Connecting restricted, high-availability, or low-latency resources to a seamless Global Pool for CMS

Kenyi Hurtado Anampa[10], Justas Balcas[3], Brian Bockelman[9], Jadir Marra Da Silva[7], Dirk Hufnagel[5], Bo Jayatilaka[5], Farrukh Aftab Khan[6], Krista Larson[5], James Letts[8], Marco Mascheroni[5], David Mason[5], Ajit Kumar Mohapatra[11], Stefan Piperov[1], Anthony Tiradani[5], Antonio Perez-Calero Yzquierdo[4], Vassil Verguilov[2]

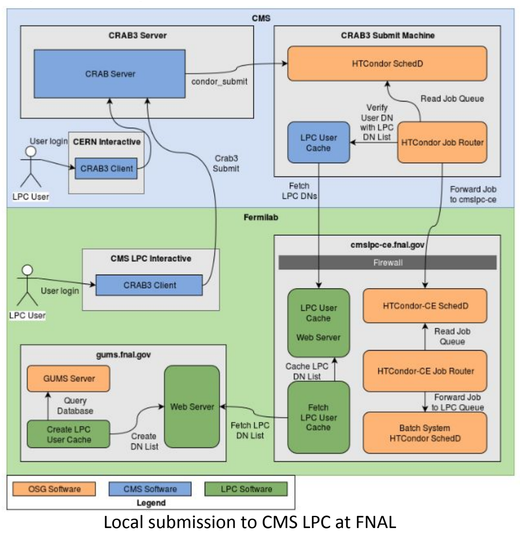
[1] Brown University (US)
[2] Bulgarian Academy of Sciences (BG)
[3] Caltech (US)
[4] CIEMAT (ES)
[5] Fermilab (US)
[6] National Centre for Physics (PK)
[7] Universidade Estadual Paulista (BR)
[8] University of California, San Diego (US)
[9] University of Nebraska (US)
[10] University of Notre Dame (US)
[11] University of Wisconsin-Madison (US)

Connecting diverse and sometimes non-Grid enabled resource types to the CMS Global Pool has been a major goal of CMS and a challenge. This work includes not only adding Central Facilities or Opportunistic resources to the Global Pool, but also managing prioritization to local users of beyond WLCG pledged resources at CMS sites and mechanisms to access these resources through local Institutional schedulers.

Connecting different Resource Types

FNAL LPC

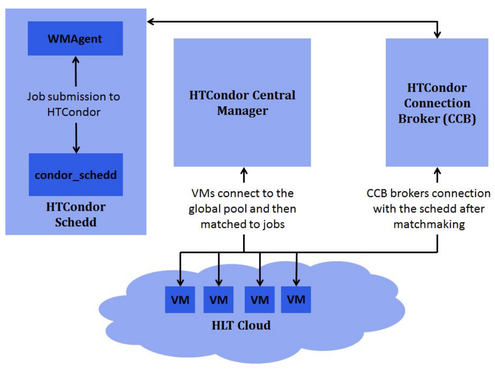
- The LHC Physics Center (LPC) is a regional center at Fermilab funded by the U.S CMS Operations Program to provide computing resources for analysis to its members and it has a local submission interface.
- The CMS Remote Crab Analysis Builder (CRAB) is the standard python-based CMS tool to submit jobs to the grid. The current framework relies on a server-client model, where the user interacts with the client and the server is in charge of submitting remotely to the CMS Global Pool, meaning local submission from the client is not possible.
- **Since CRAB3 does not support local submission and the LPC submission interface cannot be exposed globally, enabling CRAB3 submission at the LPC is not straightforward.**
- In order to solve this problem, LPC user jobs are routed from the CRAB3 grid schedulers to the LPC using a submission interface set up specifically to accept CRAB3 jobs.
- An HTCondor CE (referred as "CMS LPC CE"), sitting behind a firewall to only allow CRAB 3 scheduler connections (any other inbound traffic is denied by default), accept these jobs and route them to the local LPC batch system (HTCondor).
- The CMS LPC CE also periodically fetches the list of LPC users from the GUMS server. An additional process runs on the GUMS server to query the DNs of all LPC users and write the result to a file. The node running the GUMS server already runs a web server. The file containing DNs of the LPC users is made available on this web server, from where it is retrieved by CMS LPC CE. GUMS web server is not supposed to be exposed externally, so this additional hop at the CMS LPC CE is needed.
- Job submission up until the submission to CRAB3 schedulers works just as it would for another grid site. On the schedulers (or schedds), there is an HTCondor job router daemon running.
- The job router running on the scheduler fetches and caches the DNs of LPC users. The job router routes a job to CMS LPC at FNAL if it is submitted by an LPC user and the job whitelists 'T3_US_FNALLPC' as its desired Site to run on. All other jobs on the scheduler are ignored. As a safety measure, the HTCondor CE is also configured to only accept jobs from LPC users. A non LPC user job that gets routed to the CE by mistake is rejected.



