EPJ Web of Conferences **214**, 03056 (2019) https://doi.org/10.1051/epjconf/201921403056
CHEP 2018

Improving the Scheduling Efficiency of a Global Multi-Core HTCondor Pool in CMS

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

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

Abstract. Scheduling multi-core workflows in a global HTCondor pool is a multi-dimensional problem whose solution depends on the requirements of the job payloads, the characteristics of available resources, and the boundary conditions such as fair share and prioritization imposed on the job matching to resources. Within the context of a dedicated task force, CMS has increased significantly the scheduling efficiency of workflows in reusable multi-core pilots by various improvements to the limitations of the GlideinWMS pilots, accuracy of resource requests, efficiency and speed of the HTCondor infrastructure, and job matching algorithms.

1 Multi-core Pool Scheduling

The CMS Submission Infrastructure (SI) Group is responsible for GlideinWMS [1] and HTCondor [2] pool operations in the CMS experiment at CERN, as well as setting and communicating our priorities to the respective software development teams. Our group constructed and operates a global HTCondor pool [3] that connects Grid resources at both WLCG and non-WLCG sites into a single, flexible batch system for production and physics analysis activities in the CMS experiment. In recent years, the submission infrastructure of CMS has expanded to include Cloud resources both directly connected to the Global Pool and also accessible through federated pools. The most important federated pool is at CERN, which includes all of the resources pledged to CMS at the host laboratory, and is part of the data taking chain of the experiment.

* Corresponding author: jletts@ucsd.edu

The CMS Global Pool is at once a GlideinWMS instance and a HTCondor pool. As seen in Figure 1, in response to demand for resources from job schedulers (schedd's), a GlideinWMS frontend queries job queues on the schedd's and sends requests to several GlideinWMS factories to submit multi-core pilot jobs (also called "glideins") to Grid and Cloud sites world-wide. These pilots instantiate HTCondor resources (a startd) that join one of the several HTCondor pools managed by the SI group. HTCondor Central Managers running one or more Negotiators then match these resources to jobs on schedulers, primarily hosted at CERN and Fermilab, completing the circle.

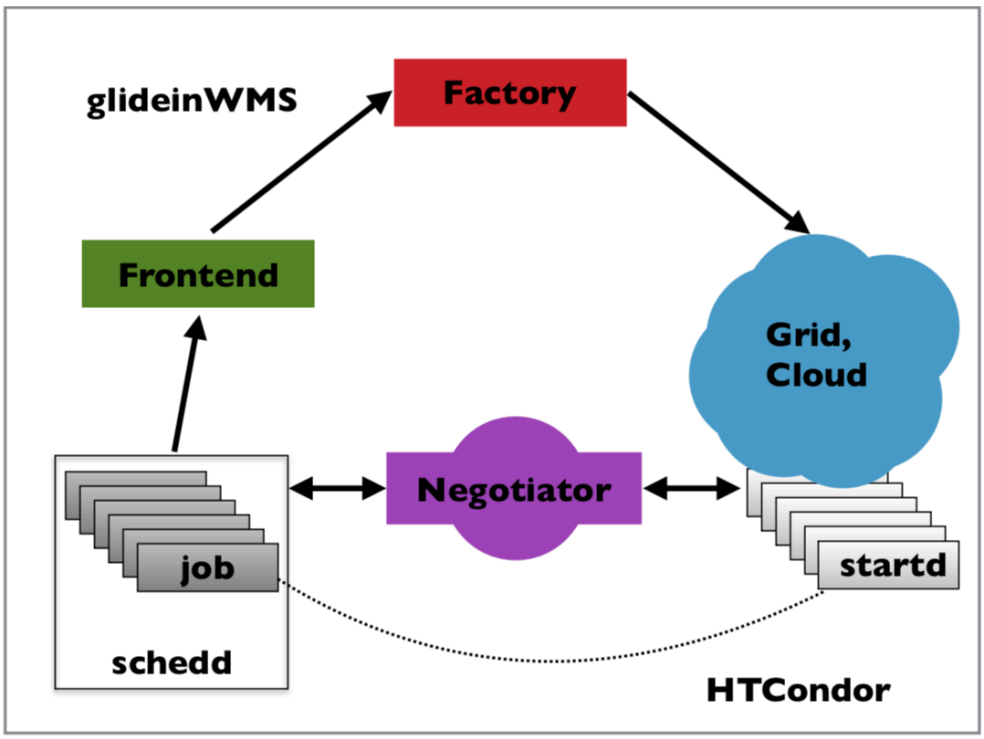

Fig. 1. Elements of GlideinWMS and HTCondor pools in CMS. Starting from the lower left and moving clock-wise: In response to the presence of queued jobs on HTCondor schedd's, a GlideinWMS frontend requests factory to submit pilots (glideins) to Grid and Cloud sites. The pilots launch HTCondor startd's which are matched to the queued jobs by a HTCondor Negotiator.

