Joint Research Centre (JRC)

EO&SS@BigData pilot project

Text and Data Mining Unit
Directorate I: Competences

Flexible and Cost-effective Petabyte-Scale Architecture with HTCondor Processing and EOS Storage Backend for Earth Observation Applications

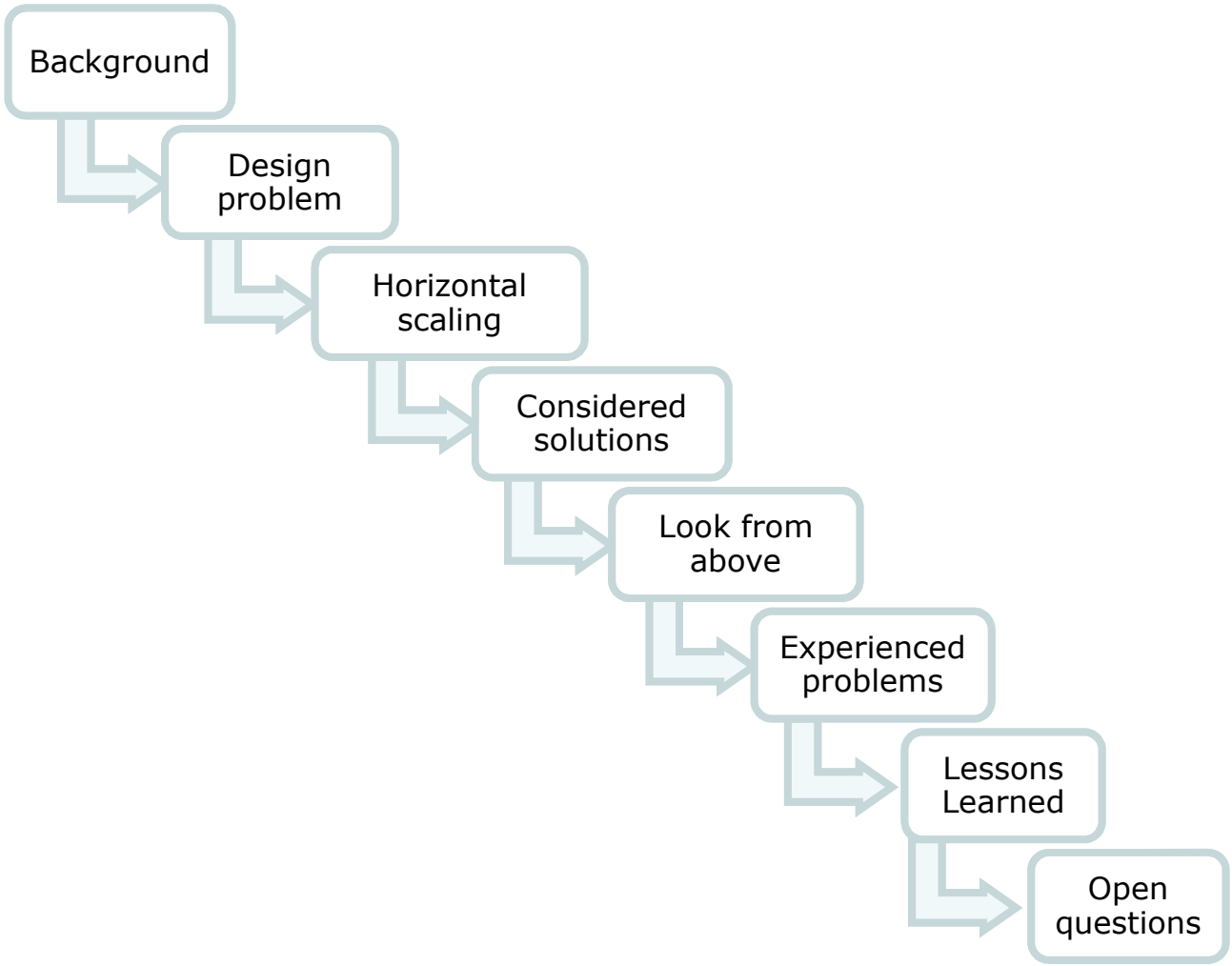
V. Vasilev, D. Rodriguez, A. Burger and P. Soille

