EPJ Web of Conferences **245**, 03016 (2020) https://doi.org/10.1051/epjconf/202024503016
CHEP 2019

Evolution of the CMS Global Submission Infrastructure for the HL-LHC Era

Antonio Pérez-Calero Yzquierdo[1,2,*], *Maria* Acosta Flechas[3], *Diego* Davila Foyo[4], *Saqib* Haleem[5], *Kenyi* Hurtado Anampa[6], *Todor Trendafilov* Ivanov[6], *Farrukh Aftab* Khan[3], *Edita* Kizinevič[7], *Krista* Larson[3], *James* Letts[4], *Marco* Mascheroni[4], and *David* Mason[3], for the CMS Collaboration

[1]Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
[2]Port d'Informació Cientifica (PIC), Barcelona, Spain
[3]Fermi National Accelerator Laboratory, Batavia, IL, USA
[4]University of California San Diego, La Jolla, CA, USA
[5]National Centre for Physics, Islamabad, Pakistan
[6]University of Notre Dame, Notre Dame, IN, USA
[7]Vilnius University, Vilnius, Lithuania

Abstract. Efforts in distributed computing of the CMS experiment at the LHC at CERN are now focusing on the functionality required to fulfill the projected needs for the HL-LHC era. Cloud and HPC resources are expected to be dominant relative to resources provided by traditional Grid sites, being also much more diverse and heterogeneous. Handling their special capabilities or limitations and maintaining global flexibility and efficiency, while also operating at scales much higher than the current capacity, are the major challenges being addressed by the CMS Submission Infrastructure team. These proceedings discuss the risks to the stability and scalability of the CMS HTCondor infrastructure extrapolated to such a scenario, thought to be derived mostly from its growing complexity, with multiple Negotiators and schedulers flocking work to multiple federated pools. New mechanisms for enhanced customization and control over resource allocation and usage, mandatory in this future scenario, are also described.

1 Introduction

The Submission Infrastructure (SI) team within the CMS collaboration is in charge of managing the infrastructure in which all experimental and simulated physics data processing, production, reconstruction, and analysis tasks takes place. Resources from geographically distributed providers supporting the CMS experiment are joined into a dynamically sized and centrally managed HTCondor [1] pool, the CMS Global Pool [2]. The Global Pool is built mainly by the submission of GlideinWMS [3] pilot jobs (*glideins*) to computing elements (CEs) at the Worldwide LHC Computing Grid [4] (WLCG) sites, but it can also incorporate other types of resources such as computing power allocated at HPC centers or commercial and public Cloud CPUs. Having a unified Global Pool allows diverse resource types to be

*e-mail: aperez@pic.es

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

seamlessly integrated, as well as CMS policies, such as workload prioritization, to be centrally managed, endowing CMS computing operations with high flexibility and efficiency.

2 Global Pool scale and complexity evolution

As the CMS Global Pool has played a major role in the successful realization of the CMS scientific program during the LHC Run 2, it is convenient to examine its evolution in recent years, which, together with our current understanding of future CMS resource needs and technology trends, will help us to preventively detect potential weaknesses and evaluate the reach into the future of our current model. The following subsections motivate our efforts in exploring the scaling boundaries of our pool, as well as the need to incorporate new features in our SI to further increase its flexibility, which will be later described in these proceedings.

2.1 Increasing scales during LHC Run 2

The CMS Global Pool has experienced sustained growth during the LHC Run 2, driven by the increasing resource needs in this period. Moreover, during the Run 2 and LS2 periods, we have been progressively adding opportunistic (out-of-pledge) resources, including the HLT farm for offline processing [5] when not in data taking mode. As a result of this, our main HTCondor pool has doubled its size in the past 3 years, now running routinely 250,000 cores, reaching 300,000 CPU cores in peaks, as shown in figure 1, including the smaller CERN pool (described in figure 3).

