Jupyter application in weather forecasting DeeepRain project

Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

DeepRain

Project

Features

Modern methods of machine learning (ML) to improve precipitation forecasts in Germany

Collaborative project

Partners

JÜLICH Forschungszentrum

JACOBS UNIVERSITY

UNIVERSITÄT BONN

Deutscher Wetterdienst
Wetter und Klima aus einer Hand
DWD

UNIVERSITÄT OSNABRÜCK

Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

DeepRain

Project

Objectives

Innovative Fusion of Earth System data

Ground-breaking application of deep learning for weather forecasting

24-hour predictions of precipitation on a 1 km grid (super-resolution)

Knowledge transfer among scientific disciplines and to end users

Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

JÜLICH Forschungszentrum

DeepRain

Data & Resources

Data

1.8 PB DWD data products

Some available online, and some only locally on two HPC systems

Machine – Learning (**ML**)

Mostly using NN utilizing well–known python libraries (scikit–learn, PyTorch, Tensorflow,. .)

HPC

JUWELS Cluster/ Booster hosted by FZJ

Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

JÜLICH Forschungszentrum

DeepRain

Challenges

Data

- Access
- Quality Assurance
- Provenance

ML

- Pre-processing
- Reproducibility

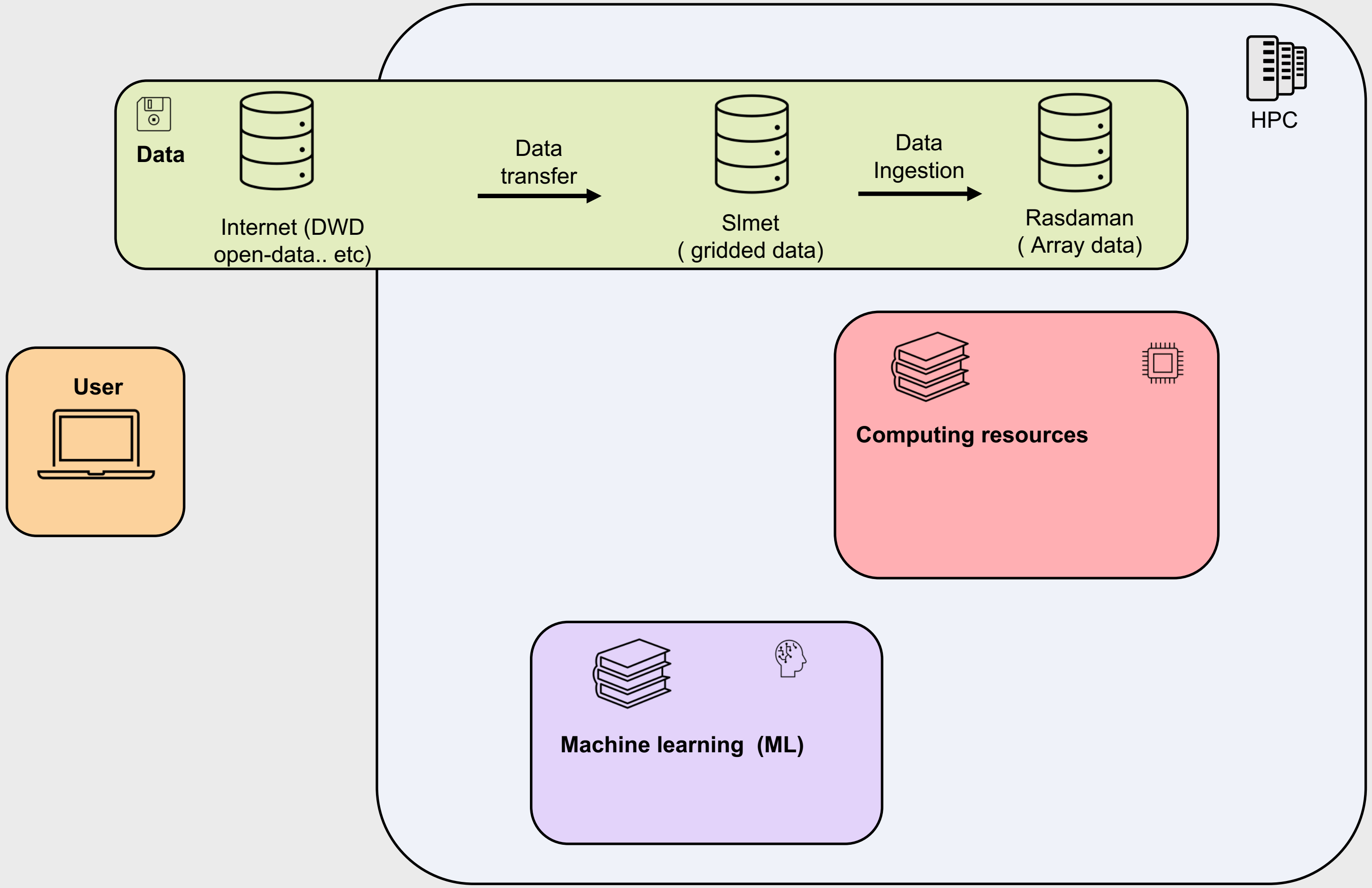
HPC

- Deployment & Environment
- Visualization

Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

4 interfaces

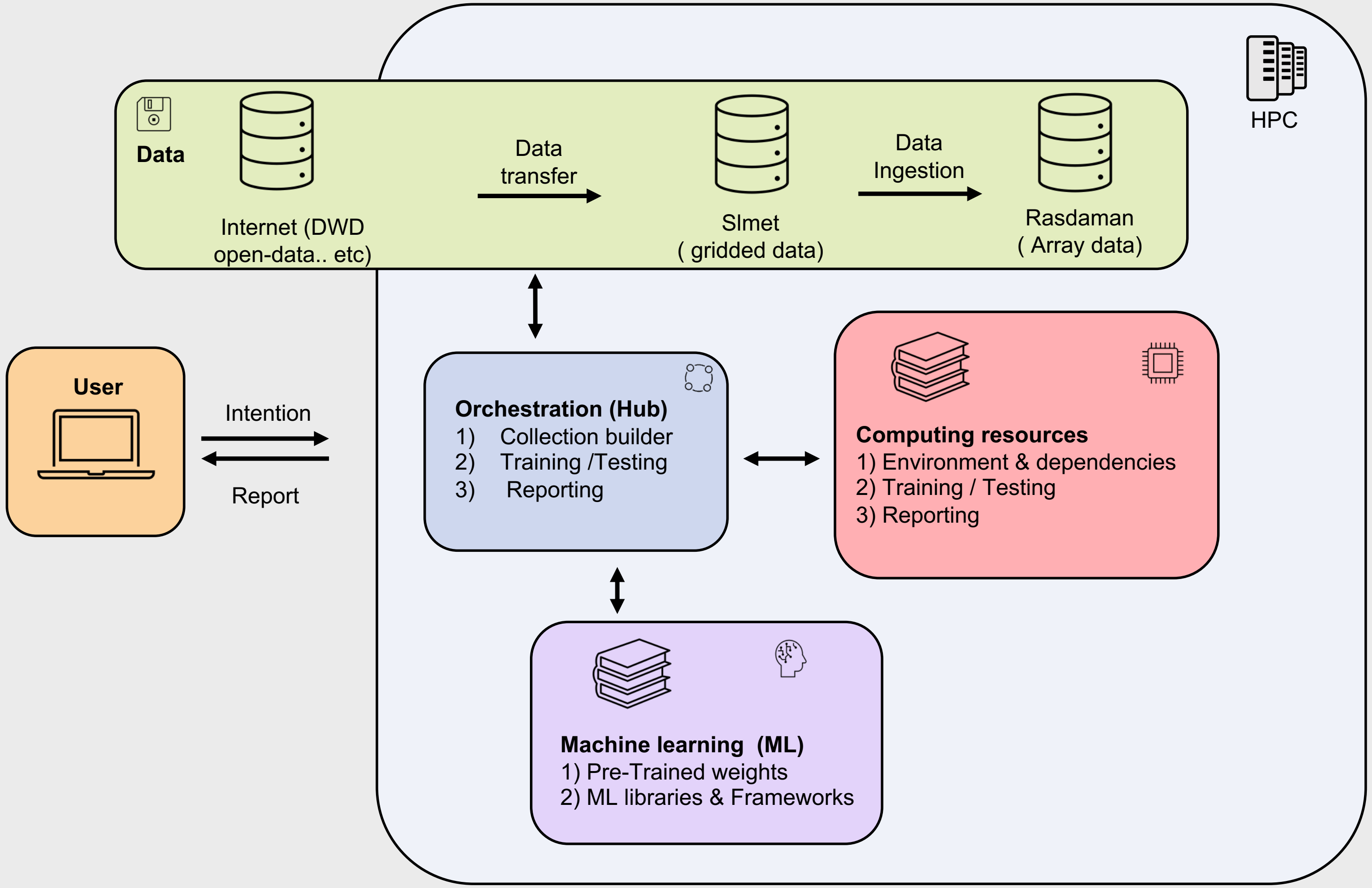
- User
- Data
- ML
- Computing resources



Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

CWFR in HPC

- Orchestration HUB

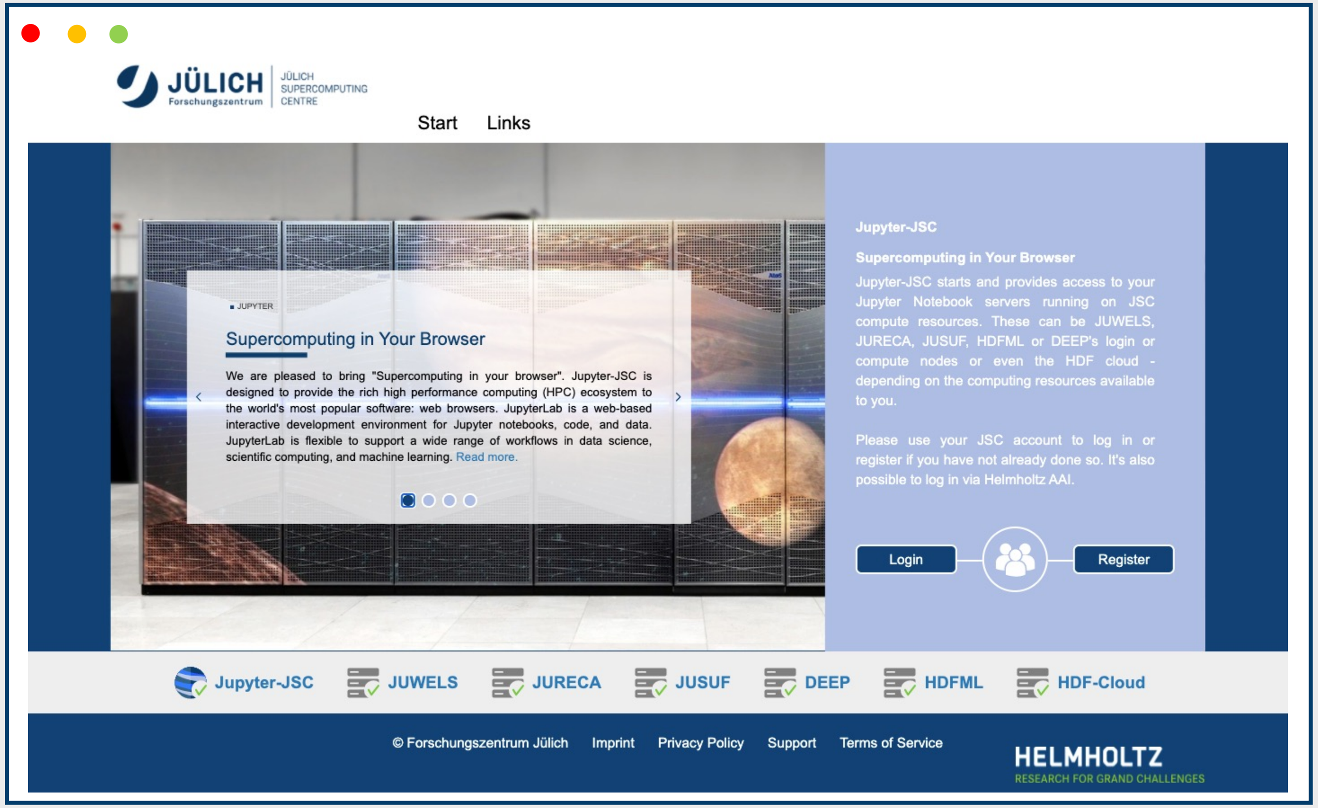


Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

CWFR in HPC

-> Jupyter-JSC

- Accessibility
- Lower technological barrier