Contacts: Pierre.Soille@jrc.ec.europa.eu
Armin.Burger@jrc.ec.europa.eu

Joint Research Centre

European HTCondor Workshop 2017

OUTLINE



Joint Research Centre

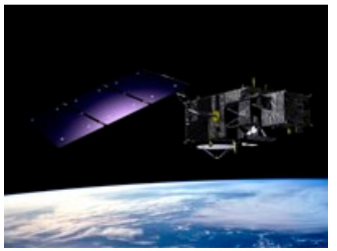
European Commission

Earth Observation & Social Sensing Big Data Pilot Project - Background

- Wide usage of Earth Observation (EO) data at JRC
- The EU ***Copernicus*** Programme with the ***Sentinel*** fleet of satellites acts as a game changer:
  - *expected 10TB/day of **free and open** data*
  - *Requires new approaches for data management and processing*
- JRC Pilot project launched in 2015
  Major goal: set up a central infrastructure:
  ⇒ *JRC Earth Observation Data Processing Platform (JEODPP)*

Sentinel-1 (Credits: ESA/P. Carril)

Sentinel-2 (Credits: ESA/P. Carril)



Sentinel-3 (Credits: ESA/J. Huart)

Joint Research Centre

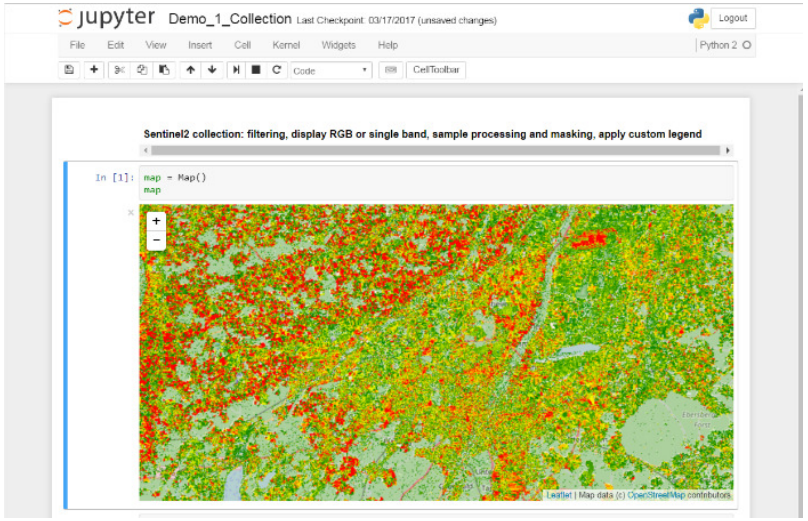
European Commission

Solving the design problem

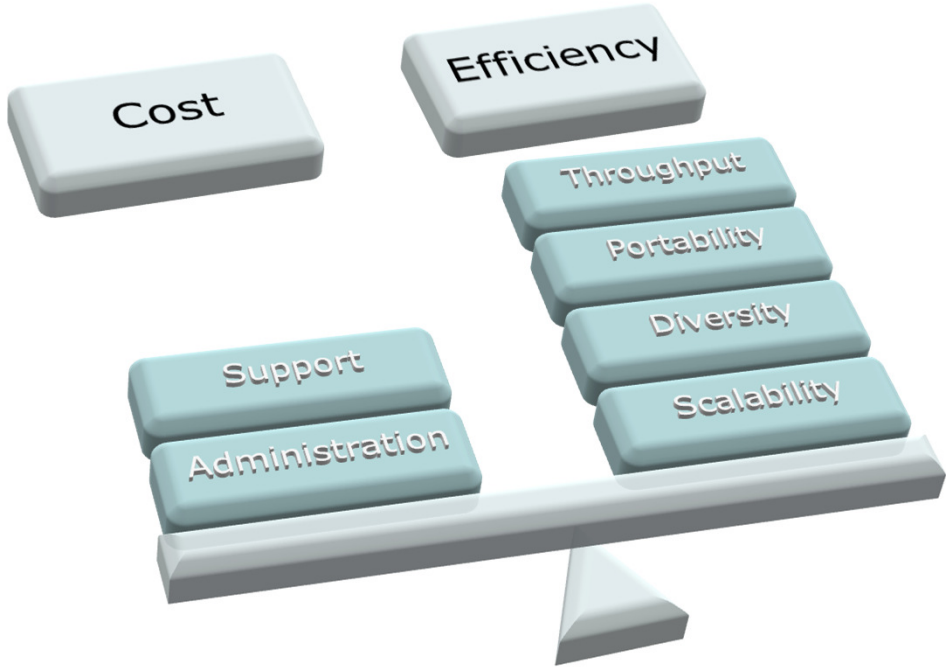
Common problem:

Investment vs Productivity and Flexibility

Interactive web processing



&



Batch cluster processing

```
                Machines Owner Claimed Unclaimed Matched Preempting  Drain

      X86_64/LINUX       37     0       0        37       0          0      0

             Total       37     0       0        37       0          0      0
```

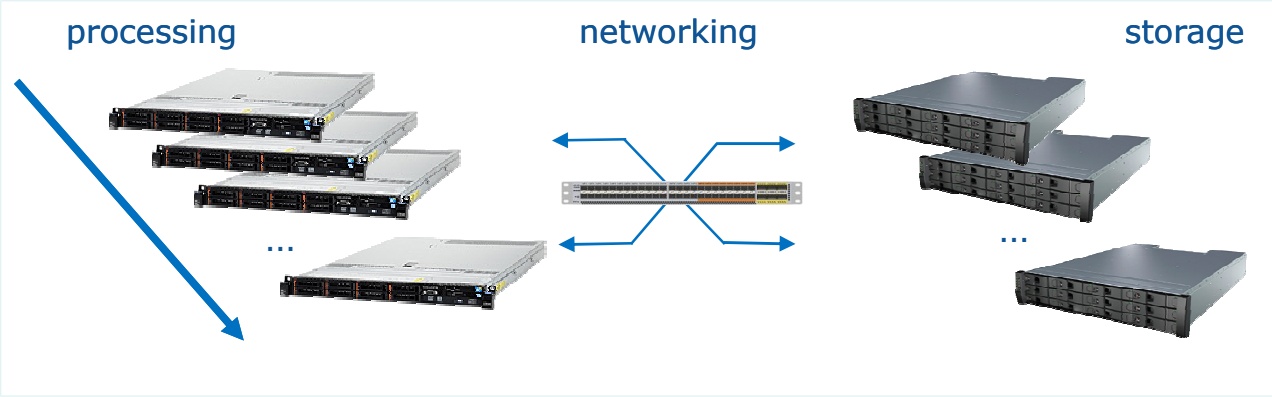
Joint Research Centre

European Commission

Horizontal scaling with commodity hardware

Solves:

- the hardware aspect of the problem



Comes at the price of:

- system administration and network usage

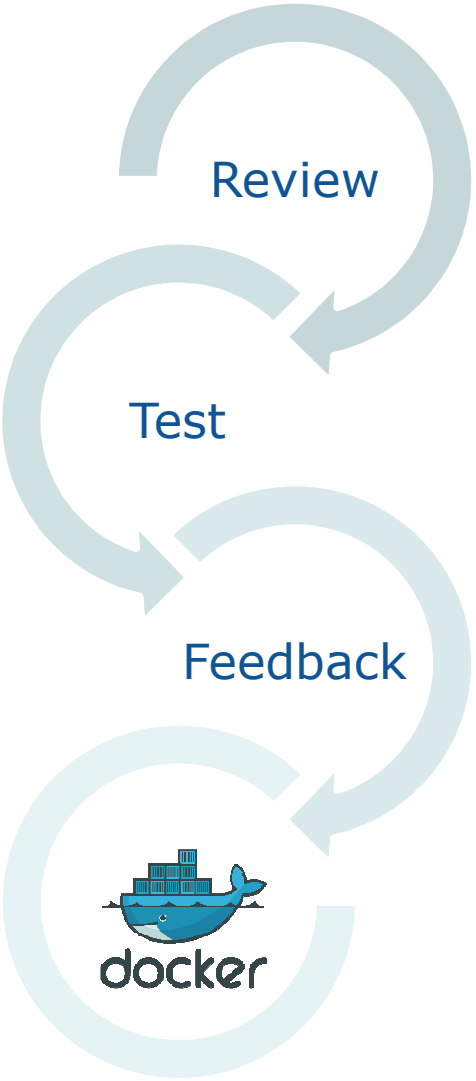
Requires flexible solutions for:

