EPJ Web of Conferences **214**, 03002 (2019) https://doi.org/10.1051/epjconf/201921403002
CHEP 2018

Exploring GlideinWMS and HTCondor scalability frontiers for an expanding CMS Global Pool

Antonio Pérez-Calero Yzquierdo[1,2,*], *Brian Paul* Bockelman[3], *Diego* Davila Foyo[4], *Kenyi* Hurtado Anampa[5], *Todor Trendafilov* Ivanov[6], *Farrukh Aftab* Khan[7], *Amjad* Kotobi[8], *Krista* Larson[7], *James* Letts[9], *Marco* Mascheroni[9], and *David* Mason[7], for the CMS Collaboration

[1]Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
[2]Port d'Informació Cientifica (PIC), Barcelona, Spain
[3]University of Nebraska-Lincoln, Lincoln, NE, USA
[4]Benemérita Universidad Autónoma de Puebla, Puebla, México
[5]University of Notre Dame, Notre Dame, IN, USA
[6]University of Sofia, Sofia, Bulgaria
[7]Fermi National Accelerator Laboratory, Batavia, IL, USA
[8]University of Malaya, Kuala Lumpur, Malaysia
[9]University of California San Diego, La Jolla, CA, USA

Abstract. The CMS Submission Infrastructure Global Pool, built on GlideinWMS and HTCondor, is a worldwide distributed dynamic pool responsible for the allocation of resources for all CMS computing workloads. Matching the continuously increasing demand for computing resources by CMS requires the anticipated assessment of its scalability limitations. In addition, the Global Pool must be able to expand in a more heterogeneous environment, in terms of resource provisioning (combining Grid, HPC and Cloud) and workload submission. A dedicated testbed has been set up to simulate such conditions with the purpose of finding potential bottlenecks in the software or its configuration. This report provides a thorough description of the various scalability dimensions in size and complexity that are being explored for the future Global Pool, along with the analysis and solutions to the limitations proposed with the support of the GlideinWMS and HTCondor developer teams.

1 Introduction

The CMS Global Pool is a dynamically sized and centrally managed HTCondor [1] pool, built by the submission of GlideinWMS [2] pilot jobs (*glideins*) to the Computing Elements of the multiple resource providers supporting CMS distributed across the Worldwide LHC Computing Grid [3] (WLCG), and extended to additional resources [4] as well, such as opportunistic (e.g. grid sites not pledging CPU to CMS), Cloud, and allocations on HPC. The Global Pool, making use of many of its HTCondor native features, such as partitionable slots, accounting groups and priority handling, has become an essential tool for the successful operations of CMS Computing.

As a unified pool [5], it joins geographically distributed resources together under control of a single HTCondor Central Manager (CM) and also provides a single entry point for tasks

*e-mail: aperez@pic.es

with diverse resource requests (e.g. single-core and multi-core jobs [6], jobs with higher-than-standard memory requirements, etc). It allows CMS policies such as workload prioritization and fair-shares (e.g. production workflows vs. analysis tasks) to be centrally managed. The Global Pool has also demonstrated flexibility to integrate non-pledged resources (e.g. opportunistic usage of the CMS High Level Trigger farm [7]). Finally, all of this is achieved while ensuring a high level of efficiency in the utilization of the allocated CPUs [8].

Considering how successful such an infrastructure and its operation has been for CMS during the LHC Run 2, and looking onwards to the coming years, the CMS Submission Infrastructure (SI) team has been performing tests in order to explore the potential limitations of our current model when increasing in size and complexity, which will be described in the following sections.

2 Motivation for scale testing

The first driving motivation for the scale test activity is the steady increase in the size of the Global Pool over the past several years, which is expected to continue. As the CMS CPU needs have grown, more resources have been deployed by the grid sites, a trend shown in Fig. 1, covering the last two years. The major component of the pool comes from multi-core pilots running at both Tier-1 and Tier-2 CMS grid sites [9], while a remainder of single-core pilots persists on smaller Tier-2 and Tier-3 sites. The size of the Global Pool is determined both by resource deployments (which tend to happen later in the calendar year, as sites deploy their pledged CPU), as well as resource requests (e.g. relatively low usage during 2016 and 2017 summer months is observed in Fig. 1). In addition to increasing pledged CPU capacity, during the LHC Run 2 the CMS HLT farm was commissioned as opportunistic resource (with peaks of 60k cores in 2018), which, together with additional HPC, Cloud and grid opportunistic resources, have pushed the size of the pool to routinely reach 250k cores nowadays. Extrapolating historical usage trends, by LHC Run 3 the CMS Global Pool must be able to manage stably and efficiently 0.5M CPU cores, about a factor 2 higher than the current size.