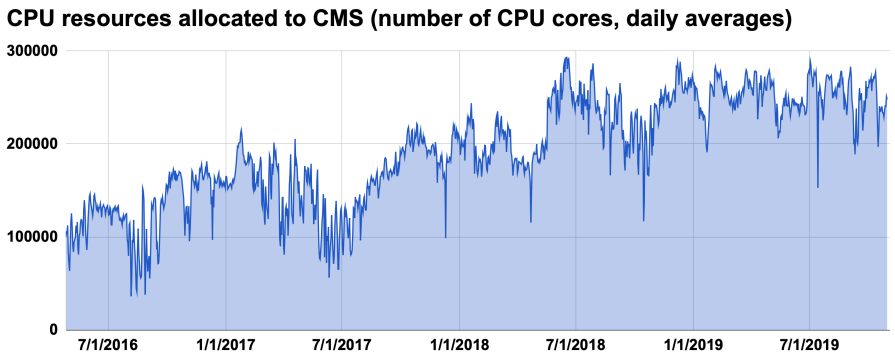

Figure 1. Global Pool size in number of CPU cores averaged daily over the last 3 years

2.2 A multi-core pool

CMS moved to employing multi-core pilots by the start of Run 2 [6], supporting dynamic partitioning of slot resources (e.g. CPU, Memory) according to the workload needs. Partitionable slots are thus fragmented into dynamic slots, each matching and executing a job. The fragmentation level of the Global Pool slots varies depending on the composition of the CMS workload (e.g. the evolving mix of single and multi-core jobs). Every dynamic slot has to regularly report its state to the Global Pool *collector*, a daemon running in the Central Manager (CM) that keeps the status of the pool and feeds that information to the *negotiators*, daemons in charge of the request to resource matchmaking. The level of load on the collector is therefore driven by the total number of dynamic slots. Evidence of scalability limitations

in the collector performance has been detected for the Global Pool in situations where the average number of CPU cores per job approaches 2 (the system is running a majority of single core jobs), causing our 300.000 cores to be divided into 150.000 running dynamic slots, as shown in figure 2.

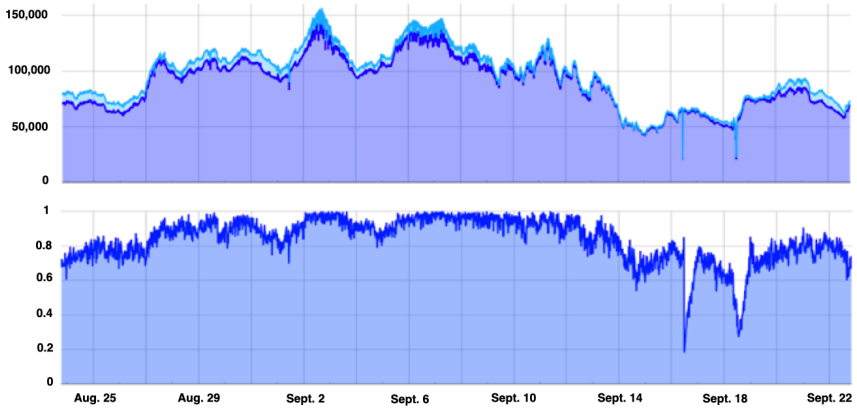

Figure 2. Total number of dynamic slots in the Global Pool (top) and collector duty cycle percentage (bottom). It can be observed that the collector reaches saturation load when the number of slots in the pool approaches 150.000

2.3 Increasing complexity

The CMS SI is evolving from a model based on a single Global Pool into a more complex system of multiple federated pools, as shown in figure 3. A pool of resources is defined by a *collector* and at least one *negotiator*. A number of centralized workload submitters (*schedds*) are attached to a given pool, therefore they are allowed to claim resources from this pool. However, when the resources found on such pool are not sufficient to cover the demand, *schedds* are then authorized to request further resources from certain neighboring pools (*flocking*).