- processing and storage nodes aggregation

Joint Research Centre

European Commission

Libraries capsulation



Solves: libraries capsulation problem

- Prior infrastructure solutions solving the library capsulation problem were relying heavily on virtual machines
- Containerization as a “new” approach of tackling diversified library environments
- Followed a standard decision making path we:
  - reviewed several solutions;
  - tested less;
  - and finally decided to implement Docker as a key component of our batch and interactive setup

EOS storage

EOS is the CERN disk-only file storage, targeting physics analysis use cases

Solves: storage aggregation problem, sits on commodity hardware

- Supports multiple access protocols

  EOS primary protocol is XRootD
  it also supports HTTPS/WebDAV, S3, gridFTP, SRM, ROOT and FUSE (providing POSIX like mounts)

- Scalable and flexible infrastructure

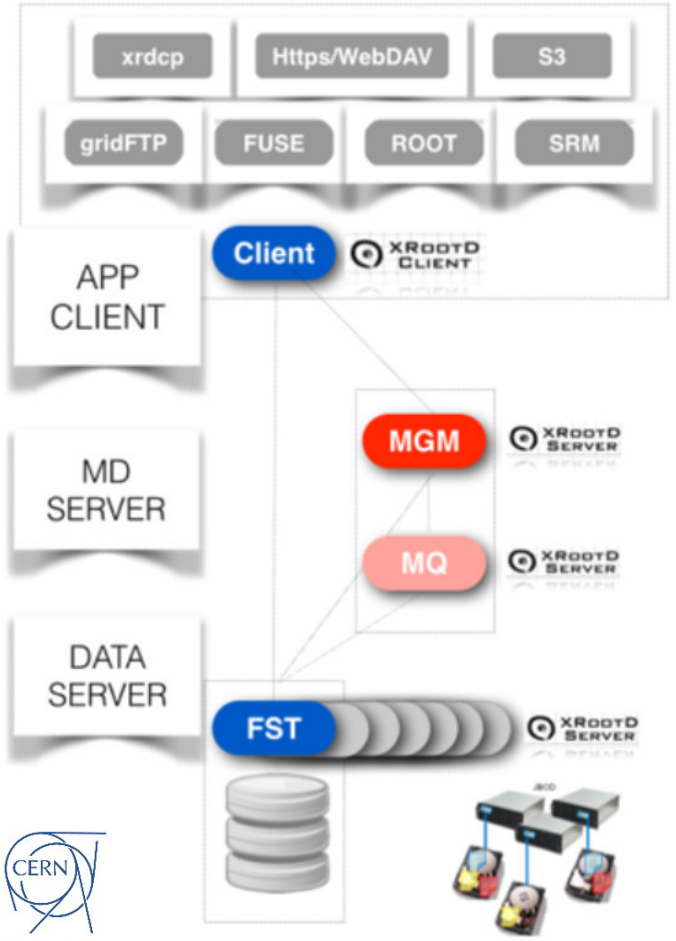
  Adding new file storage nodes (FST) is straight forward