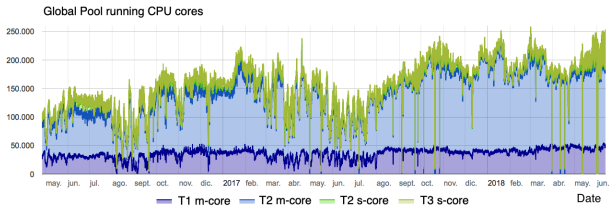

Figure 1. Global pool size in number of CPU cores over the last 2 years composed of multi-core pilots at Tier-1 and Tier-2 CMS grid sites (in dark and light blue respectively), plus single-core pilots running at smaller Tier-2 and Tier-3 sites (green)

Secondly, the need to evaluate the limitations of the Global Pool model comes from its already increasing complexity. In the classical pilot-based pool, in use for most of the Run 2 years, upon detecting workload in a job submit node (*schedd*) queue, the GlideinWMS Front-End (FE) triggers the submission of multi-core pilots from the factories to the matching grid or cloud resources [2]. Each pilot creates a partitionable execute slot (*startd*), which registers at the Global Pool *collector* and can be assigned workload by the *negotiator* (both daemons

running in the CM). In the new extended model, as displayed in Fig. 2, new methods for job submission and resource allocation have been incorporated, such as CMS Connect [10] and DODAS [11]. The interconnection of multiple federated pools (referred in HTCondor terms as "pool flocking") has also been now enabled. All these elements provide additional flexibility to the infrastructure.

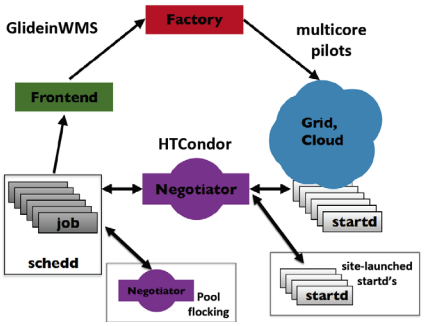

Figure 2. Extended schema of the CMS Global Pool, as described in the main text

One key element in the last part of the LHC Run 2 has been the redeployment of the CMS Tier-0 resources from dedicated OpenStack VMs at CERN into slots in a Tier-0 dedicated farm at CERN, shared with other LHC VOs. CMS decided to join all CERN HTCondor resources, up to now split into partitions dedicated to Tier-0 and Tier-2 tasks exclusively, into a single CERN pool (as shown in Fig. 3). This greatly increases the flexibility with which CMS can utilize those resources, as tasks priorities and shares are subsequently fully under CMS control. This new CERN pool is federated to the Global Pool, resulting in a more complex picture from the point of view of the schedds, which now have to run negotiation cycles with multiple negotiators over the two pools.

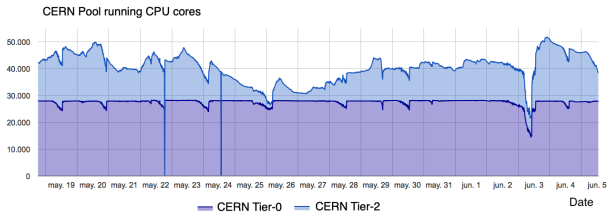

Figure 3. The new CMS CERN pool, deployed in 2018, reaching scales of about 50k CPU cores

In addition, the emergence of the HEPCloud [12] in the USA, designed to combine grid, cloud and HPC resources under local control (potentially to be followed by similar initiatives appearing in Europe), signals that our infrastructure could well be evolving into a model of multiple federated pools, with centralized HTCondor schedds. How many pools (negotiators) can a schedd efficiently interact with before reaching the limits of its capabilities is not known at this point. It is worth mentioning however that CMS is already using three negotiators in the Global Pool plus another one for the CERN pool.

Finally, looking onwards to the HL-LHC era (around 2025), it is expected that CMS needs for processing power will grow by a factor of 20 compared to Run 2 needs [13], resulting in increasing number of jobs and CPUs to manage. As much of this is expected not to be provided by traditional grid sites, this growth will also be a driver of the model complexity, as CMS computing will depend in increasingly higher proportions on resources coming from HPC sites and cloud resource providers.