Local submission to CMS LPC at FNAL

CMS HLT Integration

- CMS employs a multi level triggering system to collect data from the detector on the LHC. The highest level of the trigger system (known as the High Level Trigger or HLT) is a large compute farm of over **15,000 core**s and is completely owned and managed by CMS.
- Given the size of the resource, it was decided to use the compute resources during LHC fills when they are completely idle. Therefore, CMS overlaid an OpenStack infrastructure over the HLT resources during the first long shutdown of the LHC.
- The use of HLT as a general purpose compute resource is intended to never interfere with its primary purpose. The compute farm is still primarily being utilized for its triggering role during data acquisition. It is only during the idle cycles that the cloud infrastructure becomes active and used as a general purpose resource like normal grid sites.
- Workflow management agent (or WMAgent) is a python based agent used to handle submissions to the HTCondor scheduler daemon (condor_schedd). WMAgent is responsible for collecting, building and submitting tasks as well as monitoring their status.
- HLT cloud is registered as "T2_CH_CERN_HLT" in the CMS site database (siteDB). WMAgent uses this site name for submission. All the relevant HTCondor configurations are part of the images VMs used by the HLT.
- During their instantiation, the VMs are passed a service certificate. This service certificate is then used by the VMs to authenticate with the global pool. Once the VMs show up in the pool, they are matched to jobs whitelisting 'T2_CH_CERN_HLT' by the negotiator running on the central manager.
- Once a match has been made, HTCondor connection broker establishes a connection between the scheduler and the VM. A block diagram of integration has been shown in figure diagram at the bottom.
- When the HLT needs to be used as a trigger, the VMs are suspended to disk. The running jobs are then configured to wait 24 hours for the VMs to come back up. This allows jobs to resume from they had left off and increases throughput.
- The HLT VMs also connect to schedulers and back up central infrastructure at FNAL, allowing them to make use of the HA capability of the CMS Global pool.



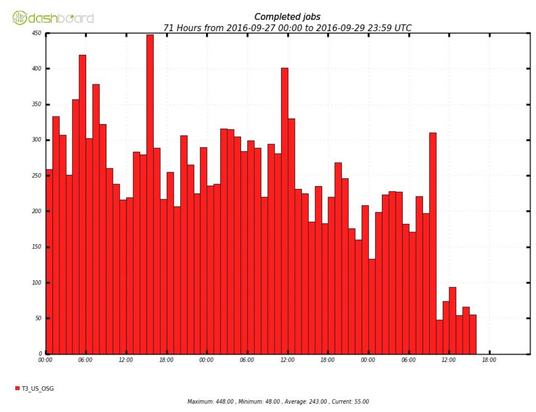
HLT integration into the Global Pool

Opportunistic Resources OSG Resources

- CMS makes use of many over the pledge compute resources opportunistically in the global pool. Most of this comes from grid sites already part of the CMS collaboration.
- OSG and CMS work in close collaboration with each other and an effort was made to explore the possibility of using OSG VO resources for CMS general purpose computing. To that end a special site name 'T3_US_OSG' was registered in the CMS site database.
- 'T3_US_OSG' contains many compute elements willing to run CMS jobs opportunistically. Most sites under the umbrella of T3_US_OSG support either VO ATLAS or VO OSG. These VOs have almost identical requirements from their worker nodes as CMS making it easier for CMS to make use of them and not put any additional operational overhead on the sites.
- CMS identifies sites in OSG glideinWMS factories using a classAd named 'GLIDEIN_CMSSite'. This classAd was set on all CEs which can be opportunistically utilized by CMS. The CMS VO specific glideinWMS frontend then picks up all entries marked with 'T3_US_OSG' GLIDEIN_CMSSite name.
- Currently, T3_US_OSG contains the following resources:

T3_US_OSG → Resource Names
- AGLT2 (U. Michigan)
- Clemson-Palmetto (Clemson U.)
- MWT2 (U. Chicago, U. Illinois, Indiana U.)
- SU-OG-CE / SU-OG-CE1 (Syracuse U.)

- These resources are used for production jobs. CMS cannot utilize local storage at these sites and hence can only run GEN-SIM workflows. Stage out for these workflows is done to FNAL where all the subsequent steps which process staged output then execute at FNAL. The following dashboard report shows production jobs completed on this resource.



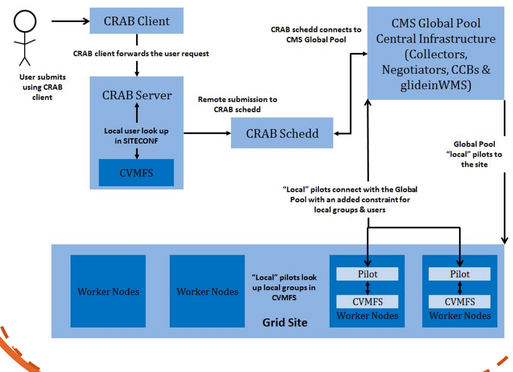
Dashboard reporting on September 27 to 29, 2016 for T3_US_OSG

High Priority Access for a restricted User Community

PRIORITY BEYOND WLCG PLEDGES

CMS Grid Sites pledge resources for the experiment to use and the prioritization of jobs is handled by the CMS central infrastructure so all CMS users are treated equally. But what happens when a Site wants to give priority to their local physics group on top of their pledge? The following diagram explains how a local Site can make use of their dedicated over the pledge resources.

