Experiences with HTCondor Universes on a Petabyte Scale Platform for Earth Observation Data Processing

D. Rodriguez, V. Vasilev, and P. Soille
OUTLINE

• JRC Earth Observation Data and Processing Platform
• Batch processing
• HTCondor universes
• Use-cases and achievements
  • *Global Human Settlement Layer*
  • *Search for Unidentified Marine Objects*
  • *Marine Ecosystem Modelling*
• HTCondor python binding
• Lessons learned and open questions
• 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)
**Versatile** platform bringing the users to the data and allowing for

- Running large scale batch processing of existing scientific workflows thanks to lightweight virtualisation based on Docker orchestrated by HTCondor
- Remote desktop capability for fast prototyping in legacy environments
- Interactive data visualisation and analysis with Jupyter
Main focus on satellite image data
Support for existing processing workflows and environments (C/C++, Python, Matlab, Java)
Development of processing API for EO data
  * in Python, based on C/C++ modules
  * used for
    Low-level batch processing
    High-level interactive processing
Components of the **JEODPP**
Software & tools require different computing environments

[from 2015 JRC survey]
JEODPP HTCondor setup

**HTCondor** workload manager

- Central and submit machine on same node:
  - Simplify Management
  - Maximize computing power capability - *Each CPU is important*

- Processing cluster of commodity hardware
  - Currently ~ 1000 CPU cores, ~15 TB RAM
  - 10 Gb/s connectivity, dedicated switches

- Using **Docker** containers for executing jobs
  - Flexible management of processing environments
Workload manager - HTCondor

Great variety of applications need a great flexibility

**Vanilla universe**: allow running almost any single core apps.

**Docker universe**: a container instance is the HTCondor job (job sandboxing).

**Parallel universe**: when multiple processes of a single job must be running at the same time on different machines.

- Global Human Settlement Layer
- MARitime SECurity
- SEACOAST
  Hydrodynamic and ecosystem simulations
Global Human Settlement Layer
Running in Vanilla and Docker universe

- Pre-processing of the 5,026 products resulting in a volume of ~19TB;
- Produced data (S1 GHSL output): ~4TB;
- Duration of batch processing: ~12 h with ~240 concurrent jobs.
- Single core application based on MATLAB
- Several containers by host are running (small containers)
- More than one container by host.

Dynamic or Static slots can work.
Search for Unidentified Maritime Objects (SUMO) Running in Docker universe

- Processing of those 11,451 Sentinel-1 products resulting in a volume of ~7.5GB; 480 concurrent jobs at a time; whole process completed in 1.30h
- Extraction of S1 metainfo (3.4GB) with MATLAB script; 390 concurrent jobs at a time; whole process completed in 1h.
- Multi-core core application.
- Large containers
- No communication between containers.
• OpenMPI application based on FORTRAN
• Model grid execution divided in 64 domains for 10km res. of the Mediterranean sea
• Workflow executed in two steps:
  • One job triggers the startup of a virtual HPC cluster based on Docker containers
  • Job run the working script on the virtual HPC Cluster by using OpenMPI

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

Based on S. Yakubov et al.
Ongoing work – Condor Python binding

What user can do?

- Trigger a dedicated container in order to run HTCondor notebook
- Import classad & htcondor classes
- Edit job requirements
- Submit jobs
- Monitor jobs and queue status
Problems experienced

- Scheed limits. Maximum number by submission 20K mainly due to memory usage and disk transactions.
  - Job are forced to run a stack of inputs (chunk).
  - If one job fails all the stack should be re-process again.
Problems experienced

- Job enter in a “Vicious circle”. Example, Host cannot start docker container but still are accepting jobs from the same job cluster.
Problems experienced

- Job enter in a “Vicious circle”. Example, Host cannot start docker container but still are accepting jobs from the same job cluster.
Problems experienced

- Job enter in a “Vicious circle”. Example, Host cannot start docker container but still are accepting jobs from the same job cluster.

```
[rodrird@s-jrciprjeop16lp docker4]$ condor_qedit 355 requirements 'Machine != "s-jrciprjeop200p.cidsn.jrc.it"'
Set attribute "requirements" for 2262 matching jobs.
[rodrird@s-jrciprjeop16lp docker4]$ condor_q

--- Schedd: s-jrciprjeop16lp.cidsn.jrc.it : <139.191.240.161:96187... @ 06/07/17 14:38:29
OWNER   BATCH_NAME        SUBMITTED  DONE  RUN  IDLE  HOLD  TOTAL  JOB_ID
rodrird CMD: start_process.sh 6/7 14:27 1 858 554 850 2263 355.0-2262

2262 jobs; 0 completed, 0 removed, 554 idle, 858 running, 850 held, 0 suspended

[rodrird@s-jrciprjeop16lp docker4]$ condor_release 355
All jobs in cluster 355 have been released
[rodrird@s-jrciprjeop16lp docker4]$ condor_q

--- Schedd: s-jrciprjeop16lp.cidsn.jrc.it : <139.191.240.161:96187... @ 06/07/17 14:39:04
OWNER   BATCH_NAME        SUBMITTED  DONE  RUN  IDLE  TOTAL  JOB_ID
rodrird CMD: start_process.sh 6/7 14:27 1 908 1354 2263 355.0-2262

2262 jobs; 0 completed, 0 removed, 1354 idle, 908 running, 0 held, 0 suspended
```
Problems experienced

- Condor “Docker universe” does not force multithread apps run on a single core
Problems experienced

- Condor “Docker universe” does not force multithread apps run on a single core

If we are lucky we can set the environment variable (ej, OPEN_NUM_THREAD, GDAL_NUM_THREAD)
Lessons learned?

- Increase load by job improve the overall performance.
- Keep everything locally, R/W directly in the share file system can drastically decrease the overall performance.
  - `should_transfer_files = YES`
- Must be organize with logs
  - `/submit_machine/local/job$(Cluster)_(process)`
- Set time limit for job execution (job can stay in running status forever).
- Foresee big scratch space for your cluster nodes, place `/var/lib/docker/` and `/var/lib/condor` in separate volume, some unforeseen errors could fill up space with charm
- Parallel shell is a good alternative for `condor_config`. (to stop, start and reconfigure not just condor also dockers etc)
Open questions

It is possible to add universe?
For example a combination of docker universe with parallel universe

Universe: PVM  #Parallel Virtual Machine
Thank you all

EO&SS@BigData pilot project

Text and Data Mining Unit
Directorate I: Competences