



# A versatile environment for large-scale geospatial data processing with HTCondor

**Antonio Puertas Gallardo**  
**Dario Rodriguez-Aseretto**  
**Pierre Soille**

European Commission, Joint Research Centre  
Directorate I Competences, Unit I.3 Text and Data Mining

# OUTLINE

- DG Joint Research Centre
- JRC Earth Observation Data and Processing Platform
- Batch processing architecture
- Geospatial analysis with HTCondor
  - *Mosaicking Copernicus Sentinel-1 Data at Global level*
  - *Optimizing Sentinel-2 image selection in a Big Data Context*
  - *Mediterranean Ecosystem simulation*
- Lessons learned and open questions



# DG Joint Research Centre



Established in

**1957**



**3000 staff**

Almost **75%** are scientists and researchers.



**10**

Directorates



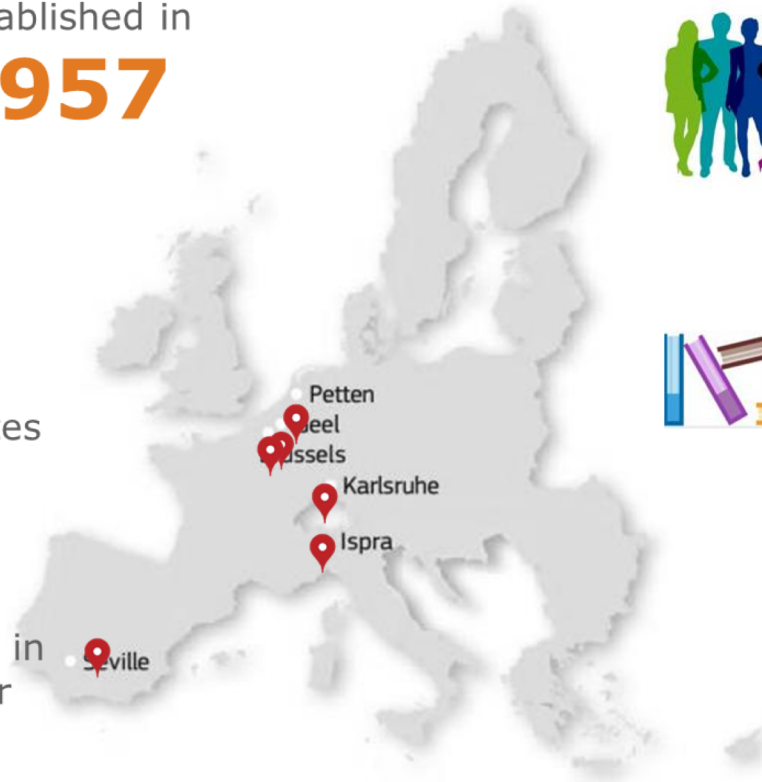
**>1000**

Publications per year



**6**

Locations in  
5 Member  
States



**42**

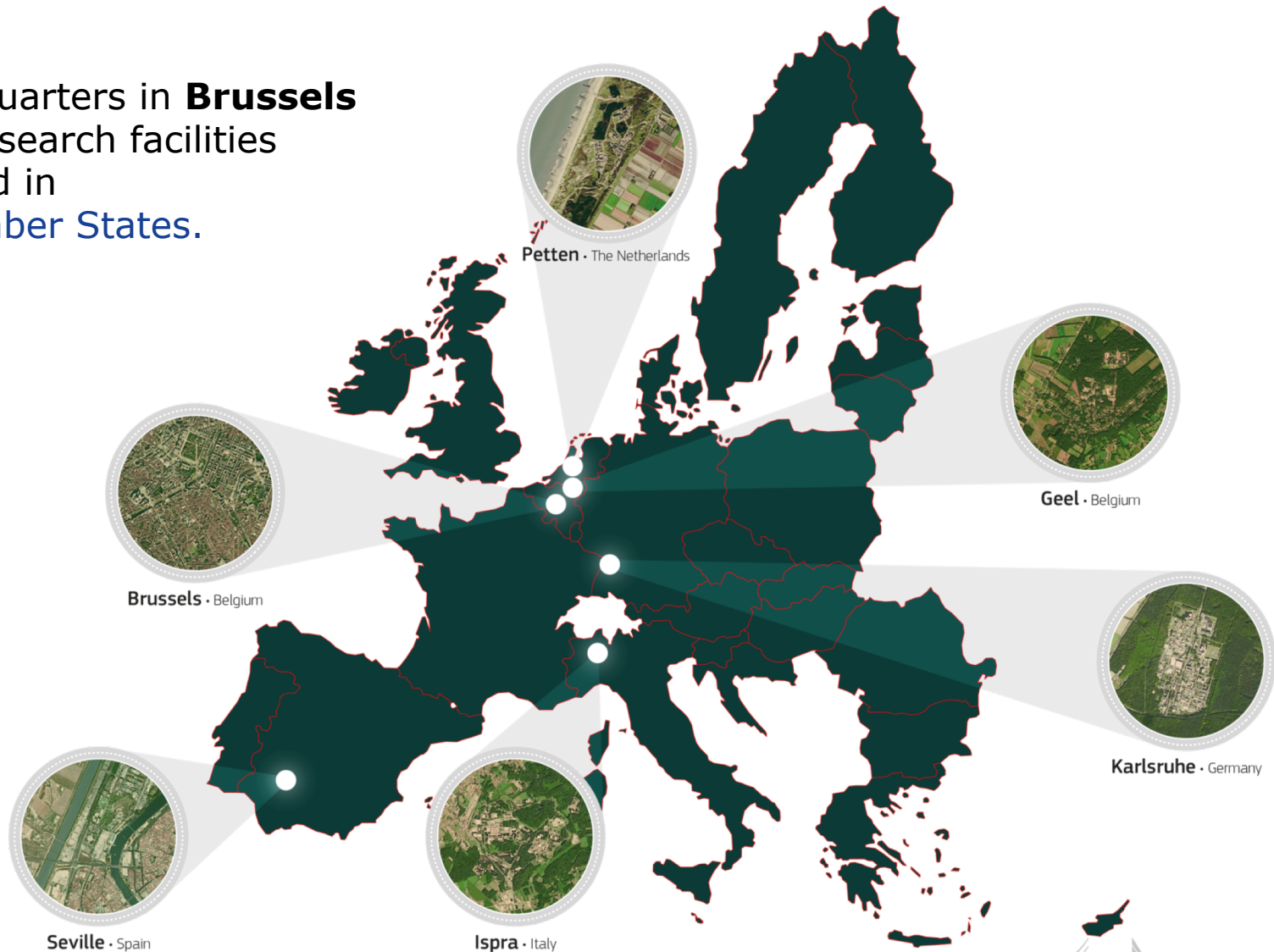
Large scale  
facilities



European  
Commission

# The Joint Research Centre at a glance

Headquarters in **Brussels**  
and research facilities  
located in  
**5 Member States.**







## **DG JRC's Vision:**

**"To play a central role in creating, managing and making sense of the collective scientific knowledge for better EU policy."**

## **DG JRC's Mission:**

**"As the science and knowledge service of the Commission our mission is to support EU policies with independent evidence throughout the whole policy cycle."**

