Stability and scalability of the CMS Global Pool: Pushing HTCondor and glideinWMS to new limits

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 052031 doi:10.1088/1742-6596/898/5/052031

Stability and scalability of the CMS Global Pool: Pushing HTCondor and glideinWMS to new limits

J Balcas[1], B Bockelman[2], D Hufnagel[3], K Hurtado Anampa[4], F Aftab Khan[5], K Larson[3], J Letts[6], J Marra da Silva[7], M Mascheroni[3], D Mason[3], A Perez-Calero Yzquierdo[8,9], A Tiradani[3]

[1] California Institute of Technology, Pasadena, CA, USA
[2] University of Nebraska - Lincoln, NE, USA
[3] Fermi National Accelerator Laboratory, Batavia, IL, USA
[4] University of Notre Dame, South Bend, IN, USA
[5] National Centre for Physics, Quaid-I-Azam University, Islamabad, Pakistan
[6] University of California San Diego, CA, USA
[7] Universidade Estadual Paulista, São Paolo, Brazil
[8] Port d'Informació Científica, Barcelona, Spain
[9] Centro de Investigaciones Energéeticas, Medioambientales y Tecnológicas, CIEMAT, Madrid, Spain.

E-mail: aperez@pic.es

Abstract. The CMS Global Pool, based on HTCondor and glideinWMS, is the main computing resource provisioning system for all CMS workflows, including analysis, Monte Carlo production, and detector data reprocessing activities. The total resources at Tier-1 and Tier-2 grid sites pledged to CMS exceed 100,000 CPU cores, while another 50,000 to 100,000 CPU cores are available opportunistically, pushing the needs of the Global Pool to higher scales each year. These resources are becoming more diverse in their accessibility and configuration over time. Furthermore, the challenge of stably running at higher and higher scales while introducing new modes of operation such as multi-core pilots, as well as the chaotic nature of physics analysis workflows, places huge strains on the submission infrastructure. This paper details some of the most important challenges to scalability and stability that the CMS Global Pool has faced since the beginning of the LHC Run II and how they were overcome.

1. The CMS global pool

The CMS Global Pool is a single HTCondor [1] pool covering all Grid computing processing resources pledged to CMS plus significant Cloud and opportunistic resources. Resource provisioning is performed by a glideinWMS [2] frontend, which contacts several glideinWMS factories in order to submit pilot jobs to sites. Payload jobs are then matched to pilots by the HTCondor Negotiator which runs as part of the Central Manager of the pool, as can be seen in Figure 1. The other main element of the HTCondor Central Manager is the Collector, which maintains information about the various HTCondor pool daemons described below.

The main components of this Global Pool include a glideinWMS frontend and factories, the HTCondor Central Manager and Condor Connection Broker (CCB), deployed in 24-core 48GB (RAM) virtual machines (VMs) running on hypervisors with 10 Gbps ethernet connectivity. Such a set up is deployed at CERN with an analogous infrastructure for High Availability (HA)

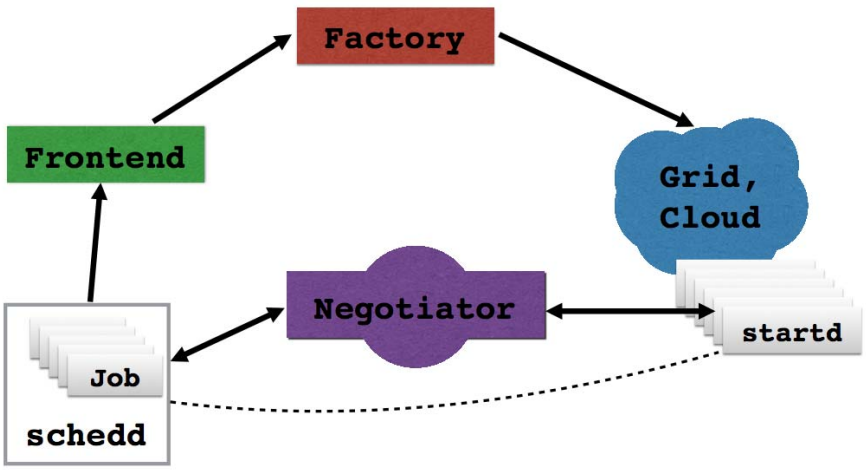

Figure 1. Schematic view of the CMS global pool, showing the main elements of the glideinWMS and HTCondor layers.

at Fermi National Accelerator Laboratory (FNAL). Additional glideinWMS factories are also deployed at UCSD and OSG-GOC.