As shown in figure 3, sets of Tier-0, CRAB, and WMAgent *schedds* (diverse workload management tools handling prompt collector data aggregation and reconstruction, analysis jobs and centralized production workloads respectively) are spread between CERN and FNAL sites, and each group belongs to a pool and is allowed to *flock* jobs into a number of other pools, as indicated. For example, in the case of the CERN pool, its main job submitters are those related to Tier-0 specific tasks, given their criticality when CMS is in a data taking period. However, when those resources are not in use by the Tier-0, regular production and analysis tasks from the Global Pool are allowed to consume them. In every matchmaking negotiation cycle, Tier-0 schedds are given preference access to the slots, so in practice the mix of Tier-0, analysis and production tasks running in the CERN pool fluctuates according to the demands of the Tier-0.

Effectively, resources are still mostly acquired with GlideinWMS pilots submitted to WLCG sites. However, they now also include vacuum-instantiated slots (e.g Cloud based DODAS [7] managed resources), BOINC [8] slots (at the heart of CMS@Home [9]), opportunistic (including HLT slots and slots acquired by pilots running in non-CMS OSG sites). The HEPCloud pool [10] is managed from the Tier-1 site at FNAL, and enables access to cloud and allocation-based HPC resources in the USA. The CERN pool now also includes

Batch on EOS Extra Resources (BEER) [11], where computing tasks are allowed to opportunistically make use of the unused capacity on CERN storage servers. Finally, the model also allows for externally managed pools, aggregating a diversity of resources that conform to CMS requirements, to be used by tasks that reach them via job flocking.

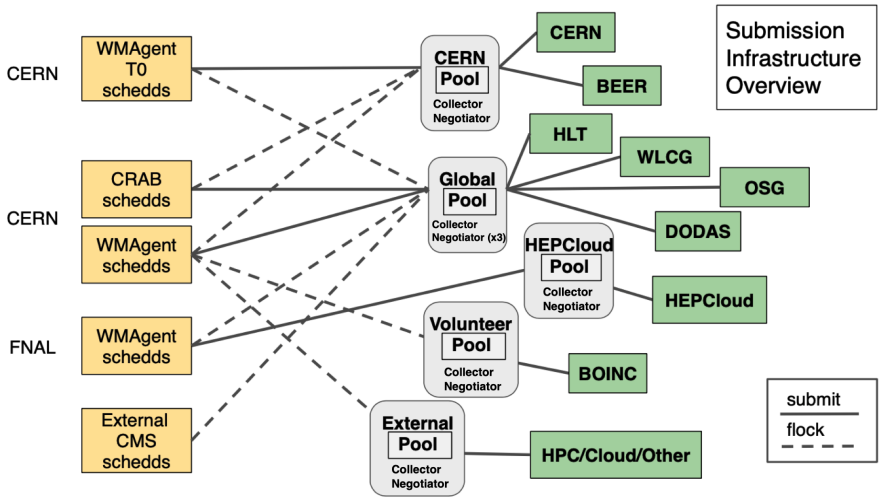

Figure 3. CMS SI current model, including multiple federated pools of resources from diverse origins and sets of distributed job schedulers

2.4 Prospects for LHC Run 3 and HL-LHC

CMS processing needs projected into the HL-LHC era (see HSF roadmap [12]) are foreseen to grow due to the increasing number of events to be collected and with the increased event complexity (increasingly larger $pile-up$). A growth factor of up to 20 times more CPU compared to the current levels has been predicted (note however that the CMS HL-LHC computing demands are regularly reviewed, which could likely suggest a lower growth factor). Improvements in software, including moderate progress in further job parallelization, are still expected, which could help reduce the load on the Workload Management (WM) infrastructure, by limiting or even completely removing the need to run single-core workflows. It is still not determined how many schedulers will be needed to manage such increasing workloads. Moreover, the current understanding of the future funding policies imply that the required growth in CPU resources is likely not to be simply obtained from increasing capacities at existing or new Grid sites. On the contrary, an increasing fraction of HPC [13] resources is expected, probably including non-standard processors such as GPUs. The ability to acquire and utilize such a substantial fraction of the total available capacity is therefore essential for the future success of the experiment.