3 Scale testing of the Submission Infrastructure

The Global Pool resources are configured as partitionable slot startds allowing dynamic provisioning of slots (CPU, Memory, Disk). Partitionable slots can become fragmented into lower core count dynamic slots, with the degree of fragmentation oscillating according to the composition of the workload mix. CMS workload is dominated by single-core and 8-core jobs, although intermediate values are allowed, as multi-core jobs can be resized at matchmaking stage. As a consequence, even for a pool of a total stable size, the number of dynamic slots can greatly oscillate, driving the load on the collector and negotiator. Typical values, illustrated in the 30-day period shown in Fig. 4, average at about 110k slots, with peaks at 150k when the pool is dominated by single-core requests.

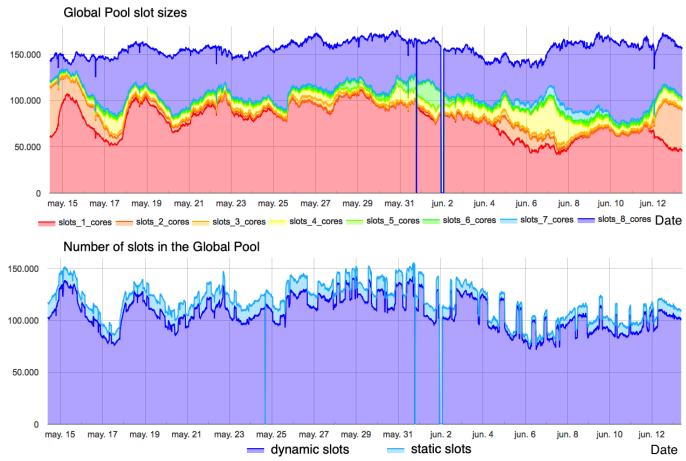

Figure 4. Fragmentation of the pool into slots of diverse number of cores (top) and total corresponding number of dynamic and static slots (bottom)

In previous studies, CMS and OSG teams studied the scalability and stability of a single HTCondor pool [14, 15], identifying bottlenecks in the I/O between and within pool components, the combinatorics at matchmaking and in the speed and memory usage of individual components. Increasing the use of multi-core workflows will provide some headroom for further growth, but "single-core storms" still may expose setup vulnerabilities. Therefore, exploring the limitations of the number of dynamic slots in a single pool is still relevant to study. We are also interested in the additional scalability challenges introduced by working with federated HTCondor pools, including schedulers interacting with multiple negotiators. An assessment of the maximum schedd job start, submission and completion rates is needed in order to determine how to profit from continuously growing resources. Finally, we are interested in the exploring the capabilities of the multi-threaded negotiator algorithm.

A testbed to run scaling tests has been deployed at CERN, including a GlideinWMS Front-End (24 cores, 46 GB, same as production, running with glideinWMS 3.2.21 and HTCondor 8.6.10), plus a dedicated factory. The test pool's CM host is comparable to that of the Global Pool (40 CPU cores and 120 GB, running HTCondor 8.7.8). A total of ten schedds (32 cores, 60 GB, highIO disks, running HTCondor 8.6.11, as used in production) were employed.

Using so-called uber-glideins [15] allows to run multiple startds on one batch slot (e.g. launching 32 slots per pilot allows the creation of 500K core pool on only 16K real batch cores). Artificial workload can be used to simulate diverse scenarios, with job requests being randomly generated (number of cores, memory, desired sites lists, etc).

4 Scaling tests results

The setup described in the previous section has been used to run successive test rounds increasing workload pressure and targeting the maximum pool size. In total, up to two million 8-hour jobs were injected. All jobs were submitted as single-core to maximize the pool fragmentation and stress on the pool's CM. Figure 5 shows four trials, where the first and second attempts found limitations per schedd to grow beyond 50k simultaneously running jobs. Having submit capacity saturated with 6 schedds, the pool could not grow beyond 300k slots. For the next rounds, the number of dedicated schedds grew to 10, which removed the limiting factor from the submit side. With the increased capacity to run jobs, the maximum scale achieved close to 350k dynamic slots in total. This scale represents about a factor 3 in slots compared to the production Global Pool, which would be equivalent to a pool of 600k cores, considering the present average cores to running jobs ratio. The following subsections describe the limitations observed at each of the elements of the pool with the present setup and software.