- Strong security

  Kerberos, Unix, SSS

- Comes with many features

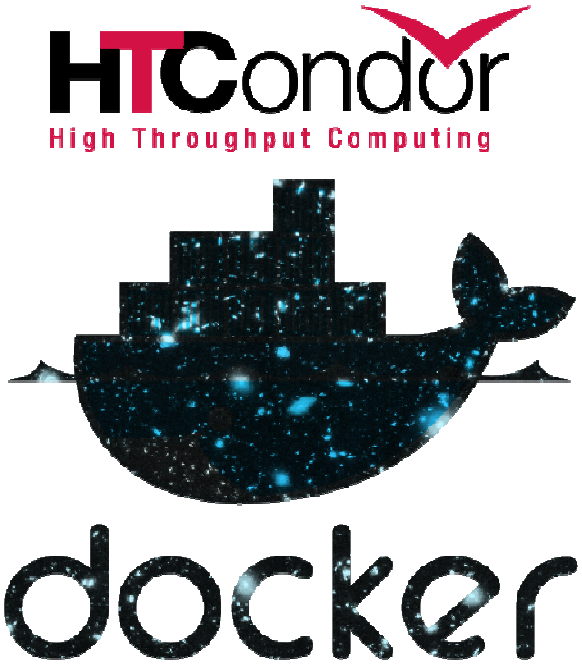
  Atomic upload, quotas, recycling, symbolic links, ACL's, subtree accounting and many others