- Better UI

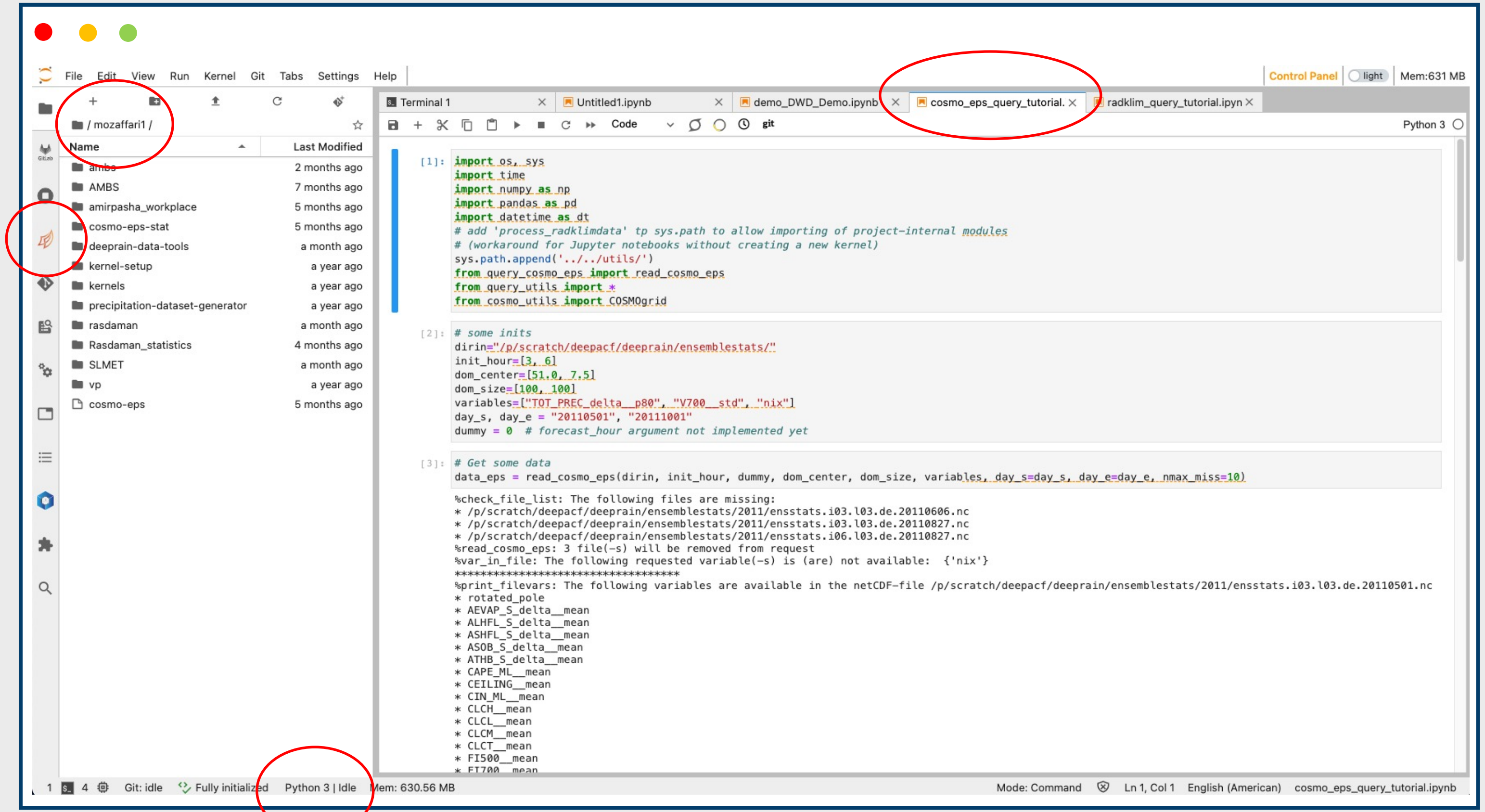


Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

CWFR in HPC

-> Jupyter-JSC

- Familiar UI
- Customizable kernel
- Access to HPC

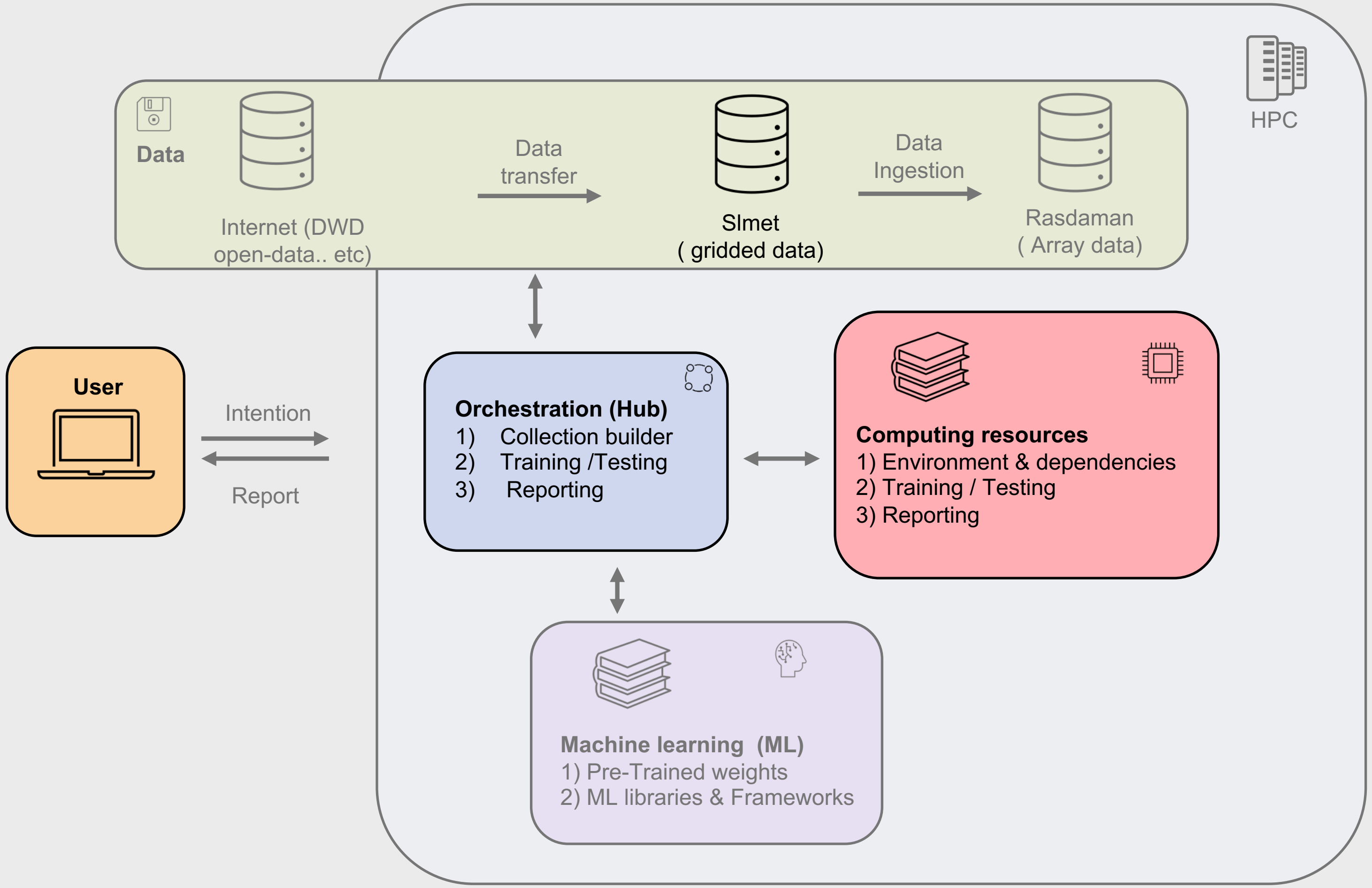


Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