There are some 30 job submission nodes, generically referred to as schedds, connected to the pool. About 20 schedds dedicated to Monte Carlo simulation and other data processing activities are deployed at CERN and FNAL on 16-core 64 GB physical machines and also 24-core 48 GB VMs. About ten schedds for physics analysis jobs are run at CERN on 24-core 48GB VMs (currently being moved to 32-core 58 GB VMs) and some physical machines at UCSD.

The main limits to the scalability of this system come from the I/O between the pool components, the combinatorics at the Negotiator, which has to process queued jobs against all potentially matching pilot execution slots, generically known as startds, as well as the speed of individual elements themselves.

2. CMS Global Pool: Scalability and Stability

The challenge CMS faces is getting the Global Pool to work stably at higher and higher scales each year. CMS computing needs grew 40% year over year between 2015 and 2016. By the end of 2016, CMS had regular access to approximately 150,000 dedicated CPUs, including spare cycles on the Tier-0 [3] and the High Level Trigger (HLT) computing farm [4], plus another 50,000 CPUs for which we compete opportunistically.

In addition to traditional Grid resources, connecting new and diverse resource types keeps pushing the scale even faster [5]. Examples include resources at the LHC Physics Center at Fermilab, Cloud resources (e.g. AWS used as an extension to the Fermilab Tier-1 site [6]), the opportunistic use of CMS HLT farm, the inclusion of High-Performance Computing sites, etc.

The monthly average concurrently used CPU cores running CMS jobs since the beginning of Run II across all computing tiers is shown in Figure 2, where it can be seen that analysis usage has doubled while production usage has trebled. Instantaneous peaks of 180,000 CPUs in the global pool, utilizing Tier-1, Tier-2, Tier-3, and HLT resources, were achieved in 2016 as seen in Figure 3.

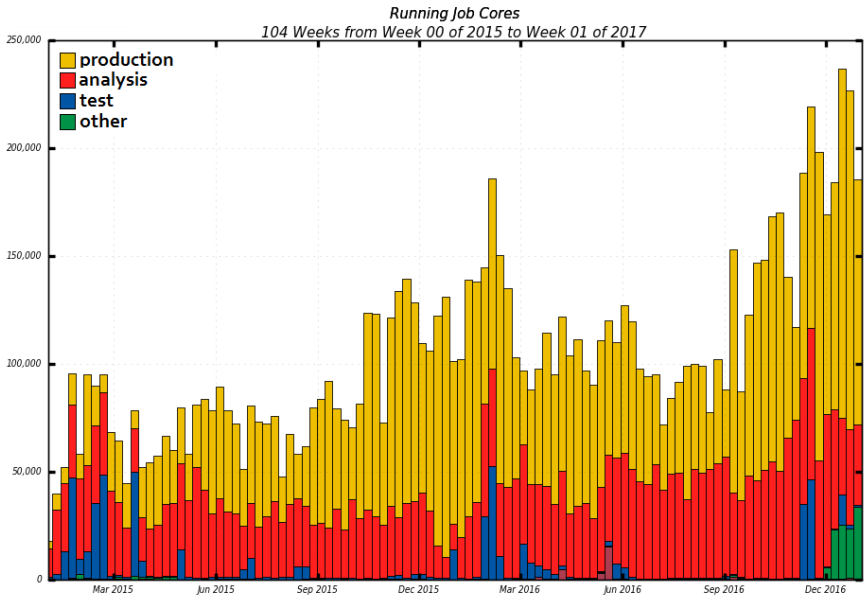

Figure 2. Usage of the CMS global pool resources during LHC Run II (2015-2016), as a function of job type.