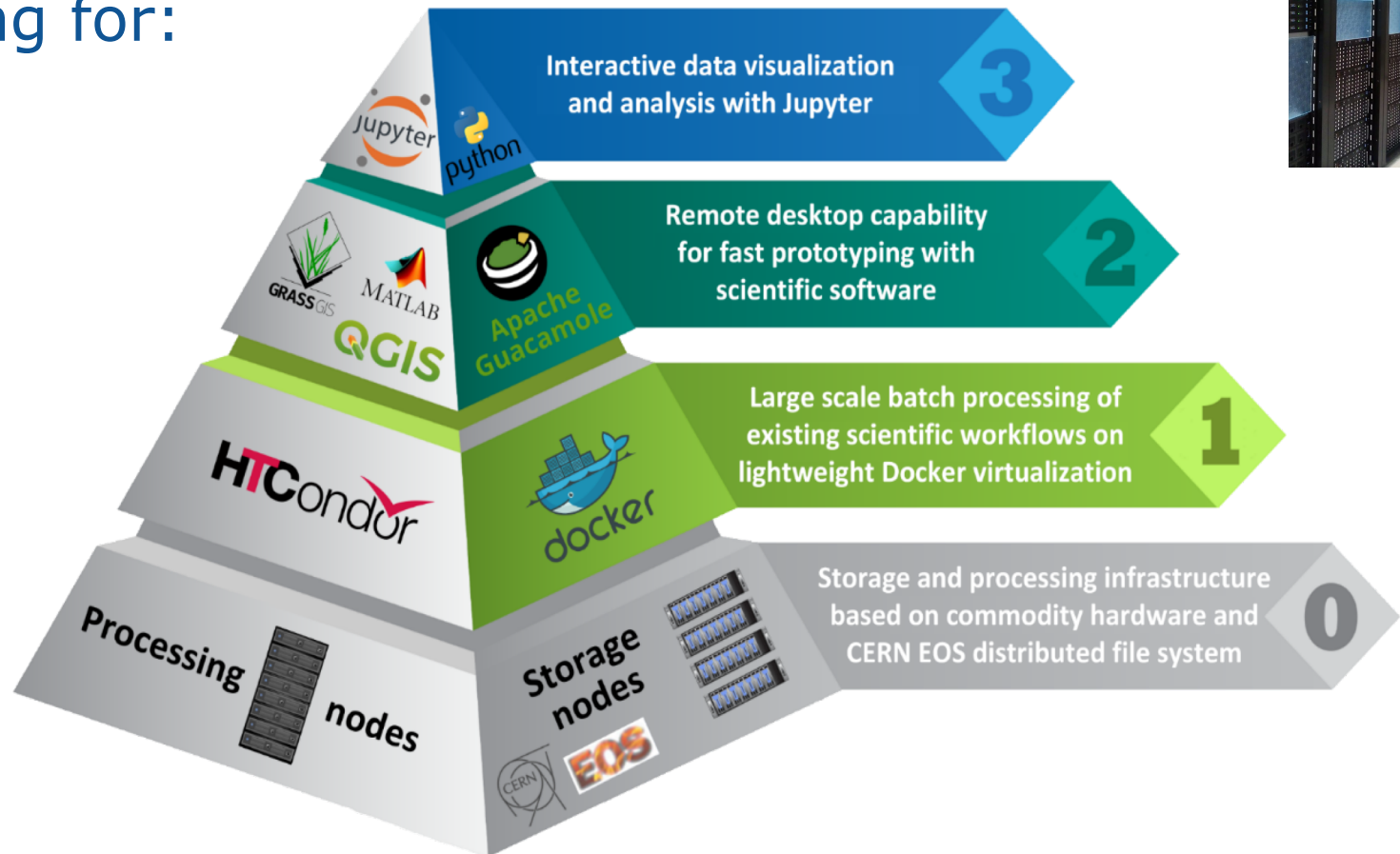


## DG JRC Role

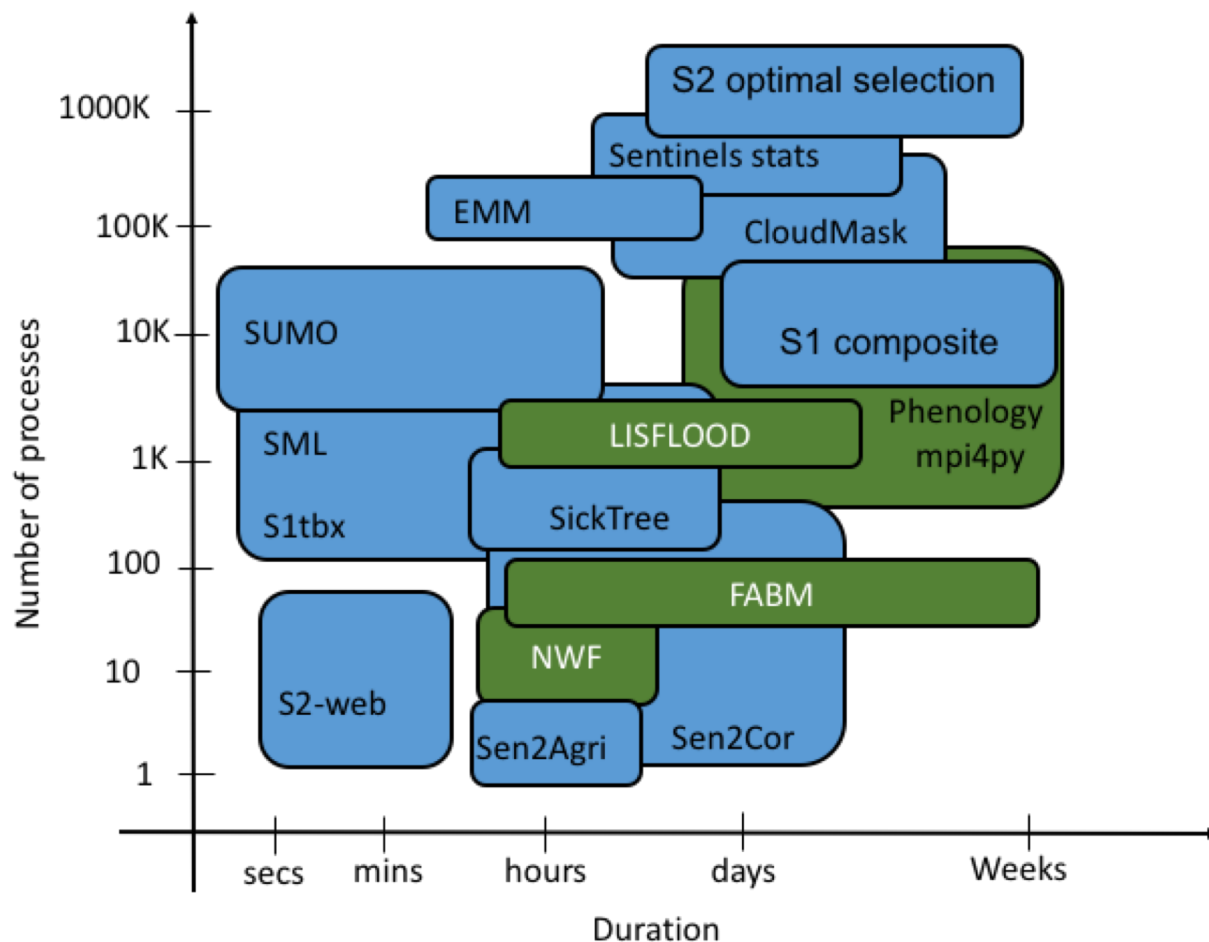
- **Independent** of private, commercial or national interests
- **Policy neutral**: has no policy agenda of its own
- **Transversal** service - cuts across policy silos
- **30%** of activities in **policy preparation**, **70%** in **implementation**
- **Expertise in a wide range of areas** from economics and financial analysis to energy and transport, health, environment and nuclear safeguards

