Python binding based monitoring at CMS

Diego Davila, on behalf of the Submission Infrastructure team of CMS
May, 2019
Outline

- The CMS experiment
- The CMS Global pool
- Monitoring the pools
- Pretty plots
- Conclusions
The CMS experiment

- CMS - Compact Muon Solenoid
- One of the main experiments in the LHC (Large Hadron Collider)
- Collaboration of more than 5k people
The CMS experiment

2942 PHYSICISTS
1065 ENGINEERS
281 TECHNICIANS
229 INSTITUTES
51 COUNTRIES & REGIONS
The CMS experiment

Particles are injected into the LHC and accelerated using magnets
Bunches of particles circle around the LHC in opposite directions on separated tubes
After reaching certain amount of speed (close to the speed of light), the particles are positioned so that they will crash to each other inside the different detectors.
Collisions are recorded, using the many sensors inside each detector

This produces a lot of data to be analyzed
The CMS experiment

The institutes within the collaboration provide computing resources to store and analyze the data.
Not only 1 but 5 pools

Global pool

CERN pool

Volunteer, ITB and ITBDev

5  Central Managers
7  Negotiators
76  Schedds
134k Slots
300k Cores
What we wanted

● A monitor to watch them all
● Be able to reuse the monitoring infrastructure provided by CERN
  ○ short and long term storage
  ○ auth
  ○ pretty plots software (grafana & kibana)
● Written in a friendly language (python)
● and most importantly….
What we did

● For each classAd, create a list of attributes we were interested in
  ○ How many attributes you can find in the Negotiator classAd?

    [ddavila@vocms0804 ~]$ condor_status -negotiator -l | wc
    391 1220 17030

● Wrote < 300 lines of python using the HTCondor python bindings
● Used a cronjob to execute the script for each of our pools
A very simple example

```python
import htcondor

collector = htcondor.Collector("collector-hostname.domain")

projection=["Name", "Disk", "Memory"]
condor_ad_type = htcondor.AdTypes.Startd
constraint="Cpus > 0"

#SELECT Name, Disk Memory FROM Startd WHERE Cpus > 0

ads = collector.query(condor_ad_type, constraint, projection)

for ad in ads:
    print ad
```
What you get

A list of these:

```
[
  Disk = 214761;
  Memory = 2048;
  Name = "slot1_10@glidein_64_218192637@b6515245b4.cern.ch"
]
```

- We use a home made function to transform into JSON
- Since 8.8.2 the ClassAd module has a new printJson()
Some pretty plots

Number of cores in use by job type
Some pretty plots

Number of cores grouped by slot size

CPU cores for jobs by job core size

Some pretty plots

The 10 schedds with the highest duty cycle
Some pretty plots

Length of each negotiation phase
Conclusions

- There is a lot of data you can get from condor
- Using the HTCondor python bindings you can collect these data, in a easy and friendly way.
- The most difficult part is to decide what to monitor
- Most of the attributes in the classAds are well documented
Thank you for listening!