CMS schedulers flock jobs to multiple pools, the largest pool being the CMS Global Pool, which connects CMS Tier-1, Tier-2 and Tier-3 sites world-wide. Typically this pool reaches scales of 200,000 CPU cores or more. A dedicated HTCondor pool for CERN hosts the computing resources which serve the data taking apparatus of the experiment as well as CERN-based analysis. Other federated pools serve the prototype HEPCloud [4] instance in the U.S. as well as the CMS@home [5] effort in volunteer computing.

Scheduling in this environment is challenging firstly due to initial and boundary conditions on the pilots. For example, there are delays from the time the frontend requests a

pilot to the time that the pilot starts on the remote resources. Pilot jobs also have a finite (typically 48h) lifetime and must be properly drained in order to not kill payloads at the end of the pilot life, as seen in Figure 2. Secondly, pilots can become more fragmented over time as lower core count jobs finish asynchronously, making the matching of higher core count jobs impossible, even if they are from higher priority workflows.

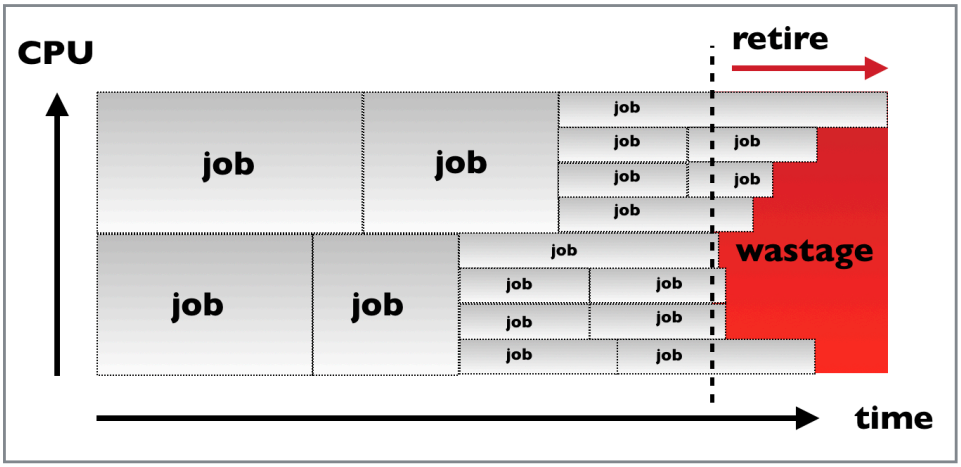

Fig. 2. Evolution of multi-core pilot fragmentation and draining. Multi-core and single-core (vertical axis) jobs are scheduled inside glideins (pilots) until the retire time (dotted line), after which the glidein drains until the last job completes, resulting in some wastage of processing time (red). Fragmentation of slots over time makes scheduling multi-threaded jobs increasingly difficult, which is why pilots need to be renewed or defragmented by draining.

2 Scheduling Efficiency

In the context of a dedicated task force beginning in 2017, CMS made a targeted effort to improve the CPU efficiency of CMS workflows. CPU efficiency in the WLCG is defined as the measured CPU time over the wall clock time weighted by the logical CPU core count. This CPU efficiency is completely factorable into a contribution from the pilot infrastructure (scheduling efficiency) and from the underlying payload job.

Note, however, that logical CPU core count is distinct from physical CPU core count. The prevalence of hyper-threading in HEP computing, both in physical and virtual machines, is not taken into account in this calculation. In many cases at CERN and at other national laboratories the sites schedule according to memory requirements rather than CPU count and thus overcommit CPU.

The pilots in CMS carry no information about which particular jobs should match to them. Matching requirements are quite generic, based on resources offered and demanded. CMS has built the submission infrastructure with this flexible model in mind. However, with the advent of Cloud and specialized resources, we are considering how to allow sites to customize pilots to properly route jobs to particular resources, or to preferentially match

certain types of jobs to pilots. This functionality exists already at some level in allowing certain types of pilots to preferentially match to jobs from locally registered users.

As seen in Figure 3, in early 2017 the scheduling efficiency in multi-core pilots was quite poor (~85% on average) relative to single-core pilots, (>95%). The CMS Submission Infrastructure Group made a dedicated effort in the second half of 2017 to improve this situation, both by tuning the pilots and also requesting and integrating improvements in GlideinWMS and HTCondor.

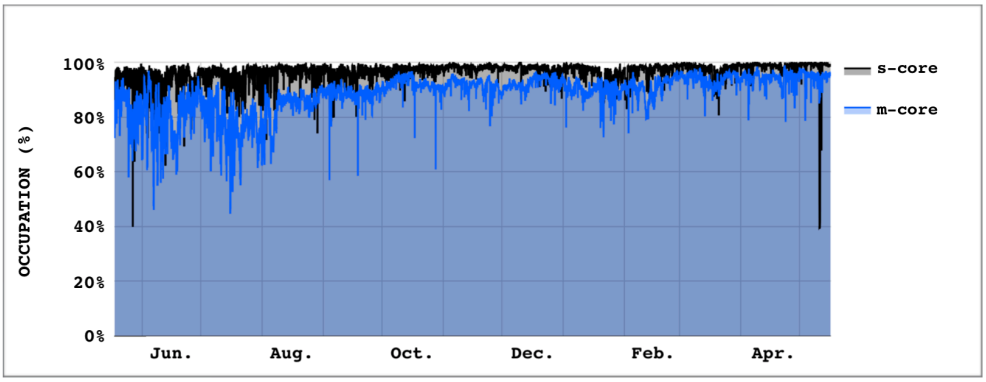

Fig. 3. Improvement of the multi-core CPU scheduling efficiency over time from mid-2017 to mid-2018. Single-core glidein scheduling efficiency is shown as the black, upper line while for multi-core glideins it is the generally lower, blue line.