# JRC Earth Observation Data and Processing Platform

**Versatile** platform bringing the users to the data and allowing for:



# Why versatile batch system?



Apps type:

- Python & mpi4Py
- C & C++
- MATLAB runtime
- Java-Tomcat
- Tensorflow+Keras (CPU/GPU based)
- Fortran+MPI

# Large scale batch processing

- Job submission via:
  - shell in web base remote desktop (Guacamole)
  - Jupyter notebooks (JupyterLab) thanks to HTCondor Python bindings
- Monitoring using customize dashboard in Grafana
- Git repository for Dockerfile to be deployed into the batch system
- Private Docker registry from which images are automatically pulled by the processing host
- One proc-user by group using condor\_map

# Software Components

## JupyterLab

## Guacamole

Python  
binding

**HTCondor**  
High Throughput Computing

Condor\_map

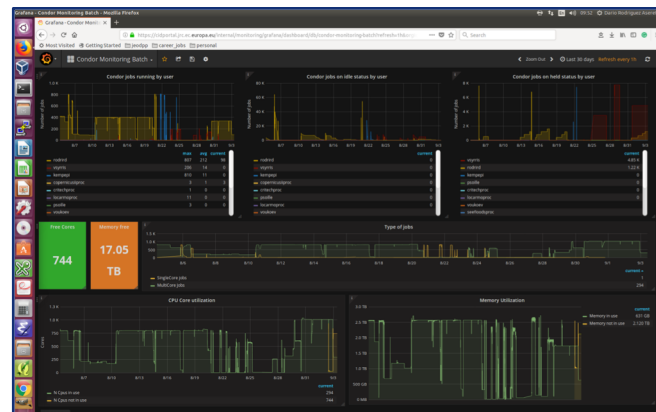
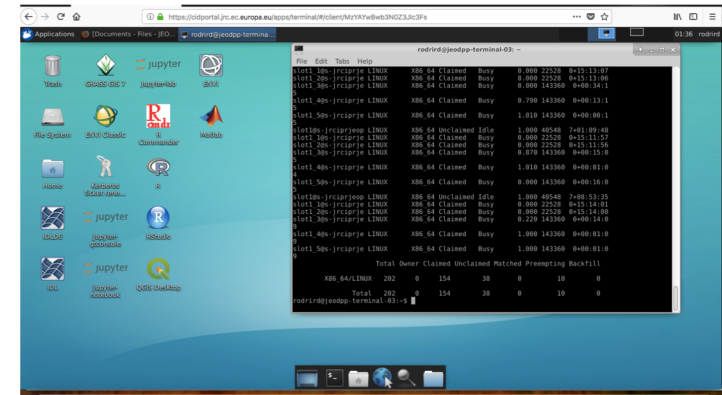
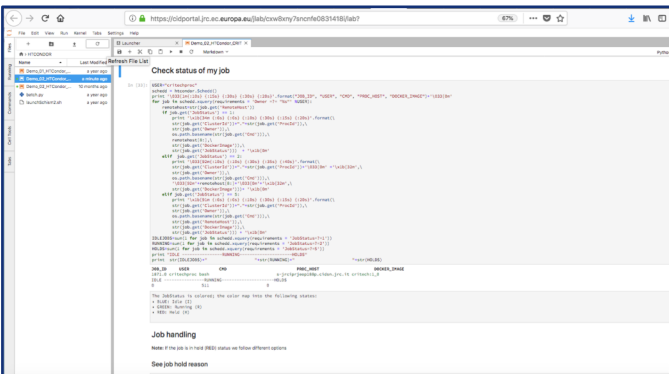
## Grafana+Graphite+Ganglia

## Docker Registry

Gangliad

Docker  
Univ.

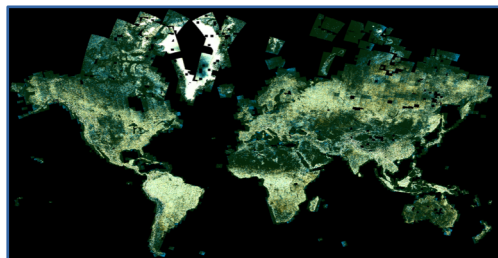
Joint  
Research  
Centre



```
root@sjrciprjeop154p:~# docker run -ti jeoreg.cidsn.jrc.it:5000/jeodpp-htcondor/jeodpp_sitbx:test /bin/bash
Unable to find image 'jeoreg.cidsn.jrc.it:5000/jeodpp-htcondor/jeodpp_sitbx:test' locally
test: Pulling from jeodpp-htcondor/jeodpp_sitbx
79f2514705: Pull complete
6ad56d5fc149: Pull complete
170e558760e8: Pull complete
65450e223f5: Pull complete
f01dc2e444: Pull complete
1a05b2780d36: Pull complete
191a8408ed1f: Pull complete
471a66cf0a4b: Pull complete
e5d170102b57: Pull complete
921ac18fa4d6: Pull complete
7fb8f991c444: Extracting [=====] 1.087GB/1.537GB
64cc796db6d: Download complete
62e236d4c908: Download complete
32eb73a05aa0: Download complete
97b13ae6222b: Download complete
6e0dc8639f2: Download complete
355f9b812e2a8: Download complete
1654c24eb876: Download complete
2e74ff68f3d9: Download complete
4b84368f9398: Download complete
97d12807f8e: Download complete
1cb0ec506450: Download complete
934641ff5de5: Download complete
87696675731: Download complete
6799b7fa334: Download complete
5a814ef6bc: Download complete
```