Joint Research Centre

European Commission

HTCondor



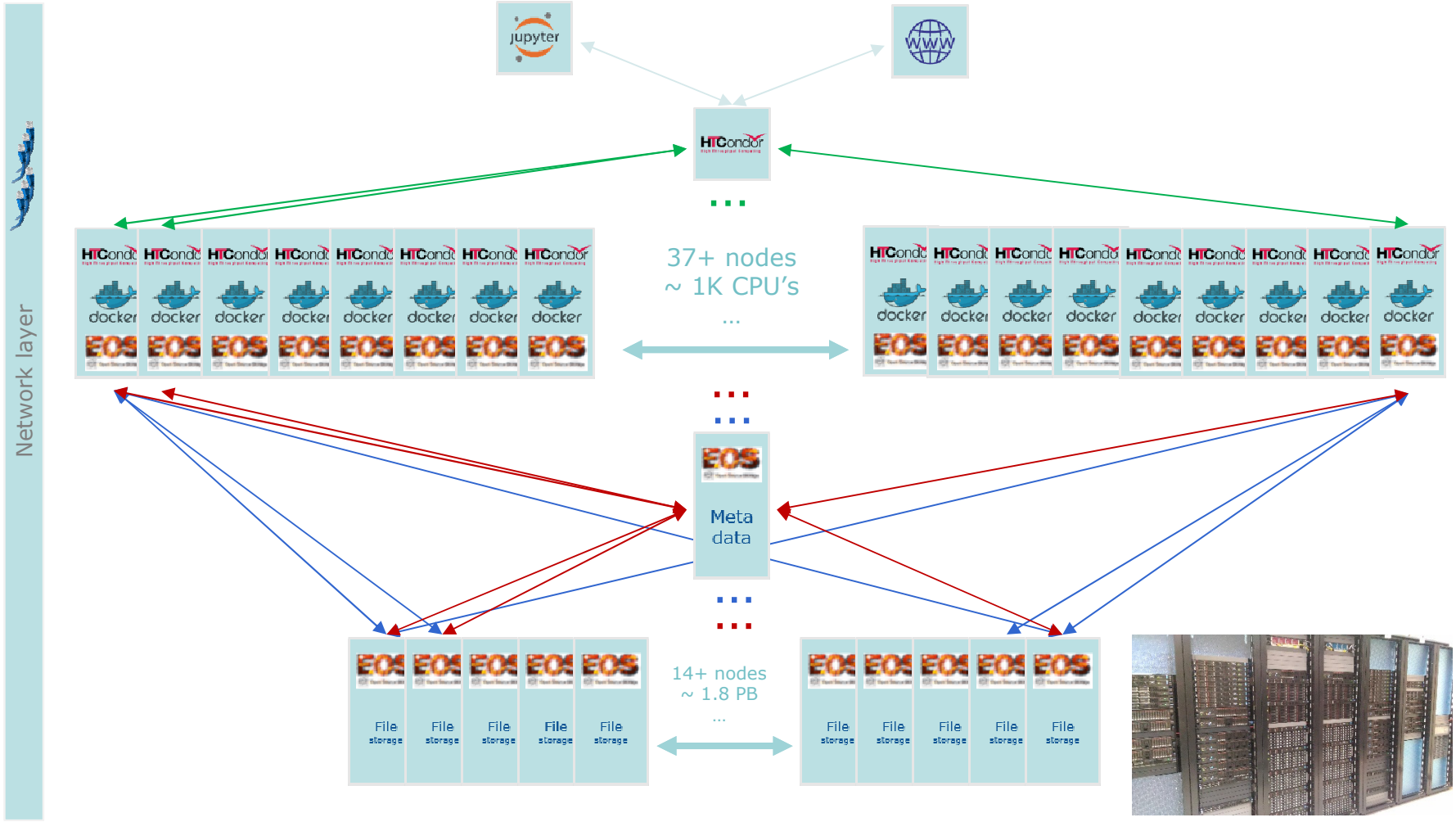
Solves: processing capacity aggregation

- Possesses maturity in cluster environment processing
- Is highly scalable
- Works among other with Docker Universe
- Is well documented and widely used
- Comes with a friendly development team and great support

Joint Research Centre

European Commission

Look from above



Joint Research Centre

European Commission

Architecture point of view

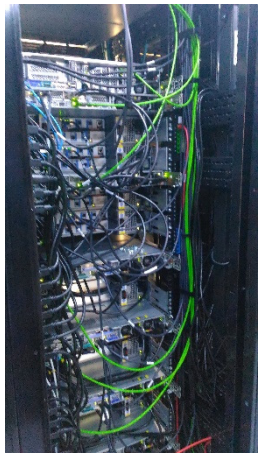
- Mission accomplished achieving the following
  - *Highly performant and scalable system thanks to:*
    - EOS storage
    - HTCondor
  - *No particular hardware requirements (controllers, size of disks, number of processors and so on)*
  - *Usage of commodity hardware, cheapest boxes are often the best*
  - *Client applications are encapsulated in Docker containers*
  - *Interactive processing based on custom developed libraries accessed through Jupyter and web services*
  - *Web submission of HTCondor jobs through Jupyter and HTCondor python bindings*
- However

Joint Research Centre

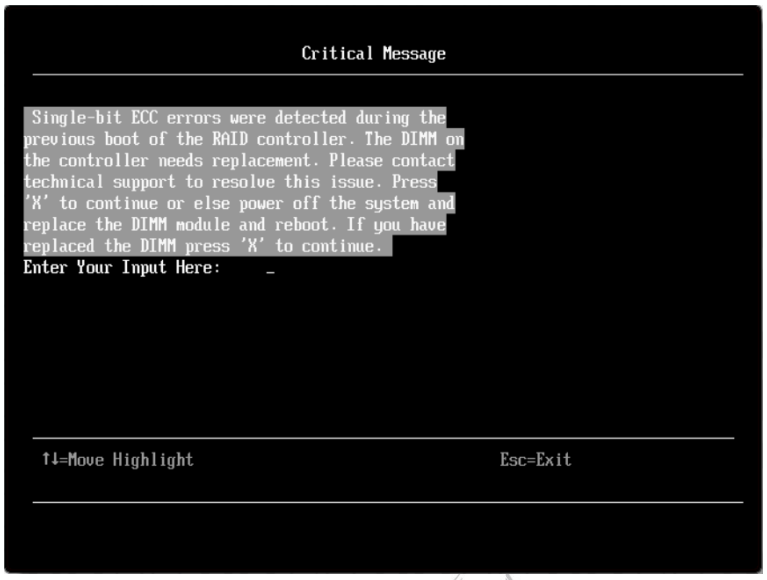
European Commission

Problems experienced: Hardware

- 10 GBit network cables disconnects
  - *We have experienced occasional network cable disconnects*
  - *As conclusion in cluster environment one needs to carefully choose network equipment providers and product characteristics*



- Hardware failures
  - *Example: RAID controllers on processing nodes (feels good to have a non RAID dependent file system)*
  - *Time consuming to solve*
  - *Difficult to prevent*



