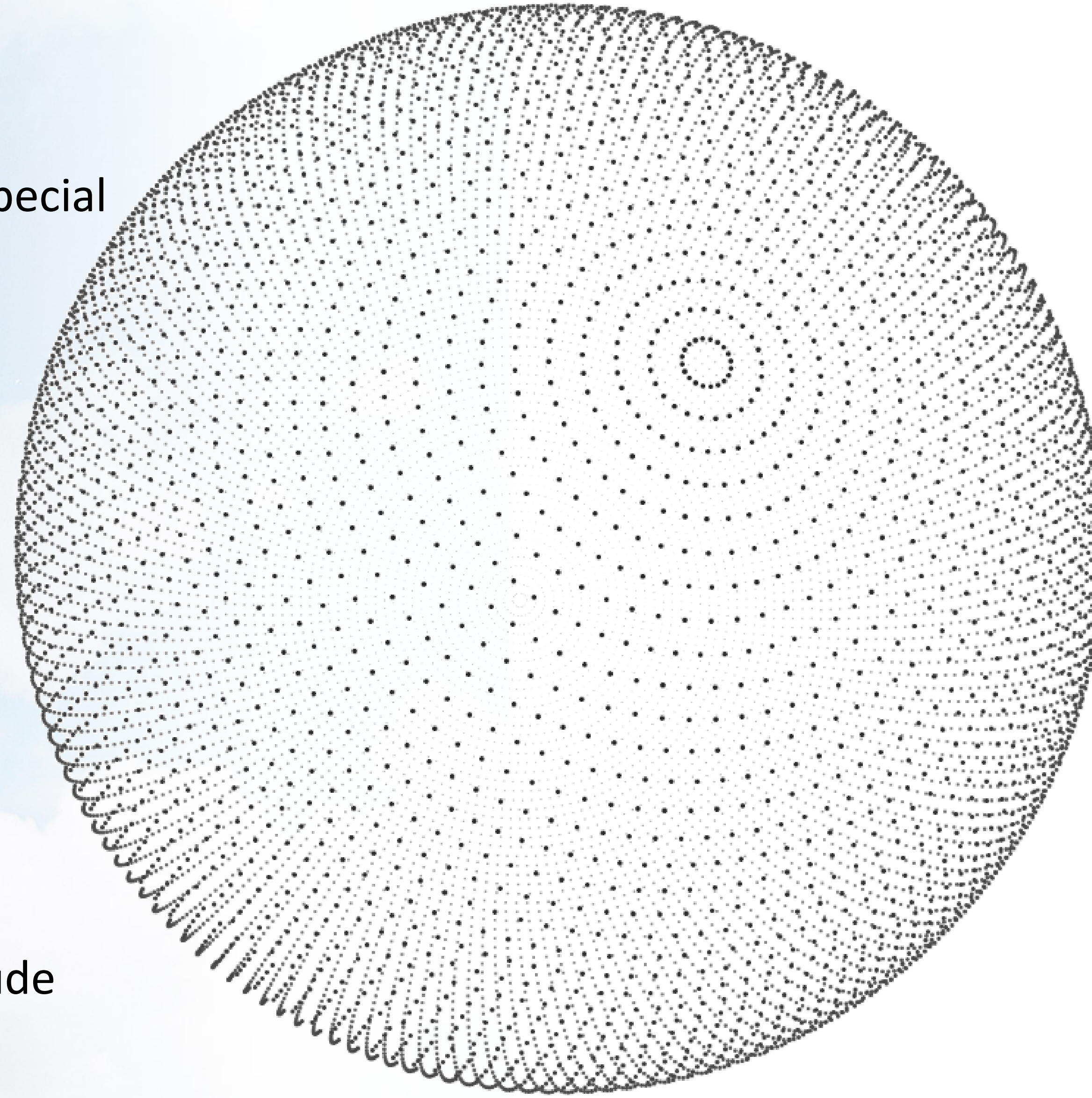
These points are far away in lat/lon (cylindrical projection)




ECMWF

# But the earth is (almost) a 3D sphere

- The poles are not special
- The anti-meridian is not special



- Solution: use (X,Y,Z) instead of latitude/longitude

# ECEF (Earth-Centered, Earth-Fixed) coordinate system

- Given latitude $\phi$, longitude $\lambda$, height $h$:

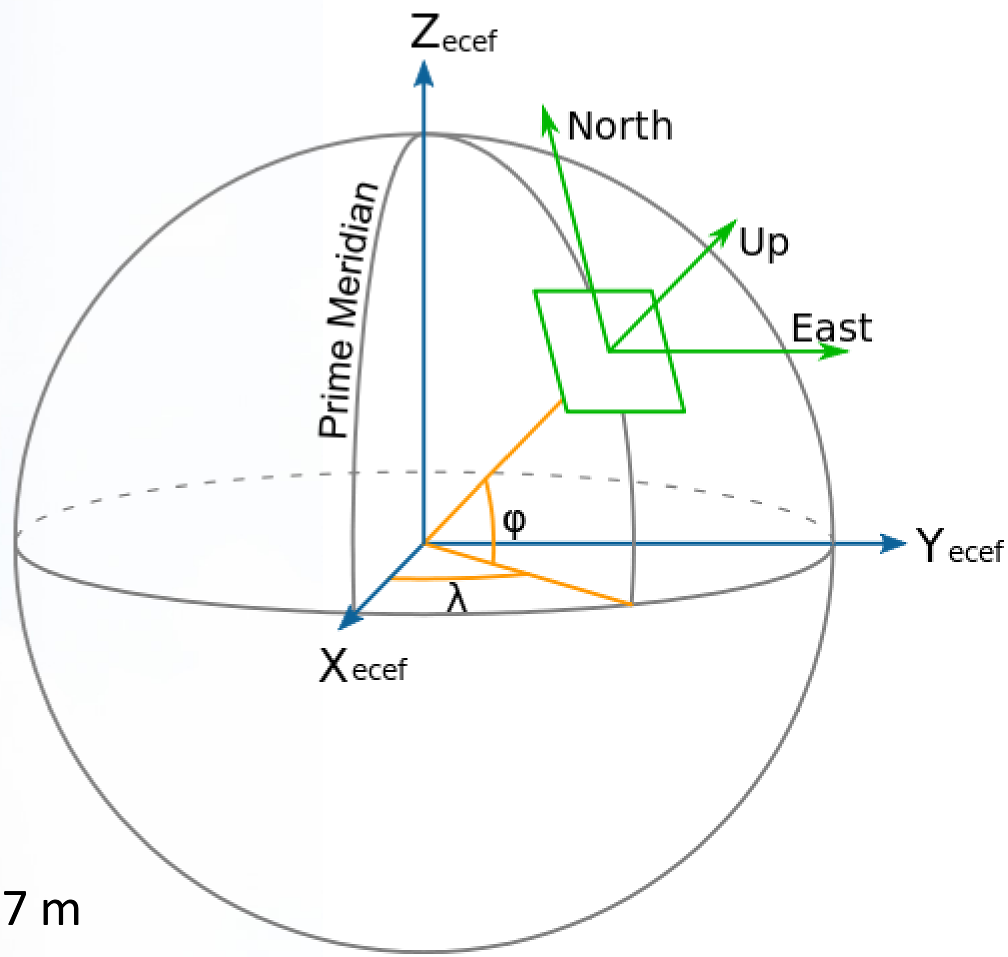
$$X = (N(\phi) + h)\cos\phi\cos\lambda$$
$$Y = (N(\phi) + h)\cos\phi\sin\lambda$$
$$Z = \left(N(\phi)(1 - e^2) + h\right)\sin\phi$$

- Where:

$$N(\phi) = \frac{a}{\sqrt{1 - e^2\sin^2\phi}},$$

- With *a* and *e* are the semi-major axis and the first numerical eccentricity of the ellipsoid respectively.