# Mosaicking Copernicus Sentinel-1 Data at Global level - Running in Docker universe



- More than 10K inputs between Sentinel 1 (SAR) remote images and (DEM).
- DAGMan workflow include:
  - Border noise removal with S1tbx ver2.0.2
  - Orthorectification and thermal noise removal with S1tbx ver6.0
  - Merging and mosaicking using in-house libraries based on Python and C++

```
File Edit View Run Kernel Tabs Settings Help
Demo_01_Coll X GSW.pyrb Demo_06_Oth X
Python 2
+ - X Copy Paste Run Code
Display JRC Global Surface Water Collection layers and Global Human Settlement Layers (yellow) on top of Global S1 Mosaic

The Global Surface Water (GSW) layers rendered in this notebook downloaded from: https://global-surface-water.appspot.com/download

They can also be viewed in a JEOCPP static web-viewer at https://cidportal.jrc.ec.europa.eu/services/webview/jeoopp/databrowser/

Reference: Jean-Francois Pekel, Andrew Cottam, Noel Gorelick, S. Belward, High-resolution mapping of global surface water and long-term changes. Nature 540, 418-422 (2016). doi:10.1038/nature20584

In [1]: from ipynbwidgets import widgets
map = Map(backend='Map')
label = widgets.Label(layout=widgets.Layout(width='900px'))
label

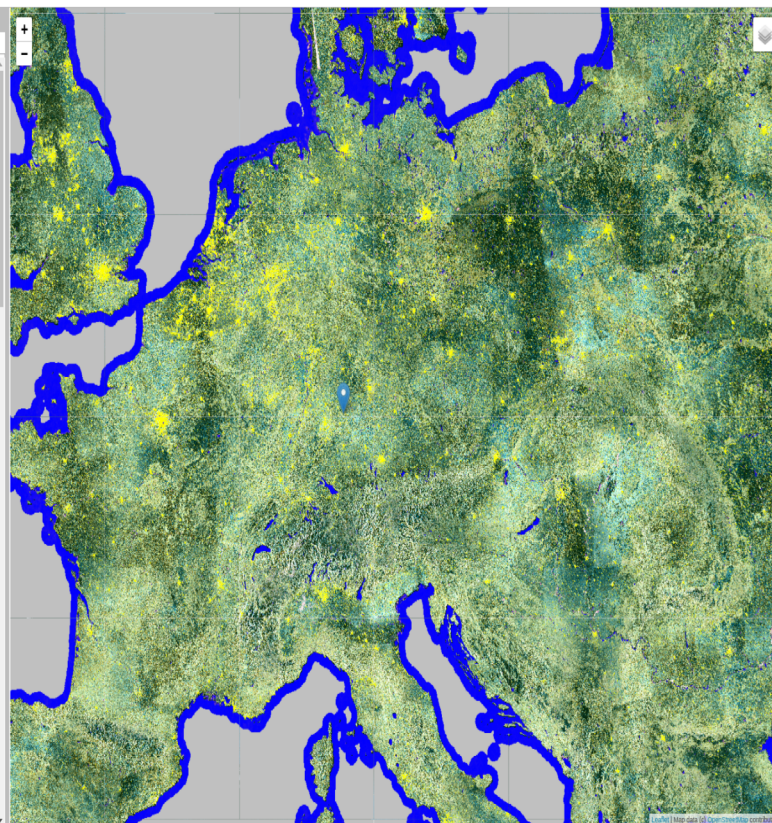
Move the marker on the map to identify pixel values

In [2]: Inter.listImageCollections()

S2 - Sentinel2 collection
LANDSAT - Landsat collection
COP - CoreS2 mosaic over Europe (parameter "mode" can be "feathering" or "seamless")
DEM - Digital Elevation Model (parameter "source" can be "EUDEM" or "SRTM" or "GEBCO")
GHSL - Global Human Settlement Layer
NOVI - NOVI Time series from 1998 to 2013
S1MSA1C - Sentinel1 sample mosaic
GSW - Global Surface Water (parameter "layer" can be "change", "extent", "occurrence", "recurrence", "seasonality" or "transitions")
CORINE - Corine Land Cover 2012
URBANATLAS - Urban Atlas 2006 and 2012
BASEMAP - Basemap collection
CUSTOM - Custom collection

View one layer at a time with a pixel identifier

View one layer at a time with a random palette and a pixel value identifier. The pixel value of the position indicated by the identifier is displayed below the first cell of this notebook together with its geographic coordinates. The identifier needs
```



Global Human Settlement Layer with Global Surface Water Occurrence on top of Global S1 mosaic.