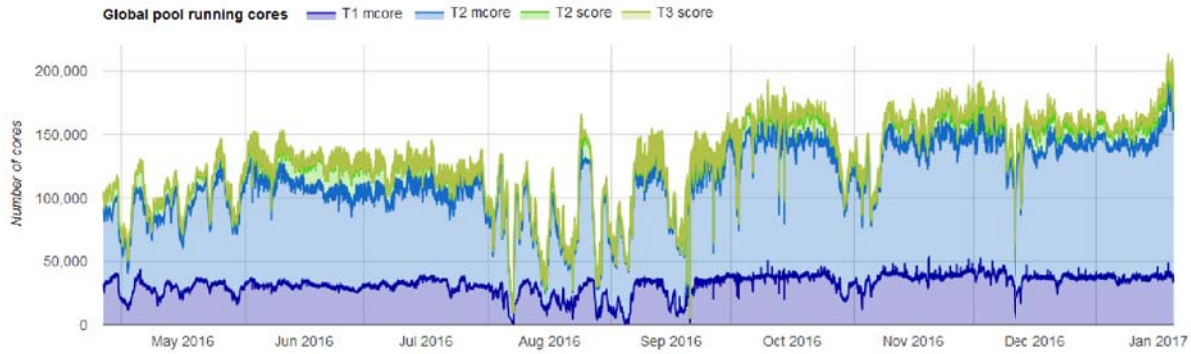

Figure 3. CMS global pool increasing scales achieved during 2016, showing the mixture of single-core and multi-core pilots at the different computing tiers.

2.1. Scalability tests with OSG

The results of scale testing to 200,000 CPUs were reported by the OSG in 2015 [7],[8]. These tests used single-CPU *über-glideins* which ran up to 32 startds per physical CPU in order to simulate the network traffic and load on central pool components but using a limited number of worker nodes.

The main recommendation from that round was to deploy the CCB on separate hardware from the rest of the Central Manager, as shown in Figure 4. This resulted in significant I/O improvement for both the CCB and Central Manager, which allowed the test pool to scale easily to manage 200,000 CPUs. CMS worked closely with the glideinWMS developers during 2015 to implement this functionality, resulting in a much more stable Central Manager capable of easily reaching scales of 140,000 CPUs in the production Global Pool.

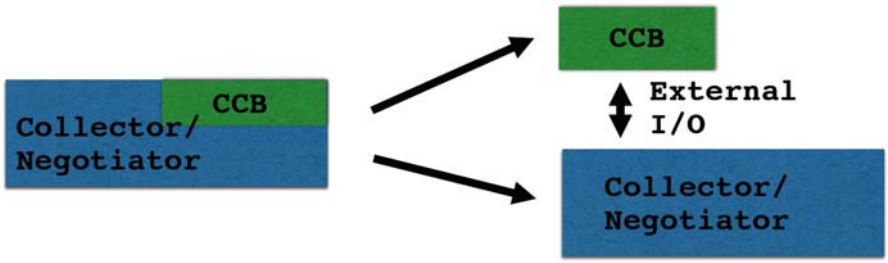

Figure 4. Separation of CCB from the HTCondor central manager, running the pool collector and negotiator, to a different hardware.

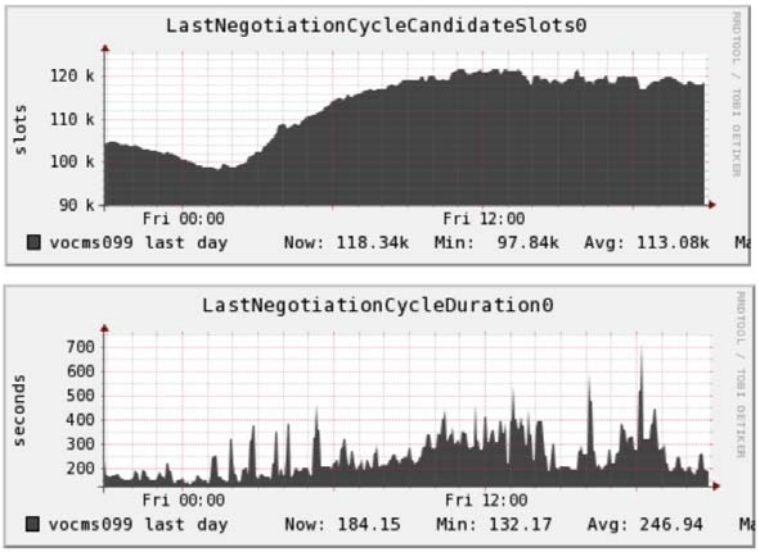

Figure 5. The CMS Global Pool scalability is constrained by the absolute amount of different combinations of job resource requests and available execution slots the Negotiator can efficiently handle: increasing number of slots per negotiation cycle (top) strongly correlates to longer observed negotiation cycle duration (bottom).