Problems experienced: Storage

- EOS storage fuse client stability
  - *(miss)configuration of the XRootD layer in regards to cluster processing*
  - *Fuse mounts are dependent on a process id ("Transport end point not connected")*
  - *EOS fuse client maintenance*
    - monitoring of clients health and memory consumption
    - auto-mounts could be configured where feasible
    - scheduled daemon restarts
    - future releases expected to decrease maintenance role
  - *manipulation of large amounts of small files bears communication overhead*

EOS storage client settings fitting HTCondor cluster environment:
placed in /etc/sysconfig/eos

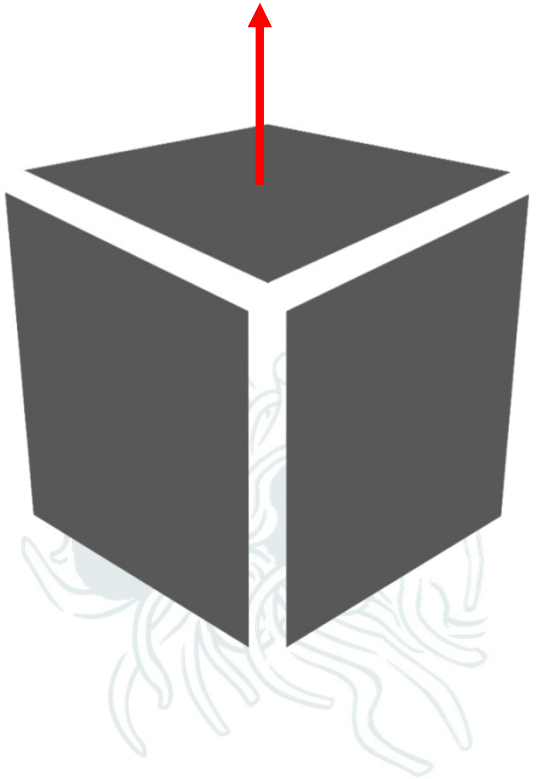
```
#Disable parallel IO where it is not needed
export EOS_FUSE_NOPIO=1
#Workers should be ~ nproc
export FUSE_N_WORKER=40
#Increase stream and request timeouts
export XRD_REQUESTTIMEOUT=300
export XRD_STREAMERRORWINDOW=300
export XRD_STREAMTIMEOUT=300
#Set XRootD copy timeout to one hour
#Default is 10min on busy system it seems low
export XRD_CPTPCTIMEOUT=3600
```

EOS storage file storage node recommended settings:

```
#Increase FST stream timeout
export EOS_FST_STREAM_TIMEOUT=900
#Specify interface name (only important for MGM statistics)
export EOS_FST_NETWORK_INTERFACE=bond0
```

Problems experienced: User Applications

- Black box user applications:
  - *could lack perceptions of cluster processing in big data environment;*
  - *one can easily discover "find", "list" and other filesystem commands than can cause trouble;*
  - *excessive script verbosity fills up disks with charm;*
- Can lead to MetaData server flooding
  - *MetaData server scaling (there is already a new EOS release under development that implements it)*

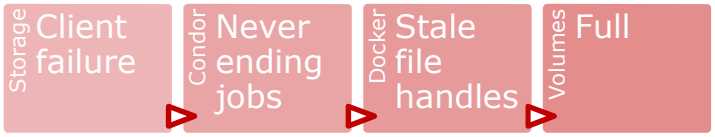


Problems experienced: Volumes

- Processing hosts root volume maintenance
  - *default Docker and HTCondor outputs are by default stored in the directory /var/lib;*
  - *even when mapped to a separate mount point script verbosity can fill up available space;*

Example – move HTCondor and Docker to separate volume:
(assuming your volume is mounted under /opt)

```
systemctl stop docker condor
mkdir -p /opt/lib/docker/
mkdir -p /opt/lib/condor/
rsync -rogtv /var/lib/docker/ /opt/lib/docker/
rsync -rogtv /var/lib/condor/ /opt/lib/condor/
mv /var/lib/condor/ /var/lib/condor.old/
mv /var/lib/docker/ /var/lib/docker.old/
ln -s /opt/lib/condor/ /var/lib/condor
ln -s /opt/lib/docker/ /var/lib/dockers
systemctl start docker condor
```