GHSL-S1 [doi:10.1080/01431161.2017.1392642](https://doi.org/10.1080/01431161.2017.1392642)

Global Surface Water [doi:10.1038/nature20584](https://doi.org/10.1038/nature20584)

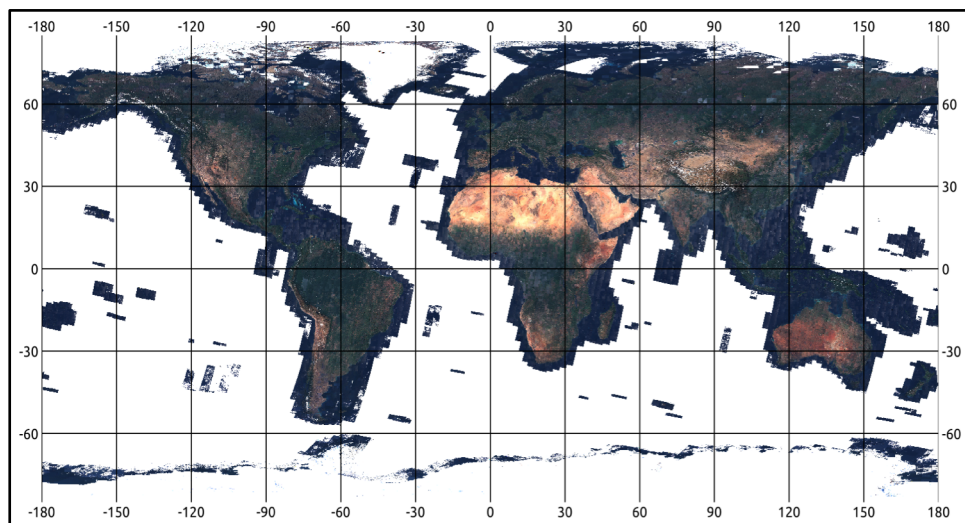
Global S1-Mosaic [doi:10.1109/TBDATA.2018.2846265](https://doi.org/10.1109/TBDATA.2018.2846265)

See also <https://cidportal.jrc.ec.europa.eu/services/webview/jeoopp/databrowser/>



# Optimizing Sentinel-2 image selection in a Big Data Context- Running in Docker universe

- “Optimal” means 100% cloud free
- The selection was made from 2,128,556 optical remote sensing images.
- Each image was assigned to one core.
- Not dependency between jobs
- To overcome schedd limitations
  - MAX\_JOBS\_PER\_SUBMISSION 100K
  - DAG with 22 instances



Global S2 Quick look Mosaic [doi:10.1080/20964471.2017.1407489](https://doi.org/10.1080/20964471.2017.1407489)

# Mediterranean Sea simulation 1958-2013

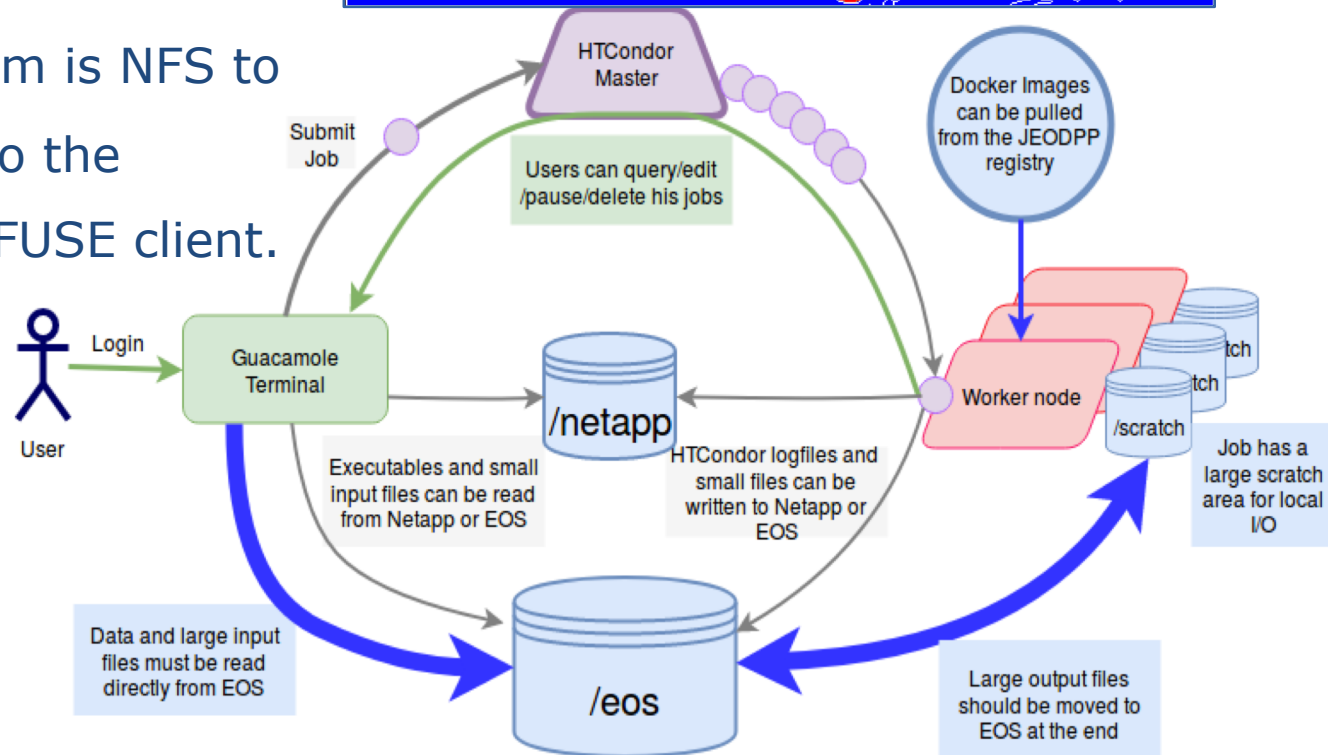
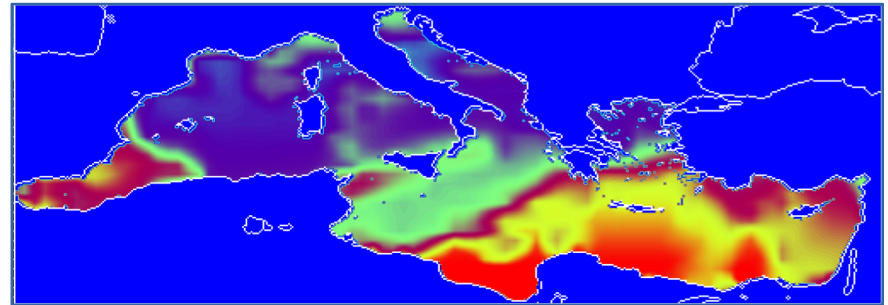
## Running in Parallel universe + Docker Swarm