JÜLICH Forschungszentrum

Example 1:

- **Data retrieval from multiple file-base system and cross validation**

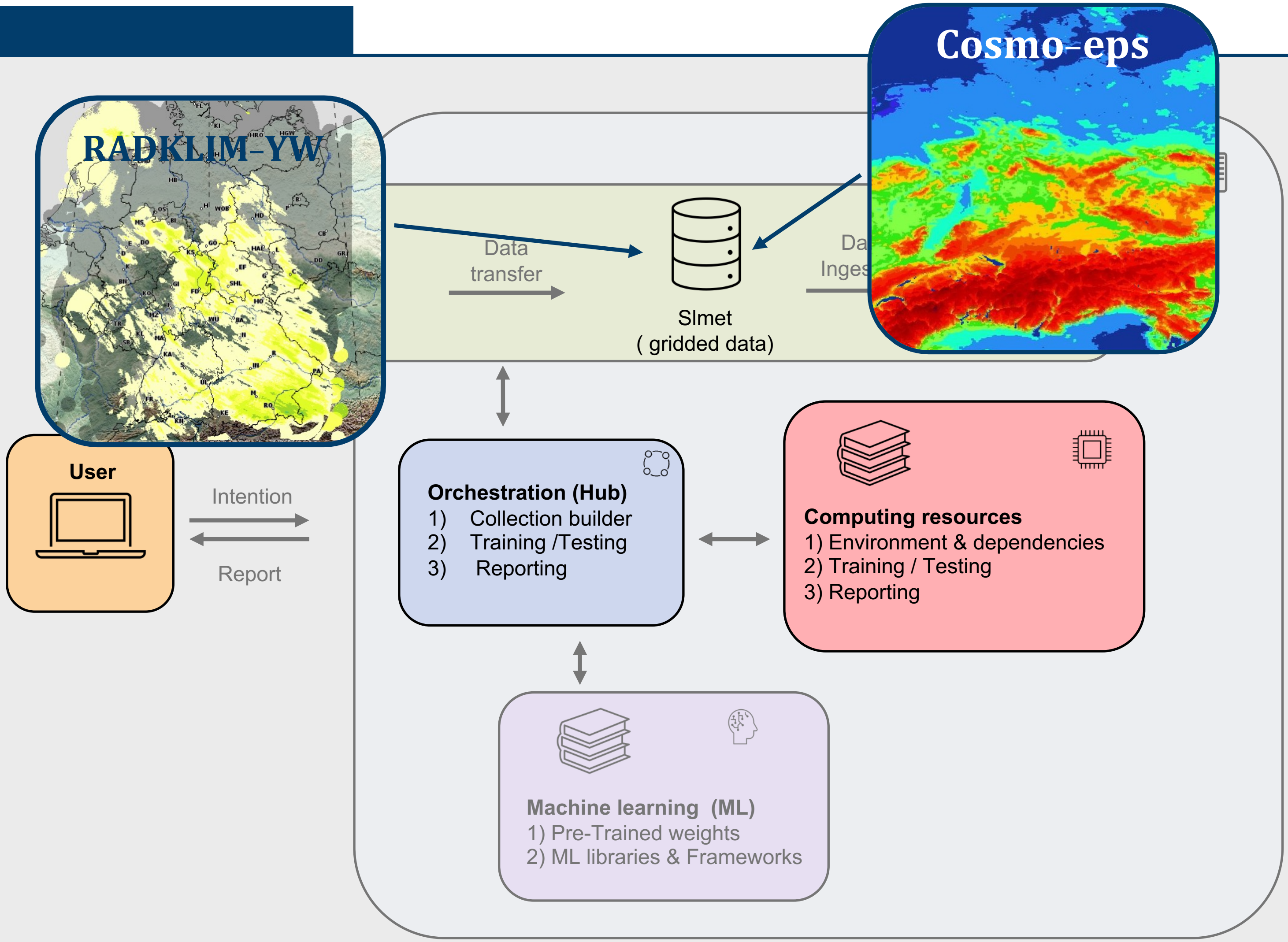


Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

JÜLICH Forschungszentrum

Example 1:

- Data retrieval from multiple file-base system and cross validation



Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

Example 1:

- **Data retrieval from multiple file-base system and cross validation**

```
# specify the top-level directory where the remapped data is stored in the Juelicher file system
# (!!! note the directory structure described at the end of the README.md !!!)
data_dir = "/p/scratch/deepacf/deeprain/radklim_process/netcdf/remapped/"

# specify the target domain in terms of dentral coordinate and domain size
dom_center = [51.0, 7.5]            # (unrotated) geogarphical coordinate (lat,lon) in degrees
                                    # a nearest neighbor approach is used to find the corresponding point on the COSM grid
dom_size = [100, 100]               # number of gridpoints in meridional (y) and zonal (x) direction of the target domain
date_start = "2016010100"           # start date of time period, format: YYYY
date_end   = "2017010100"           # end date of time period
freq       = "3H"                   # retrieve hourly precipitation every 3 hour (between date_start and date_end)
                                    # see https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#timeseries-offset
                                    # for valid frequency aliases
```

```
[1]: import os, sys
     import time
     import numpy as np
     import pandas as pd
     import datetime as dt
     # add 'process_radklimdata' tp sys.path to allow importing of project-internal modules
     # (workaround for Jupyter notebooks without creating a new kernel)
     sys.path.append('../../utils/')
     from query_cosmo_eps import read_cosmo_eps
     from query_utils import *
     from cosmo_utils import COSMOgrid

[2]: # some inits
     dirin="/p/scratch/deepacf/deeprain/ensemblestats/"
     init_hour=[3, 6]
     dom_center=[51.0, 7.5]
     dom_size=[100, 100]
     variables=["TOT_PREC_delta__p80", "V700__std", "nix"]
     day_s, day_e = "20110501", "20111001"
     dummy = 0  # forecast_hour argument not implemented yet

[3]: # Get some data
     data_eps = read_cosmo_eps(dirin, init_hour, dummy, dom_center, dom_size, variables, day_s=day_s, day_e=day_e, nmax_miss=10)
```

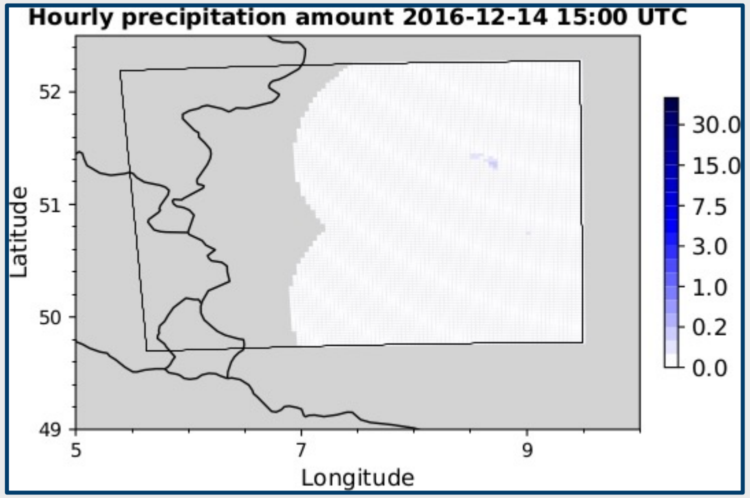

Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

JÜLICH Forschungszentrum

Example 1:

- **Data retrieval from multiple file-base system and cross validation**

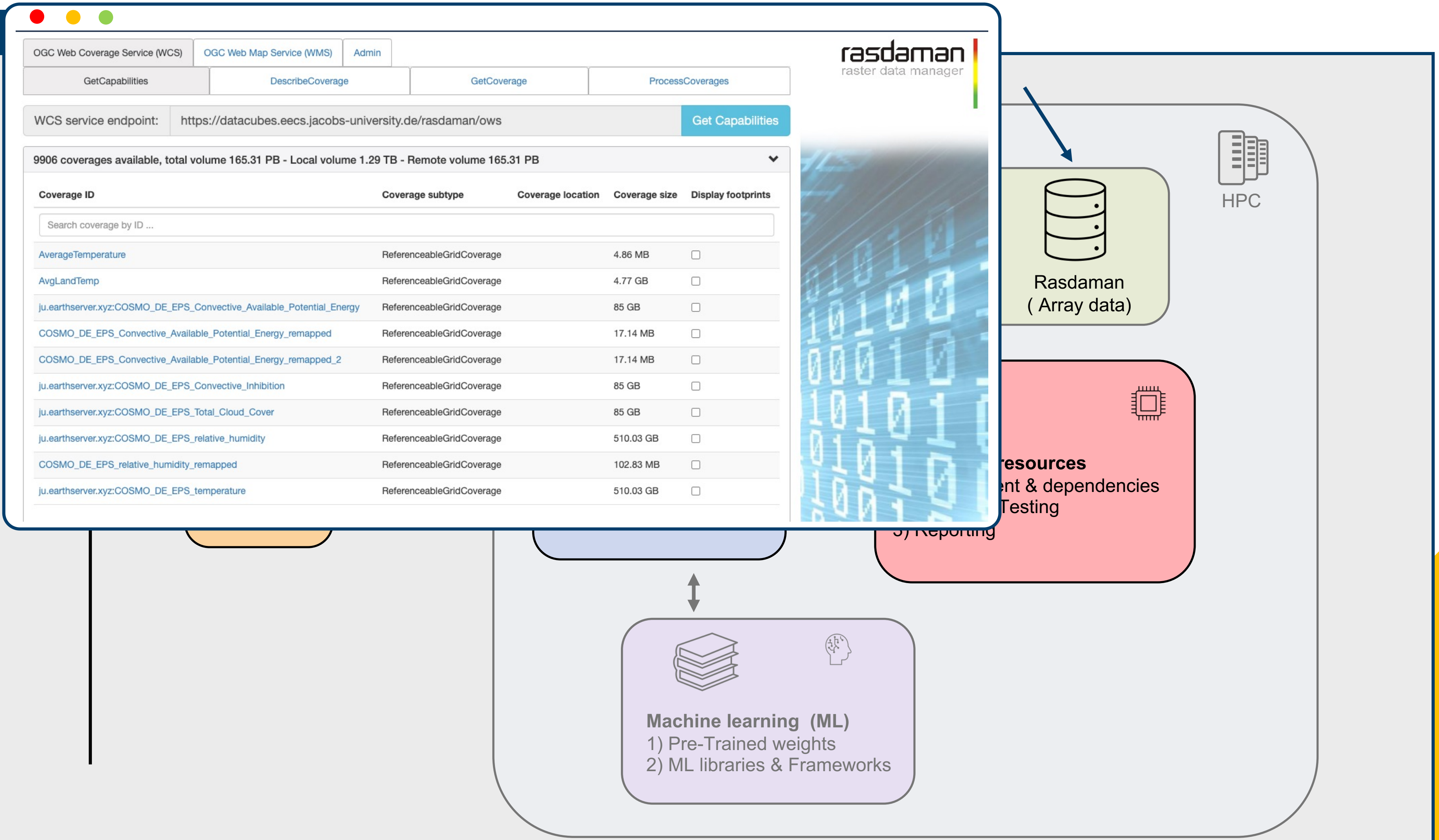
```
%check_file_list: The following files are missing:
* /p/scratch/deepacf/deeprain/ensemblestats/2011/ensstats.i03.l03.de.20110606.nc
* /p/scratch/deepacf/deeprain/ensemblestats/2011/ensstats.i03.l03.de.20110827.nc
* /p/scratch/deepacf/deeprain/ensemblestats/2011/ensstats.i06.l03.de.20110827.nc
%read_cosmo_eps: 3 file(-s) will be removed from request
%var_in_file: The following requested variable(-s) is (are) not available:  {'nix'}
***********************************
%print_filevars: The following variables are available in the netCDF-file /p/scratch/deepacf/deeprain/ensemblestats/2011/ensstats.i03.l03.de.20110501.nc
* rotated_pole
* AEVAP_S_delta__mean
* ALHFL_S_delta__mean
* ASHFL_S_delta__mean
* ASOB_S_delta__mean
* ATHB_S_delta__mean
* CAPE_ML__mean
* CEILING__mean
* CIN_ML__mean
* CLCH__mean
* CLCL__mean
* CLCM__mean
* CLCT__mean
```



Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

Example 2:

- **Data retrieval from an online array database and data visualization**



Coverage ID	Coverage subtype	Coverage location	Coverage size	Display footprints
AverageTemperature	ReferenceableGridCoverage		4.86 MB	☐
AvgLandTemp	ReferenceableGridCoverage		4.77 GB	☐
ju.earthserver.xyz:COSMO_DE_EPS_Convective_Available_Potential_Energy	ReferenceableGridCoverage		85 GB	☐
COSMO_DE_EPS_Convective_Available_Potential_Energy_remapped	ReferenceableGridCoverage		17.14 MB	☐
COSMO_DE_EPS_Convective_Available_Potential_Energy_remapped_2	ReferenceableGridCoverage		17.14 MB	☐
ju.earthserver.xyz:COSMO_DE_EPS_Convective_Inhibition	ReferenceableGridCoverage		85 GB	☐
ju.earthserver.xyz:COSMO_DE_EPS_Total_Cloud_Cover	ReferenceableGridCoverage		85 GB	☐
ju.earthserver.xyz:COSMO_DE_EPS_relative_humidity	ReferenceableGridCoverage		510.03 GB	☐
COSMO_DE_EPS_relative_humidity_remapped	ReferenceableGridCoverage		102.83 MB	☐
ju.earthserver.xyz:COSMO_DE_EPS_temperature	ReferenceableGridCoverage		510.03 GB	☐

Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

Example 2:

- **Data retrieval from an online array database and data visualization**

Service endpoint URL.

```
[3]: service_endpoint = "https://datacubes.eecs.jacobs-university.de/rasdaman/ows"
```

Function that retrieves the response of the query as an xarray dataset.

```
[4]: def get_query_response(query):
         '''
             Function returns a query as a netcdf byte-encoded response.

             Parameters: None

             Returns xarray.Dataset
         '''

         time0 = time.time()
         query_response = requests.post(service_endpoint, data={'query': query})

         # Convert bytes to file-like object
         netcdf_file = io.BytesIO(query_response.content)

         # Convert the netcdf_file like object to a xarray dataset.
         ds = xr.open_dataset(netcdf_file)

         print("Data query took {0:5.2f}s...".format(time.time() - time0))

         # Retun the xarray dataset.
         return ds
```

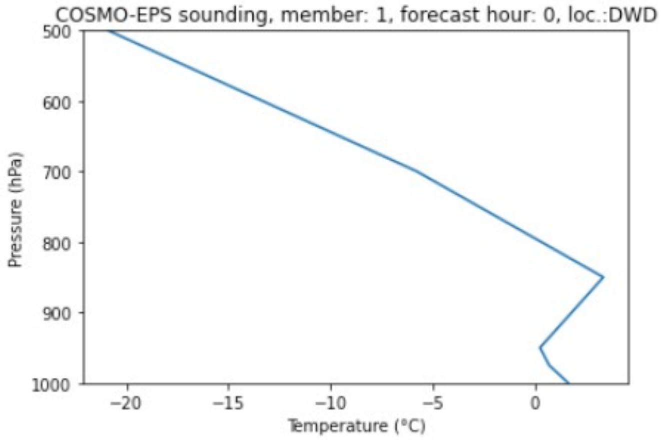

Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

Example 2:

- **Data retrieval from an online array database and data visualization**

```
[10]: fig, ax = plt.subplots()

      plt.plot(ds['temperature'], ds['pressure_level']*.01)
      plt.ylim(1000., 500.)
      plt.ylabel('Pressure (hPa)')
      plt.xlabel('Temperature (°C)')
      plt.title("COSMO-EPS sounding, member: 1, forecast hour: 0, loc.:DWD")
```

```
[10]: Text(0.5, 1.0, 'COSMO-EPS sounding, member: 1, forecast hour: 0, loc.:DWD')
```



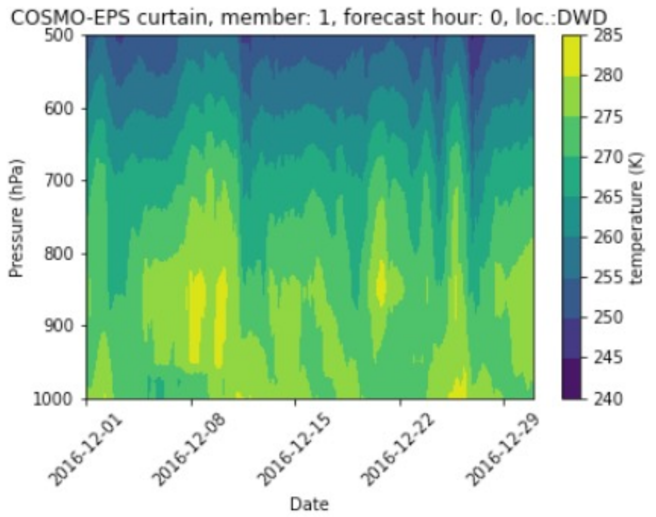
Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

JÜLICH Forschungszentrum

Example 2:

- **Data retrieval from an online array database and data visualization**

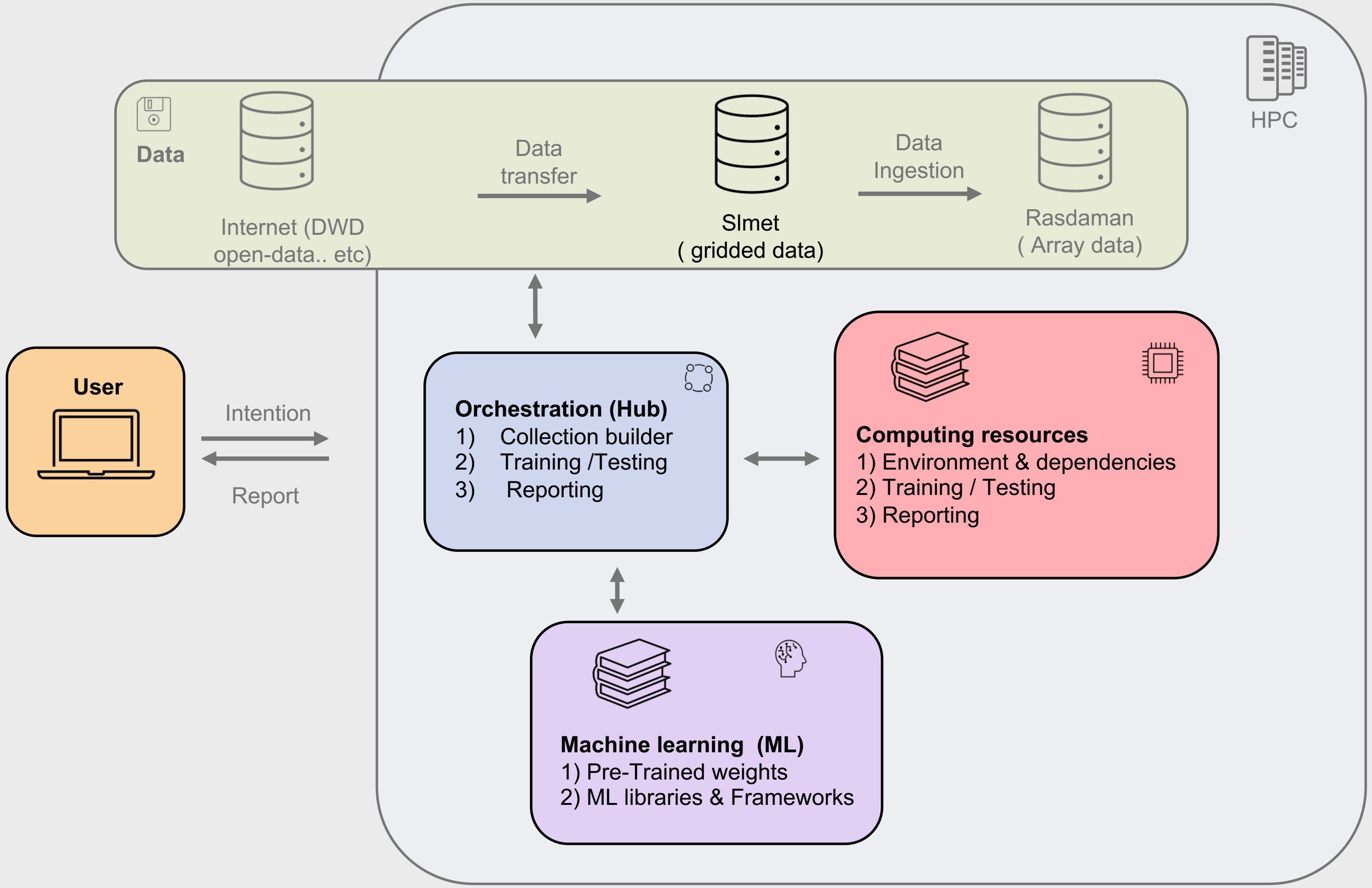
```
[12]: times = ds["ansi"].values

fig, ax = plt.subplots()

c = plt.contourf(times, ds['pressure_level']*0.01, ds['temperature'].T)
cb = plt.colorbar(c, label='temperature (K)')
plt.title("COSMO-EPS curtain, member: 1, forecast hour: 0, loc.:DWD")
plt.ylim(1000., 500.)
plt.xlabel('Date')
plt.ylabel('Pressure (hPa)')

add_date_label(ax, times)
```



Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)
JÜLICH Forschungszentrum

Example 3:

- **ML results retrieval and visualization of forecast and skill score**



Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

JÜLICH Forschungszentrum

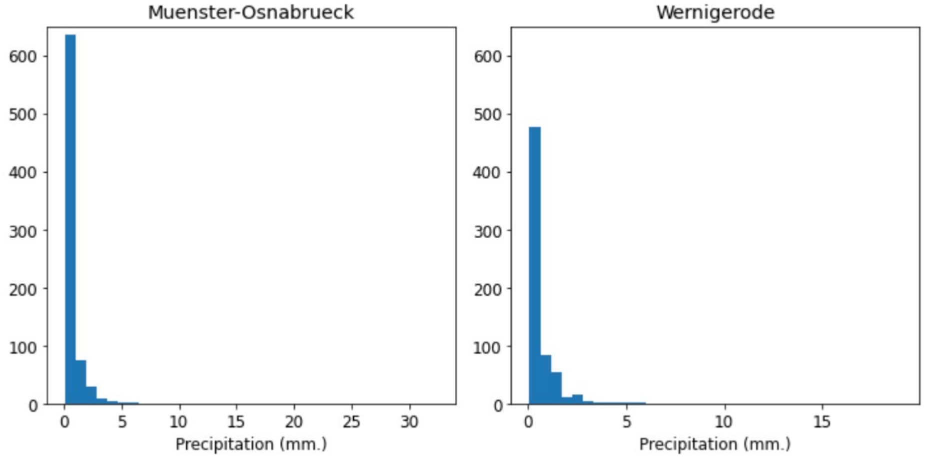
Example 3:

- **ML results retrieval and visualization of forecast and skill score**

Histograms of rain data

```
rain_histogram = []

for i, station in enumerate(stations):
    train_y = np.loadtxt('data/'+ station +'_0_4_5x5_allFeatures_created_2020-10-23/train_y.csv', delimiter=',')
    test_y  = np.loadtxt('data/'+ station +'_0_4_5x5_allFeatures_created_2020-10-23/test_y.csv', delimiter=',')
    y = np.concatenate((train_y, test_y))
    y = np.exp(y)
    print(station, y.shape)
    rain_histogram.append(y)

bins = 35

def format_hist_rain(ax, data, station):
    ax.hist(data, bins)
    ax.set_title(station)
    ax.set_xlabel('Precipitation (mm.)')
    ax.set_ylim([0,650])
```

```
np.mean(precipitation) for Muenster-Osnabrueck = 0.6601049868766404
np.mean(precipitation) for Wernigerode = 0.6971684053651266
np.mean(precipitation) for Braunlage = 0.7928571428571428
np.mean(precipitation) for Redlendorf = 0.7919494869771113
```



Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

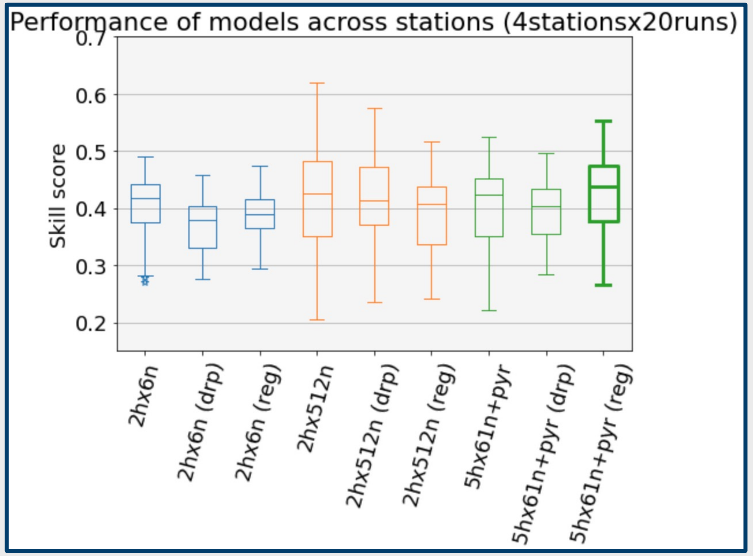
Example 3:

- **ML results retrieval and visualization of forecast and skill score**

Model comparation across stations

```
model_comp = data = np.empty([9, 80])

for j, model in enumerate(names):

    skill_model = []
    for i, station in enumerate(stations):

        skill_model_in_station = np.loadtxt(path_all+'/'+model+'/'+station+'/'+metric+'.txt', dtype=np.float64, delimiter=',')
        skill_model.append(skill_model_in_station)

    model_comp[j] = np.concatenate(skill_model)

print(model_comp.shape)


print('Median of skill score for models =', np.median(model_comp, axis=1))
print('Model with the highest median =', names_for_ticks[np.argmax(np.median(model_comp, axis=1))])
```

```
(9, 80)
Median of skill score for models = [0.41711134 0.37980181 0.38917765 0.42484066 0.41382593 0.40783788
 0.42309215 0.40392879 0.4383606 ]
Model with the highest median = 5hx61n+pyr (reg)
```



Member of the Helmholtz Association
Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

Summary

- •Jupyter (deployed on HPC system) provides an easy–to–use framework for complicated workflows
- • User–friendly interface helps the user to easily interact with extensive data and HPC system
- •Ease of porting a workflow from a local machine to an HPC system with minimal adjustment

Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

Thank you

Amirpasha Mozaffari (a.mozaffari@fz-juelich.de)

JÜLICH
Forschungszentrum