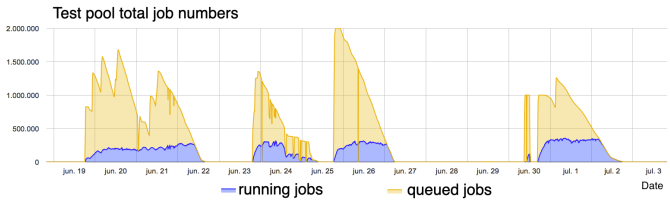

Figure 5. Maximum scales achieved with the scalability tests run in Summer 2018, with up to 2M jobs in total and 350k running jobs simultaneously

4.1 GlideinWMS Front-End

At the scales of the test, a very slow FE matchmaking cycle was observed, requiring more than 2 hours to complete compared to cycles typically lasting for 15 minutes in the Global Pool. The main potential causes are inefficient queries (as highly loaded schedds are very slow in reporting their queue to the FE) and insufficient parallelization to deal with the increasing request to resource combinatorics. With respect to increasing FE parallelism, it could be configured to use multiple workers in parallel (tested up to 8), but we ran into host memory limitations in these scales (Fig. 6), when each worker was using about 4 GB. These issues have been discussed with GlideinWMS developers to further understand and remove

bottlenecks from the FE matchmaking cycle. In any case, CMS might need to run the FE in a bigger host to speed up the FE performance, if we were to reach higher pool sizes in the short term.

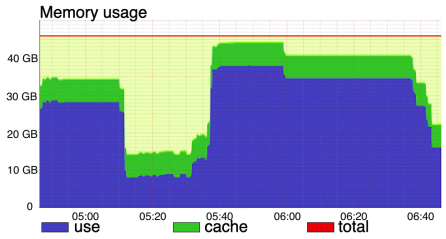

Figure 6. GlideinWMS FE host node memory usage increment during a matchmaking cycle, basically saturating the host capacity

4.2 Schedds

The capacity of our schedds, running in hosts identical to production ones, has been explored in terms of maximum numbers of jobs up to 50k jobs running and 150k jobs in queue at once. Memory consumption has been measured to increase at about 1 MB per process tracking job status (job *shadow*) when using the 32-bit binary version provided by HTCondor developers), as seen in Fig. 7. Exploring the maximum sustainable rates for job submission, start, and completion rates, the schedds achieved a total of 10k jobs started in 20 minutes statistics collecting interval, therefore a O(10) Hz start rate seems feasible for jobs averaging 8 hours.

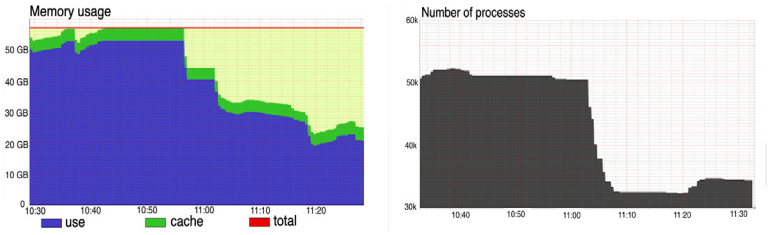

Figure 7. Schedd performance during the scale tests: memory usage (left) as a function of the number of simultaneously running job-shadow processes (right)

4.3 Collector and Negotiator

Essential elements of the pool, such as the negotiator and the GlideinWMS FE (also JobRouter-based algorithms and even monitoring tools) rely on the collector's performance. Its task is to collect and distribute information on workload and available resources quickly and efficiently. In the last of the tests described in this report, the collector was fully stressed, with slot update rates reaching above 400k per 20 minute accounting interval, saturating communications buffers even after successive increments (from 128 MB up to 512 MB), and

with duty cycle at 99.5% (Fig. 8). Parallelization was enabled by running multiple collector workers (up to 8), but, as a result of applying it, memory consumption in the host registered frequent spikes, such that the kernel's out-of-memory killer started cancelling such processes. At these scales and rates, the main collector thread was detected being killed when using 20 GB of RAM.