2.2. Next scalability bottlenecks and actions to mitigate them

The negotiation cycle length, the time it takes for the Negotiator to match jobs to CPUs, is one of the most closely watched metrics in the Global Pool. In early 2015 we found that this cycle length was increasing rapidly when the pool was managing more than 120,000 CPUs, as seen in Figure 5, resulting in an inability to efficiently match jobs to resources. In principle, negotiation of jobs to resources is both a combinatoric problem and an I/O problem, since every startd, for example, must maintain network connections to the Collector.

We worked closely with the HTCondor developers to test and integrate a prototype parallel Negotiator configuration, where the resources were split 3 ways between various computing Tiers in the old MONARC Computing Model [9] and geographic regions. This separation also allowed us to do resource-based fair-share, i.e. favor production at Tier-1 sites while having an equal balance of production and analysis at other sites. HTCondor developers have since started working on a fully parallel Negotiator, work which CMS is following closely as well.

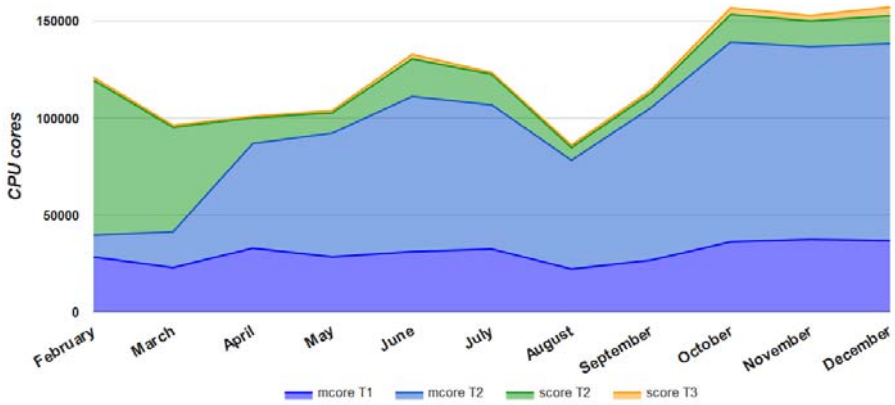

Figure 6. Evolution of CMS global pool composition by pilot type and site Tier level, moving towards a multicore pilot pool along 2016.

Another of the most important scalability improvements has been the move to multi-core pilots (generally 8 CPUs) and jobs [10]. Tier-1 resources were configured to accept multicore pilots by Summer 2015, while a deployment campaign to major Tier-2 sites followed in Spring 2016, as shown in Figure 6. As a result, the CMS CPU resource pool is currently managed at 90% level by means of multicore pilots, which are capable of simultaneously scheduling payload jobs with mixed specifications, including core-count, required memory, user id and activity (e.g. centralized Monte Carlo production and user analysis jobs).

Running multi-core jobs not only reduces their memory requirement per CPU, it also reduces the combinatorics of the matchmaking by the number of CPUs per job. However, more CPUs per pilot were initially observed to increase the slot status update rate in the HTCondor Central Managers by the same factor of 8. In an scenario dominated by single-core payloads, all eight single-core dynamic slots would report its status back to the collector whenever any of them changed status. In this way, an improvement in the combinatorics of the Negotiator was traded for an I/O increment. Joint effort by CMS and HTCondor developers concluded in the need to filter out non-essential updates, finally improving Negotiator cycle times by a factor of 2 or more (I/O improvement).

Further I/O improvements for the Central Managers involved moving the connections from the job schedulers (schedds) to their own dedicated child Collector processes, reducing I/O load on the top Collector. All child Collectors use UDP to more quickly update the top Collector instead of TCP. The Negotiator also now fetches multiple resource request lists from the schedds instead of one at a time.

Slow schedds proved to be a major obstacle for fast negotiation. For example, a slow or non-responsive schedd could block the Negotiator for several minutes. We reverted to 32-bit (not 64-bit) shadow binaries for lighter memory usage on schedds. The shadow is a process that runs on the schedd host to communicate with the remotely running job. We also disabled fsync on the schedds to prevent blocking disk I/O. In addition, CMS requested new configuration knobs for the Negotiator from the HTCondor developers to drop overloaded and/or unresponsive schedds. Finally, the Negotiator was re-configured to ignore glideins (pilots) with zero CPUs available during the matchmaking, in order to speed up negotiation by reducing the combinatorics.