3 Challenges

The CMS SI team must continue assessing the scalability and stability of our infrastructure with dedicated tests, exploring the limits of a single HTCondor pool, pushing the dynamic slot limits to higher scales. This will allow us to gain increased headroom in the collector capacity, so that single-core tasks dominating the workload mix would not induce a vulnerable state in the CM. Additionally, bursts of out-of-pledge resources (e.g. HPC) joining the

pool could be then absorbed with no degradation of the collector performance. The scalability limitations of the federated HTCondor pools model must also be tested, for example determining how many different pools (negotiators) our schedulers can efficiently flock jobs to. The scheduler maximum job submission rate needs also to be measured, so that sufficient submit capacity can be planned in order to fill continuously growing resources. We need to also test and commission the new multi-threaded negotiator capability in order to keep matchmaking time under control. Finally, a continuous re-assessment of general improvements introduced in the GlideinWMS and HTCondor software packages is mandatory to ensure their viability into higher resource and demand scales, while retaining a high efficiency and flexibility in our resource utilization, reliably allowing us to continue enforcing the CMS workload prioritization policies.

4 Exploring current HTCondor scaling boundaries

As employed in previous iterations of our tests [14], increasingly larger test pools can be simulated by running multiple multi-core slots (*startd*) per physical CPU core on the Grid (*uberglideins*). Our test HTCondor pool CM was running on a Virtual Machine (VM) host at CERN with 40 CPU cores and 115 GB of RAM (later upgraded to 256 GB RAM, as will be described), running 3 negotiator instances, in an identical setup to the production Global Pool. A pool of 10 VMs at CERN (each with 32 CPU cores and 60 GB RAM) running a corresponding number of *schedds* provide the job submit pressure. As workload for each of the tests iterations, up to 2 million jobs in total are injected in the schedd queues, all configured as single-core jobs in order to produce maximal pool fragmentation. Other job specifications, such as memory, disk, and site whitelist, are randomly generated within realistic values, to maximize job diversity. A job length with a Gaussian distribution is selected (e.g. 8 hours with a dispersion of 2 hours), which can be tuned to force the job completion and submission rates in order to vary the degree of stress on the schedds.

Several scale test rounds were launched from July to October 2019, as figure 4 shows. In the first phase, to establish the baseline performance, a relatively old HTCondor version was used (8.7.8, employed in the previous campaign of tests, and in production in the Global Pool at that time). It was observed that the amount of slot updates kept saturating the collector incoming UDP packets buffer (of 1GB). The memory usage in the CM host kept growing with scale of the pool, reaching an average at 85 GB with spikes approaching full host memory saturation observed (as additional collector workers requiring about 25 GB each were being forked to respond to external queries). The CM host crashed due to memory starvation when approaching 450,000 dynamic slots. In subsequent test attempts, HTCondor was upgraded to the latest development series version (8.9.2). However, a problem was discovered as the parent collector process was found to be forwarding status updates to itself, which required developers help to solve. In a later stage of the tests, further retuning was introduced on the CM (UDP buffer size was doubled), involving also configuration tuning for the Condor Connection Broker (CCB) element of the pool. A new 256 GB RAM CM host was deployed and fully integrated with the multi-threaded Negotiator enabled. It was followed by a new test with the latest HTCondor (8.9.3). In this last test, the capability of using IPv6 protocol was enabled in the pool, along with turning on the shared port daemon.