- 50 years simulation over the Mediterranean sea.
- CERN EOS is used for the main storage, but in the MPI applications the backend processing file system is NFS to avoid problem due to the performance of the FUSE client.


See

[http://batchdocs.web.cern.ch/batchdocs/troubleshooting/eos\\_submission.html](http://batchdocs.web.cern.ch/batchdocs/troubleshooting/eos_submission.html)!

Hydrodynamic and ecosystem simulations  
<http://mcc.jrc.ec.europa.eu>



# Constraints in HTCondor

- A given user can only be mapped to \*one\* proc-user.
- Condor\_ssh\_to\_job not work in Docker universe. 
- How implement condor\_qsub with Docker universe?
- Implementing condor\_q with python binding is not straightforward.

See all jobs running in the cluster

```
In [3]: schedd = htcondor.Schedd()
jobs = []
for job in schedd.xquery():
    jobs.append(str(job.get('ClusterId')))
jobs = list(set(jobs))
print '\033[1m{:10s} {:15s} {:7s} {:7s} {:7s} {:8s} {:7s} {:10s} {:7s} {:7s} {:12s}'.format('USER', 'CMD', 'JOBID', 'DONE', 'IDLE', \
    'RUNNING', 'HOLDS', 'TOTAL', 'RAM(GB)', 'CPUS', 'DD/hh:mm:ss')+'\033[0m'

for j in jobs:
    IDLEJOBS=sum(1 for job in schedd.xquery(requirements = 'ClusterId =?= %d && JobStatus=?=1'%int(j)))
    RUNNING=sum(1 for job in schedd.xquery(requirements = 'ClusterId =?= %s && JobStatus=?=2' %int(j)))
    HOLDS=sum(1 for job in schedd.xquery(requirements = 'ClusterId =?= %s && JobStatus=?=5' %int(j)))
    CurrentCPUUsed = 0
    CurrentRAMUsed = 0
    users = []
    for job in schedd.xquery(requirements = 'ClusterId =?= %s' %int(j)):
        #print job
        arraysize=str(job.get('TotalSubmitProcs'))
        qdate=job.get('QDate')
        cmd=str(job.get('Cmd'))
        CurrentRAMUsed = int(job.get('RequestMemory'))
        CurrentCPUUsed = int(job.get('RequestCpus'))
        users.append(str(job.get('Owner')))

    dt1 = datetime.datetime.fromtimestamp(qdate) # 1973-11-29 22:33:09
    dt2 = datetime.datetime.fromtimestamp(int(time.time())) # 1977-06-07 23:44:50
    rd = dateutil.relativedelta.relativedelta(dt2, dt1)
    timeinq=int(time.time())-qdate
    CMD=base=os.path.basename(str(cmd))
    TOTAL=arraysize
    DONE=int(arraysize)-(int(IDLEJOBS)+int(RUNNING)+int(HOLDS))
    users = list(set(users))
    for u in users:
        print '{:10s} {:15s} {:7s} {:7s} {:7s} {:8s} {:7s} {:10s} {:7s} {:7s}'.format(str(u),str(CMD),str(j),\
            str(DONE),str(IDLEJOBS),str(RUNNING),str(HOLDS),str(TOTAL),str((CurrentRAMUsed*int(RUNNING)) / 1024),str(CurrentCPUUsed*int(RUNNING)))
```