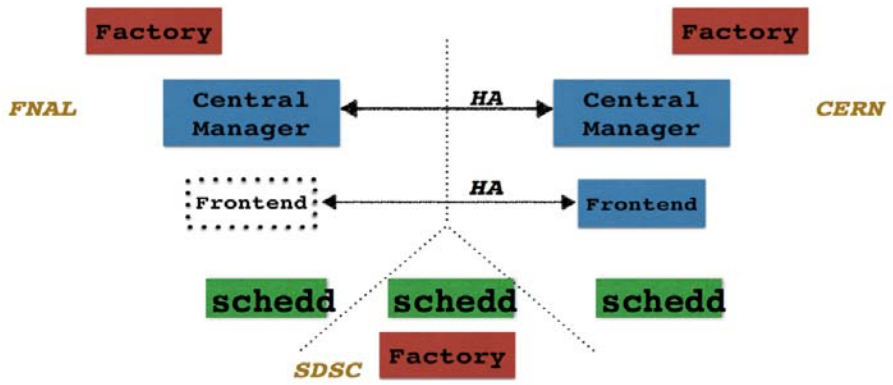

Figure 7. CMS global pool High Availability setup.

2.3. Future plans on scalability tests

CMS Submission Infrastructure has planned a new round of scale tests with the OSG for the second half of 2017. CMS is interested in examining even higher scales, following the increasing trends of CPU needs for the LHC Run II and beyond, as well as the effects of multi-core pilots, and with a more diverse job mix (resource request lists) to model actual and future CMS usage. We will look for scaling limitations and stability issues well above 200,000 CPUs so that they can be tackled during this exercise, not when they are encountered next year in a production environment.

Already in late 2016 an additional scalability issue was encountered with the production Central Manager running on OpenStack VMs at CERN [11], where the VM experienced an internal UDP backlog when the pool was managing more than 155,000 CPUs. This resulted in dropped packets and consequently inefficiency in the Negotiator. Temporarily the Negotiator was moved to a physical machine, which improved performance up to scales close to 200,000 CPUs. We are currently investigating more performant types of VMs to host the Central Manager.

2.4. Stability and High Availability

All HTCondor and glideinWMS services (Central Managers, CCBs, job schedulers, glideinWMS frontend and factories) are deployed in high availability (HA) mode or in a redundant set-up in several availability zones, as seen in Figure 7. We worked closely with the glideinWMS developers to test and integrate the new HA frontend. In case of (planned or unplanned) service interruptions at CERN, the central managers and frontend at Fermilab take over running of the Pool.

3. Conclusions

Key to the success of the Global Pool stably reaching ever higher scales has been CMS close coordination with the HTCondor developers, the glideinWMS developers, and the OSG. We have regular meetings where we discuss the priority for future work and report back on our testing of new features. Thanks to the work of the OSG we can anticipate future scaling and stability problems and stay off the bleeding edge of limitations.

Acknowledgments

The present work is partially funded under grants from the U.S. Department of Energy and National Science Foundation and Spain Ministry of Economy and Competitiveness grant

FPA2013-48082-C2-1/2-R. The Port dInformació Científica (PIC) is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universitat Autonoma de Barcelona.

References

[1] `http://research.cs.wisc.edu/htcondor/`
[2] Sfiligoi I *et al*, 2009 The pilot way to Grid resources using glideinWMS Proc. WRI World Congress on Computer Science and Information Engineering vol. 2 pp. 428-432
[3] Hufnagel D *et al*, 2015 The CMS TierO goes Cloud and Grid for LHC Run 2, J. Phys. Conf. Ser. 664 (2015) 032014
[4] Colling D *et al*, 2014 Using the CMS High Level Trigger as a Cloud Resource, J. Phys. Conf. Ser. 513 (2014) 032019
[5] Balcas J *et al*, Connecting Restricted, High-Availability, or Low-Latency Resources to a Seamless Global Pool for CMS, to be published in these proceedings
[6] Girone M *et al*, Experience in using commercial clouds in CMS, to be published in these proceedings
[7] Letts J *et al*, Pushing HTCondor and glideinWMS to 200K+ Jobs in a Global Pool for CMS before LHC Run 2, J. Phys. Conf. Ser. 664 (2015) 062030
[8] E. Fajardo *et al*, How much higher can HTCondor fly, J. Phys. Conf. Ser. 664 (2015) 062014
[9] The MONARC project http://monarc.web.cern.ch/MONARC/
[10] Perez-Calero Yzquierdo A *et al*, CMS readiness for multi-core workload scheduling, to be published in these proceedings
[11] Bell T *et al*, Scaling the CERN OpenStack cloud, J. Phys. Conf. Ser. 664 (2015) 022003