As a summary of the test results, the pool managed to keep about 450,000 jobs running simultaneously, which represents about a factor 4 over the number of dynamic slots typically measured in the current production pool. However, the efficiency in the usage of the slots was observed to degrade, as the CM barely managed to keep about 80% of slots busy, in contrast to the usual high efficiency of the production pool, typically at and over 95%. The rate of slot status updates being processed by the collector peaked at 500 Hz (600,000 updates for each

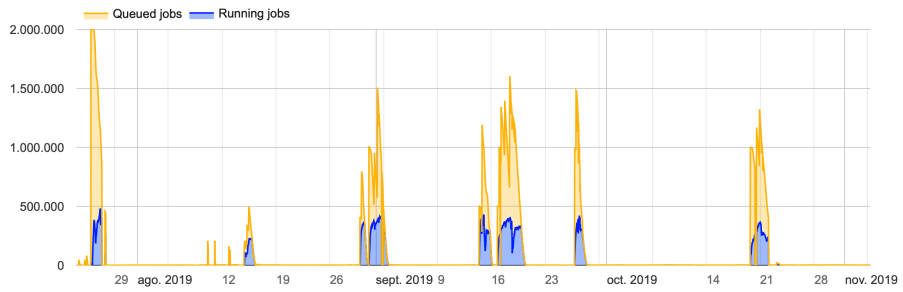

Figure 4. Number of simultaneously running and queued jobs for the scalability tests performed during 2019 on the CMS HTCondor Global Pool. Notice the successive attempts approaching the 500,000 simultaneously running jobs mark, under varying conditions, as described in section 4.

20 minutes statistics collection window), with the duty cycle saturated at 100%, signaling this as the main constraint to the growth of the pool. The UDP buffer for the collector saturated even after the increase to 2 GB, resulting in missed slot updates. This in turn explains the degraded utilization of the pool, given that the negotiator is forced to run the matchmaking using only partial and potentially outdated information about the status of the pool slots. Given that the main conclusion from these tests was the bottleneck found in the capacity of the collector to digest the high rate of slot updates, ideas on how to solve this are already being examined with the HTCondor developers team.

The CPU load on the CM did not grow to worrisome levels and the memory usage, despite peaking at 130 GB, did not produce any CM crashes, thanks to the increased memory on the host. This allows us to extract the interesting observation that, even if scaling limitations have been found for the pool, it leads to a soft and progressive degradation of its performance, rather than a massive failure that would halt the full system. Additionally, thanks to making use of the new multi-threaded version of the negotiator, the negotiation time was constrained to reasonable values (about 5 to 10 minutes for each cycle) despite the increased pool sizes, thanks to the reduction on the matchmaking phase duration. It was hence considered ready to be used in the production pool, where the final matchmaking step (driven by the combinatorics of multiple requests and resources) is the dominant component of negotiation cycle. Schedd's maximum job start rate was measured close to 4Hz, with a maximum of running jobs per schedd near 50,000 jobs. Concerning IPv6 readiness of the infrastructure, about one quarter of the startds in the test connected to the CM via IPv6. Moreover, one of the test schedds was configured to support IPv6, and it was observed that about one third of the jobs submitted from this node were being tracked by the schedd via IPv6. Confirming IPv6 communication capabilities of our infrastructure at realistic scales was therefore a very positive additional aspect of these tests.

5 New features in CMS SI for increased flexibility

The emergence of specialized resources that can be attached to existing grid sites and used opportunistically, with some extra conditions, generates the need for a mechanism to restrict the types of jobs that can be executed on such resources. A number of use cases has recently appeared, for example the requirement to prevent the execution of user analysis tasks on opportunistic resources, such as BEER slots at CERN. Another use case, important for HPC resources, would be the need to only run production jobs related to a certain physics study for which a specific resource allocation has been granted. Finally, it would be desirable to filter and execute only jobs that require no external source for input data when exploiting

resources with poor or no connectivity to an external storage service. Our infrastructure relies on a strategy based on the principle of late-binding of jobs to slots. Therefore, all slots receive from the pilots a standard set of generic matchmaking conditions, which allows them to be occupied by any type of CMS workflow. In order to enforce the restrictions described before for non-standard resources, our pilots have been modified to include the capability of importing additional local constraints from a standard location in the CMS file space (site-customizable pilots). An implementation of the new feature has already been tested and found to be fully functional, therefore the first use cases are now being addressed.