- For a spherical Earth: *e=0*, a=6378127 m



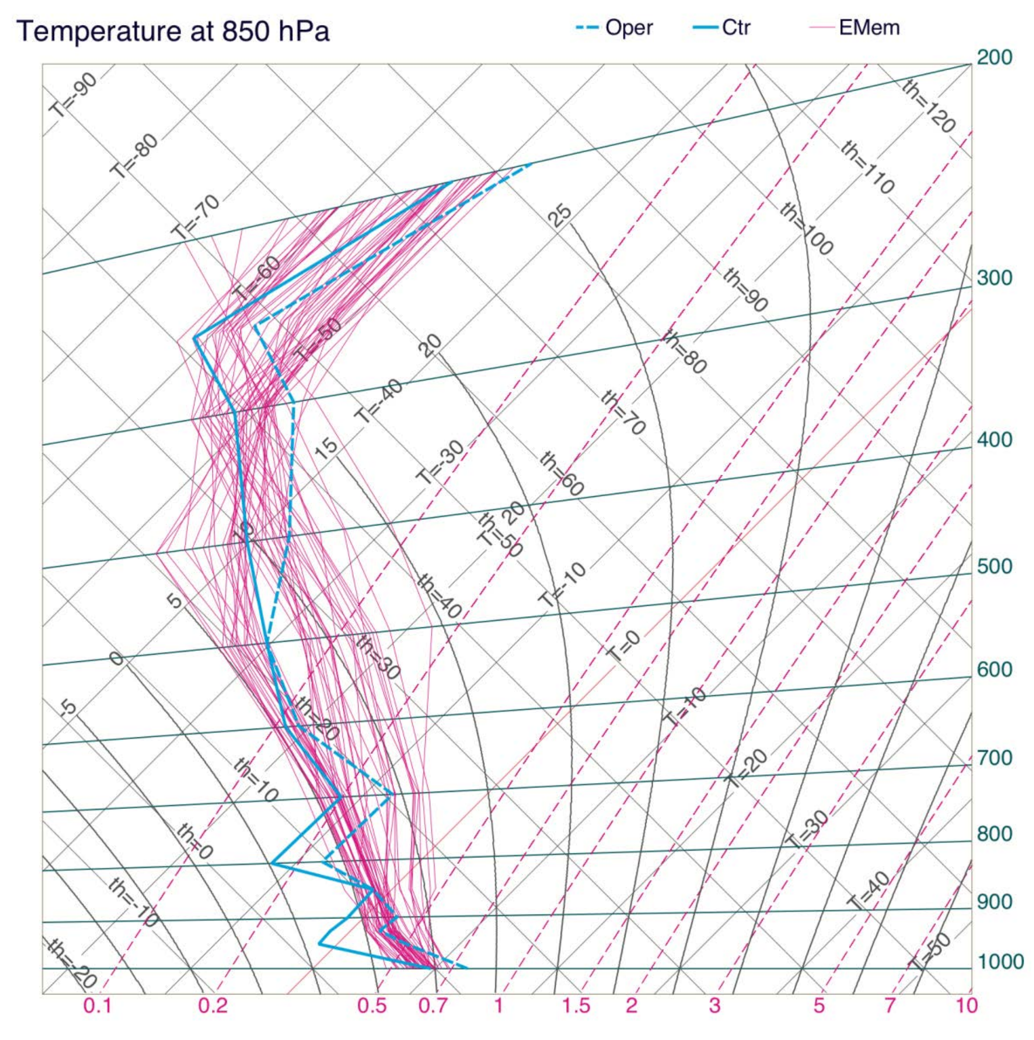
# Handling a user's request

- `{ "latitude": 51, "longitude": -1, "param": "t", "level": 500, …}`
- Use `"param", "level"`, … to lookup the FDB B-Tree index.
  - Get back a list of files and offsets where to find the GRIB records
  - Get back grid description, and GRIB packing information (data offset, number of bits, scaling factor, reference value…)
- For each GRIB
  - load corresponding kd-tree spatial index based on grid description and look up position of nearest point in GRIB using `"latitude"` and `"longitude"`
  - Open data file, position to `field offset + data offset + value index * number of bits/8`
  - Read `number of bits` bits into `packed value`
  - Return `packed value * scaling factor + reference value`
- Run each point extraction on a different thread
- Can anything that can be cached…