USER	CMD	JOBID	DONE	IDLE	RUNNING	HOLDS	TOTAL	RAM(GB)	CPUS	DD/hh:mm:ss
kempepi	optmaskql.sh	2378		8725	0	0	3928	12653	0	6/3:36:37
klimeto	run_processing.	2426		397	375	228	0	1000	11400	228 0/5:33:9

# Open questions

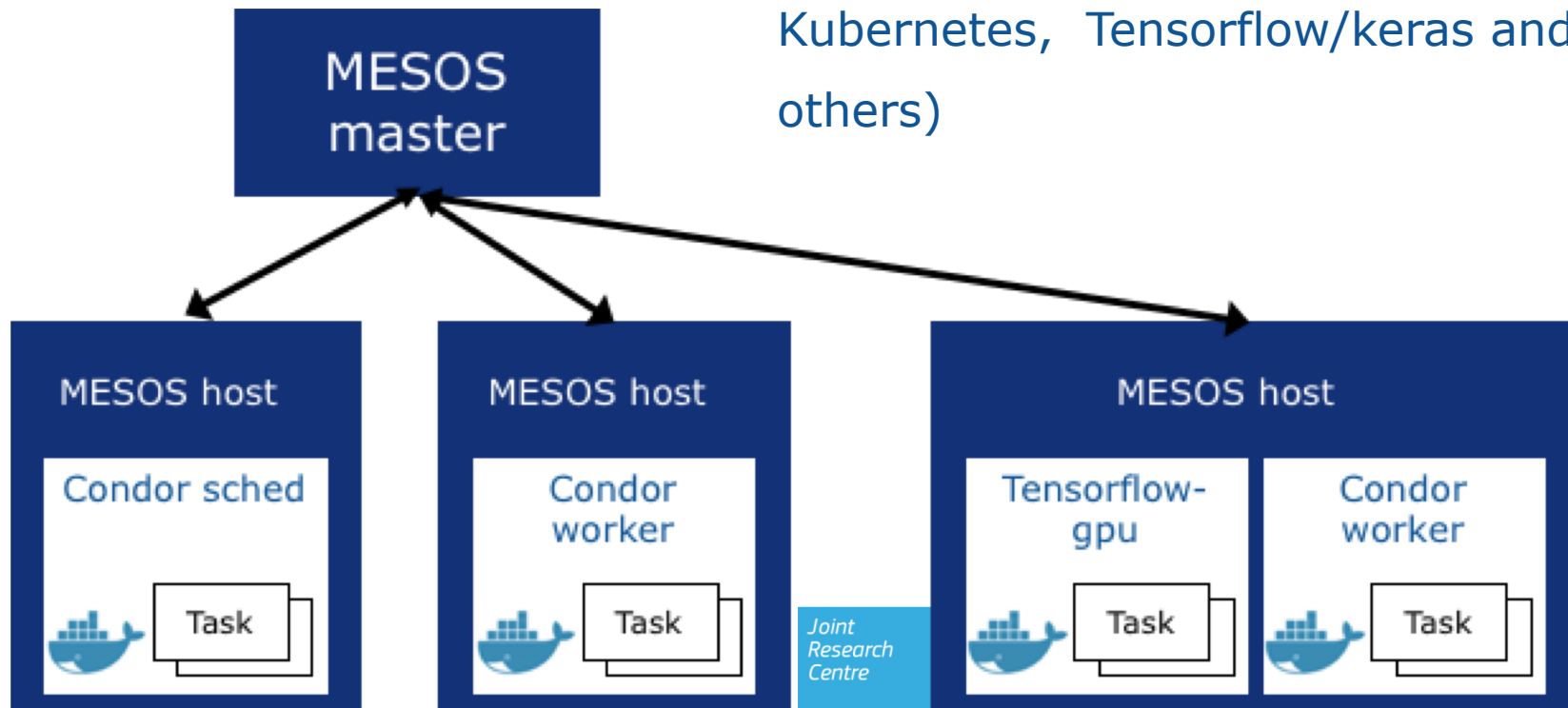


We are still working on finding solutions in the following areas:

- Docker checkpoint with HTCondor
- Allow condor\_tail to read standard error file.
- Submit job with simple dependency without specify the full graph dependency with DAGMan.
- How to suspend job in a host to allocate other jobs.

# Future - Batch System processing as service (BaaS)

- Auto-scalable service environment with MESOS, DOCKER, HTCONDOR
- MESOS allows framework integration with other services (HTCondor, Kubernetes, Tensorflow/keras and others)



# Conclusions



## Thank you

[Dario.Rodriguez@ec.europa.eu](mailto:Dario.Rodriguez@ec.europa.eu)