Thanks to our close collaboration with the HTCondor and GlideinWMS development teams, several new functionalities of interest for CMS WM capabilities are being explored. These include the ability to produce resource-based fair share (i.e. allow analysis jobs to run on at least 25% of the slots at any given Grid site, instead of globally or at best for some subsets of the whole pool), network-aware scheduling (which would limit the number of jobs that require WAN access to remote input data, or even IO-intensive jobs within a site's LAN, that can potentially block storage resources at a given site). Also, new concepts job and resource *sets* in HTCondor, along with *late−materization* of jobs into the schedd queues aim at improving the scalability of our infrastructure and in general of our WM tools. Moreover, work is being done to improve the support for heterogeneous resources (CPU+GPU slots), and use of SciToken as an alternative to GSI authentication.

6 Conclusions

The CMS SI team continues exploring potential limitations in the CMS infrastructure to detect bottlenecks preventively and assess scalability into CMS projected needs, expecting increasing scales in the 2020's, as well as more complexity in the resource mix (Grid, HPC and Cloud). A series of scale tests have been launched, pushing the size of our pool up to about 450,000 running jobs, where limitations on a key process, the top collector processing slot updates have been found. A reduced knowledge of the pool status leads to degraded slot usage, not causing however a critical failure of the system. Nevertheless, limitations have been found at much bigger scales than that of our production pool current operations, thus the present setup is evaluated to be viable for Run 3 needs. Negotiation cycle has improved thanks to multithreaded negotiators, which has an immediate impact on our production system. Finally, a number of additional techniques are being considered in order to endow CMS SI with further flexibility and reach, looking onwards to the HL-LHC challenge.

This work was partially supported by the U.S. Department of Energy, the National Science Foundation, and by Spain's Ministry of Economy and Competitiveness grant FPA2016-80994. CMS thanks our partners in the GlideinWMS and HTCondor development teams, the OSG, and our colleagues at CERN, all of whom make the shared computing infrastructure a success.

References

[1] HTCondor public web site, https://research.cs.wisc.edu/htcondor/index.html

[2] J. Balcas et al. "Using the glideinWMS System as a Common Resource Provisioning Layer in CMS", J. Phys.: Conf. Ser. **664** 062031 (2015).

[3] The Glidein-based Workflow Management System, https://glideinwms.fnal.gov/doc.prd/index.html

[4] The Worldwide LHC Computing Grid http://wlcg.web.cern.ch

[5] D. Da Silva Gomes et al. "Experience with dynamic resource provisioning of the CMS online cluster using a cloud overlay", EPJ Web of Conferences. **214**. 07017 (2019).

[6] A. Pérez-Calero Yzquierdo et al. “CMS readiness for multi-core workload scheduling”, J. Phys.: Conf. Ser. **898** 052030 (2017).

[7] D. Spiga et al. “Exploiting private and commercial clouds to generate on-demand CMS computing facilities with DODAS”, EPJ Web of Conferences. **214**. 07027 (2019).

[8] D. P. Anderson. “BOINC: A Platform for Volunteer Computing”, J Grid Computing (2019).

[9] LHC at Home, https://lhcathome.cern.ch/lhcathome/

[10] S. Timm et al. “Virtual machine provisioning, code management, and data movement design for the Fermilab HEPCloud Facility”, J. Phys.: Conf. Ser. **898** 052041 (2017).

[11] D. Smith et al. “Sharing server nodes for storage and computer”, EPJ Web of Conferences. **214**. 08025 (2019).

[12] HEP Software Foundation, “A Roadmap for HEP Software and Computing R&D for the 2020s”, HSF-CWP-2017-01, arXiv:1712.06982 physics.comp-ph (2017).

[13] A. Pérez-Calero Yzquierdo et al. “CMS Strategy for HPC resource exploitation”, to be published in these proceedings.

[14] J. Balcas et al. “Pushing HTCondor and glideinWMS to 200K+ Jobs in a Global Pool for CMS before Run 2”, J. Phys.: Conf. Ser. **664** 062030 (2015).