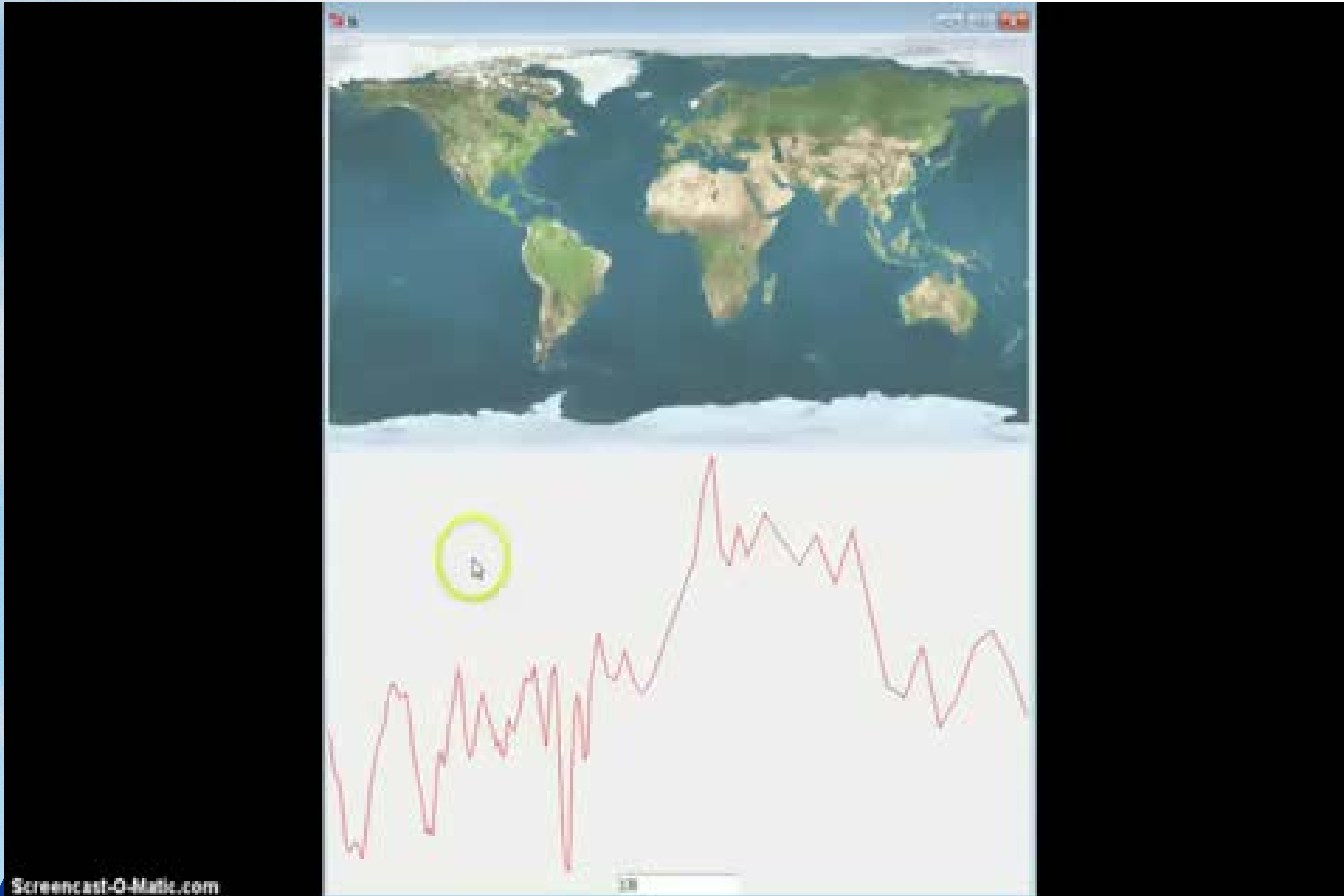


ECMWF

# Example: probabilistic tephigram



# Fast enough for interactive use (63 fields out of 3.5 millions, 1 point out of 2.1 millions )



# Conclusion

- First results are very promising
  - B-Tree to index the fields, created directly from the model
  - KD-Tree in ECEF coordinate system to index the points
    - Reuse the same tree for field with the same grid
  - Decode single data value directly from encoded GRIB
  - A lot of caching…
- Some remaining issues
  - HRES and ENS data not available at the same time
  - HPC I/O sub-system tuned for large sequential accesses, not for small random accesses
  - Most of our upper-air data is in spherical harmonics and need to be converted to grid point
    - We are working on a way to extract a value from a SH field

# Thank you.