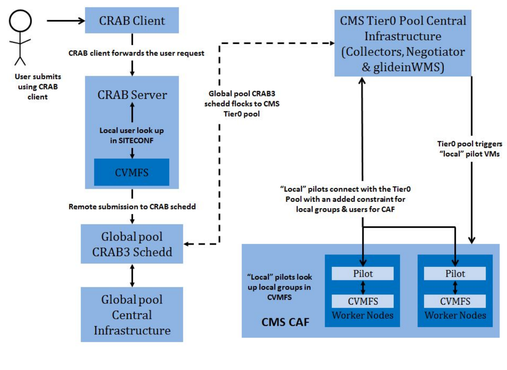
A list of users (or VOMS groups, e.g: uscms, decms, itcms, etc) is published by a site admin via CVMFS/SITECONF. When the user submits a CRAB job, the CRAB server will look at CVMFS for the user/group list to see which ones are "local" and will create a condor classAd named "CMS Groups". The Global Pool central infrastructure then look up for this information and submits a special glidein with VOMS FQAN=*'/cms/local/Role=pilot'* which the site can then use to give these pilots a higher priority.



HIGH AVAILABILITY USAGE
CERN CAF

The CMS CERN Analysis Facility (CAF) is dedicated to latency-critical activities like detector calibration and alignment, detector and trigger commissioning, and very high priority physics analyses. Connected to the Global Pool and accessible with CRAB3, the modern analysis middleware of CMS, the CAF resource is restricted to a small number of users. Similar in structure to the Tier-0 facility on OpenStack, this set of resources is available on demand with very low latency and replaced a similar resource on LSF at CERN.

OpenStack VMs are spawned by the GlideinWMS factory. These are connected to the CMS Tier 0 and global pool CRAB3 schedds flock jobs to it. The CAF activity can be bursting and have long idle periods due to its high availability and low-latency nature, so connecting them to the T0 rather than directly prevents having idle cores in the pool interfere with pilot provisioning at the grid sites.



RESTRICTED USAGE
TEXAS A&M

There are some use cases where a local administrator might want to start a glidein against a particular local user's personal account. This brings an additional constraint of not having multiple users share the glidein. Only the specified user should be able to make use of the glidein.
In such cases, a site administrator is required to set an environment variable "USER_DN" with the DN of user whose personal account is being to run the glidein.

During the glidein startup process, a local glidein will look for this environment variable. If the variable is defined, an additional constraint is added to the START expression to only allow user with this particular DN to run jobs on this glidein. This has been successfully implemented and tested using the global pool at Texas A&M University (TAMU) in the U.S.

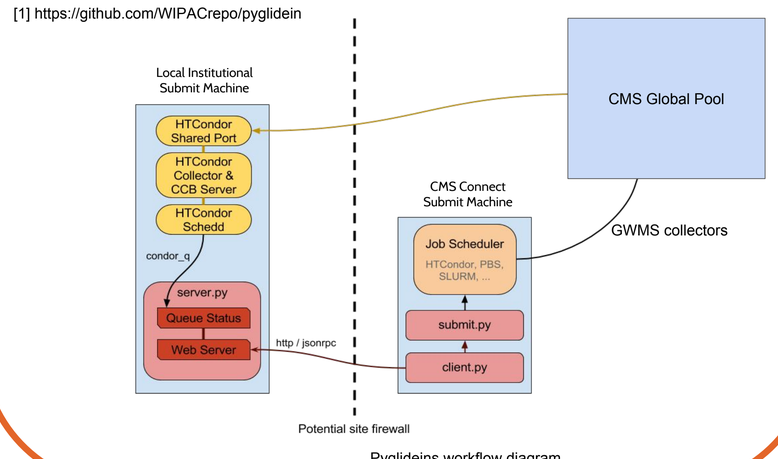
Connecting Institutional SCHEDDs to the Global Pool in a manageable way

While CRAB handles grid job submission for analysis jobs depending on the CMS framework executable (cmsRun), late-stage analysis tasks often are independent and batch systems are more suitable to handle these type of jobs. CMS Connect is used as the submission interface for condor-like jobs to the CMS Global Pool, but sometimes Institutional Pools also provide non-grid enabled local resources to their users for these tasks and using them both is the ideal.

However, allowing many Institutional schedulers to access the Global Pool can bring management problems as user accounting, job reporting and job control measurements to protect the Global pool have to be considered, so having a central submission interface these schedulers can talk to as a gateway before going to the Global Pool is important. In order to do this, glideins are submitted by the users via a python-based glidein package (pyglidein) to CMS Connect so that these can talk to their Institutional submission scheduler to overflow jobs into the Global Pool.

We use pyglideins (an ICECUBE python client-server pair for submitting HTCondor glidein jobs on remote batch systems) [1] to connect such Institutional schedds with CMS Connect.

[1] https://github.com/WIPACrepo/pyglidein



Pyglideins workflow diagram