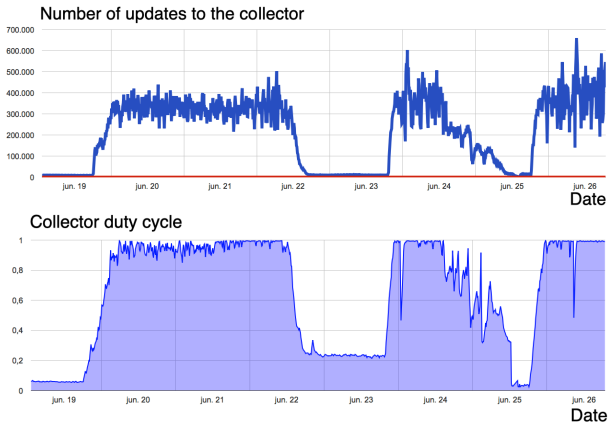

Figure 8. Collector performance during the scale tests: number of slot updates at the collector per 20 min. interval (top) and saturating collector duty cycle (bottom)

No real scalability issue was observed for the negotiators (the test pool runs with 3 negotiator instances, as in the production Global Pool), although the matchmaking cycle often reached the target time limit at 600s. As Fig. 9 shows, the cycle time is about equally split on the collection of slot updates and the matchmaking phases. In contrast, the production pool cycle is dominated by matchmaking, generally driven by slow heavily loaded schedds (dropped from the negotiation cycle after 60s). Speeding up matchmaking could benefit from running forked negotiators (included in recent HTCondor releases). However, it was not employed in these tests, considering that memory on the CM host (120 GB, the biggest standard VM available at CERN) was already saturated by running multiple collector workers.

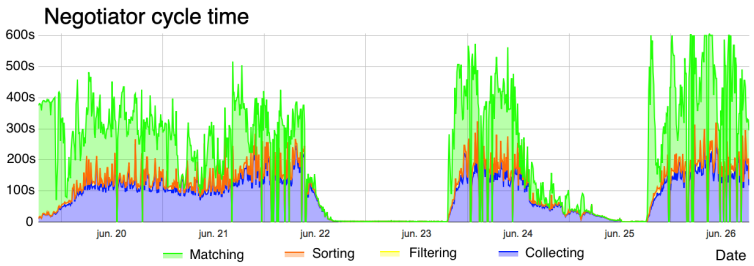

Figure 9. Negotiator performance during the scale tests: time dedicated in the negotiation cycle for each of its phases, the collection, filtering and sorting of slot classads, followed by the matchmaking

5 Conclusions and Outlook

The CMS Submission Infrastructure team is analyzing the very successful Global Pool model in order to preventively detect potential bottlenecks deriving from increasing scales or additional complexity. This is driven by the prospective of the LHC Run 3, and specially to the HL-LHC, evolving towards a more heterogeneous mix of Grid, HPC and Cloud resources to satisfy increasingly higher computing needs.

Our recent scale tests pushed the size of our pool using a configuration that maximizes its fragmentation, to the point where it revealed scaling limitations on key components, such as the GlideinWMS FE and HTCondor schedds and collector. Saturation was reached at about 350K slots, however, which represents a factor 3 in size with respect to the production pool with an equivalent setup. Results indicate therefore that our infrastructure is far from reaching an immediate scaling frontier. Nevertheless, the schedd capabilities need to be thoroughly explored and the viability of a model based on federated pools needs to be further addressed. In the longer term, we need to ensure the sustainability of HTCondor and GlideinWMS-based infrastructure to serve our workload management needs in the future.

We wish to thank the development communities of HTCondor and GlideinWMS for their continued support and close collaboration with CMS. This work was partially supported by the U.S. Department of Energy and the National Science Foundation and Spain's Ministry of Economy and Competitiveness grant FPA2016-80994.

References

[1] http://research.cs.wisc.edu/htcondor/

[2] http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/

[3] http://wlcg.web.cern.ch

[4] D. Hufnagel, "Enabling opportunistic resources for CMS Computing Operations", J. Phys.: Conf. Ser. **664** 022025 (2015).

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

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

[7] D. Da Silva Gomes et al. "Experience with dynamic resource provisioning of the CMS online cluster using a cloud overlay", to be published in these proceedings.

[8] J. Letts et al. "Improving the Scheduling Efficiency of a Global Multi-core HTCondor Pool in CMS", to be published in these proceedings.

[9] The CMS Collaboration, CMS Computing Technical Design Report Preprint CERN-LHCC-2005-023 (2005).

[10] J. Balcas et al. "CMS Connect", J. Phys.: Conf. Ser. **898** 082032 (2017).

[11] D. Spiga et al. "Exploiting private and commercial clouds to generate on-demand CMS computing facilities with DODAS", to be published in these proceedings.

[12] 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).

[13] 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).

[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).

[15] E. Fajardo et al. "How much higher can HTCondor fly?", J. Phys.: Conf. Ser. **664** 062014 (2015).