Some useful info and cleanup comds:

Remove exited containers:
docker rm $(docker ps -aq --filter status=exited)

Docker thinpools space availability useful cmds:
docker info
docker system df
docker system prune
pvs

Search for stale file handles:
lsof -nP | grep '(deleted)'

- Processing hosts Docker volume management
  - *if no post processing maintenance – easily becomes a problem;*
  - *system malfunctions sometimes also leads to volume problems;*

Joint Research Centre

European Commission

Problems experienced: HTCondor

- Spread of Docker images is done manually
- No practical way to populate credentials:
  - *When a job is submitted the submitter Kerberos ticket has to be manually populated on the processing hosts;*
  - *One solution could be the use of common shared space for user credentials.*
- Starter log contains stacktraces, which are sometimes difficult to debug
- Docker jobs are run in sequential mode, heavier images could cause run delays resulting in unused capacity

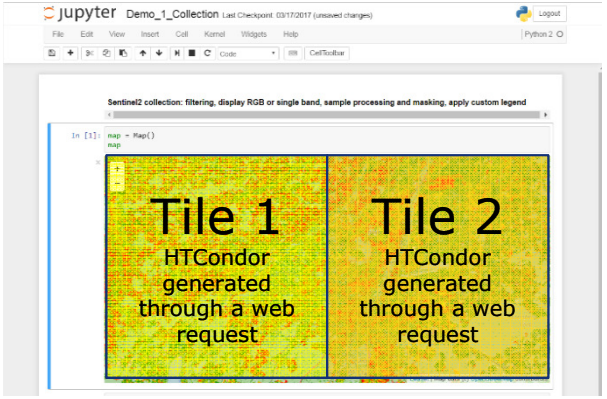
Lessons learned?

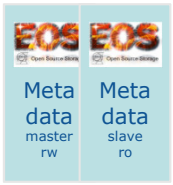
- Pay excessive attention to hardware configuration. From hardware point of view smaller nodes (1U) seem to be more robust in comparison to larger ones (2U)
- Write code with care, train your users
- Foresee “big” scratch space for your cluster nodes, place /var/lib/docker/ and /var/lib/condor in separate volume/s, unforeseen errors could fill up space with charm
- Configure Docker thinpool volume to self expand?
- Set your processing chains perform maintenance of unused containers and images
- Perform storage connection maintenance when feasible

Open questions

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

- *Submission of web processing requests through HTCondor or with its informative agreement*
- *Simultaneous submission of jobs in processing nodes*
- *Decoupling of processing from pure web services (through read only Meta Data server)*
- *Move towards singularity instead of Docker where applicable, are there benefits?*

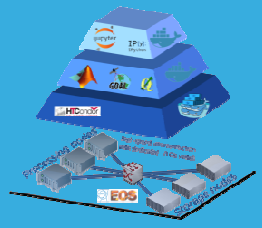




Joint Research Centre

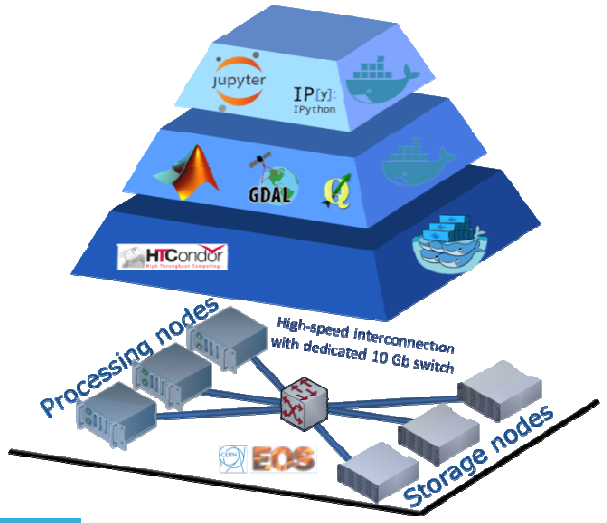
European Commission

Thank you all



EO&SS@BigData pilot project

Text and Data Mining Unit
Directorate I: Competences



Joint Research Centre

European Commission