3 Legitimate Use Cases

CPU is only one of the computing resources provided by sites that are scarce and need to be scheduled. Examples of other resources are memory, disk space, GPU's, and local or wide-area network bandwidth. CMS has several types of workflow that are particularly high in the consumption of memory. We consider it a legitimate use case to schedule most or all of the memory available to a pilot for a job that may not use many CPU cores, for example. The result, however, is that little or no memory may be available for the remaining cores in the pilot which leaves CPU idle. This flexibility in how we schedule the resources can be considered a strength of our multi-core pilot model. The Submission Infrastructure group monitors such memory-starved pilots as well as the number of logical CPU cores left idle for other reasons, as can be seen in Figure 4.

The opportunistic use of CPU cores by CMS in the high level trigger (HLT) farm at CERN is another use case where long-lived VM's remain idle waiting for work. The HLT is a special resource that is dedicated to processing trigger decisions for the experiment. However, during LHC inter-fill periods, when data is not being taken by the experiment, the ~30,000 CPU cores can be used for centralized processing in the Global Pool.

CMS also leaves a certain number of CPU cores ready for urgent calibration work at CERN during LHC data taking as part of a service called the CAF. These high-availability, low latency resources are another legitimate use case of leaving CPU idle.

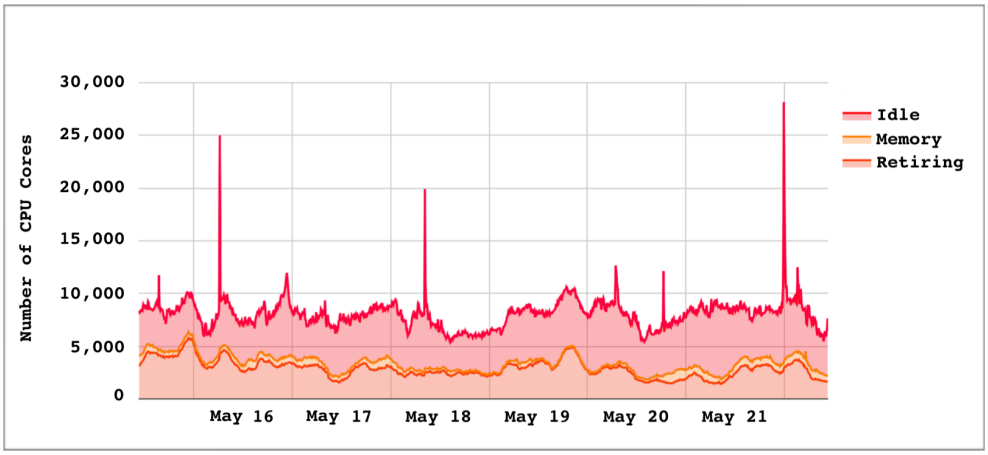

Fig. 4. Sources of idle CPU cores in the CMS Global Pool. Major contributions include retiring pilots (middle shaded area) and memory-starved pilots (lower shaded area).

In the context of the CMS CPU Efficiency Task Force, the Submission Infrastructure Group sought to minimize the contributions to overall idle CPU from any other remaining sources, such as pilot startup and retirement procedures or inefficient job scheduling.

4 Pilot Improvements

We observed that during periods of bursty job submission, as seen in Figure 5, the frequent expansion and contraction of the Global Pool often resulted in very poor (even as low as 50%) scheduling efficiency. We noted that during these periods new pilots were starting at sites long after the job pressure subsided. This decoupling in time of job pressure from resource availability was a major source of CPU wastage.

The situation was improved partly by ceasing this bursty submission pattern, but also by improvements in GlideinWMS [6] to remove idle pilots in site batch queues after a tunable amount of time. While this may cause some churn in site batch queues, it mostly eliminated this source of wastage by ensuring that no pilots were starting many hours or even days after they were requested, as was the case before July 2017, as shown in Figure 6. In a "fire-and-forget" pilot model a glidein could sit in a sites batch queue potentially for weeks before it started, perhaps long after the work that triggered the resource request has completed. While most glideins are seen to have been requested in the previous 48 hours, given the bursty nature of CMS submission patterns which changed on the timescale of hours, even 48 hours was considered too long.

We are aware that the pilots being generic carry no information about the job or type of jobs that they should match. This is an active area of consideration for future evolutions of the submission infrastructure in CMS.

Retirement of glideins is necessary not only because of job wall clock limits at sites, but also to counter fragmentation of the pilots, which can lead to priority inversions in workflow matchmaking. The length of the pilot retirement time (and consequent CPU

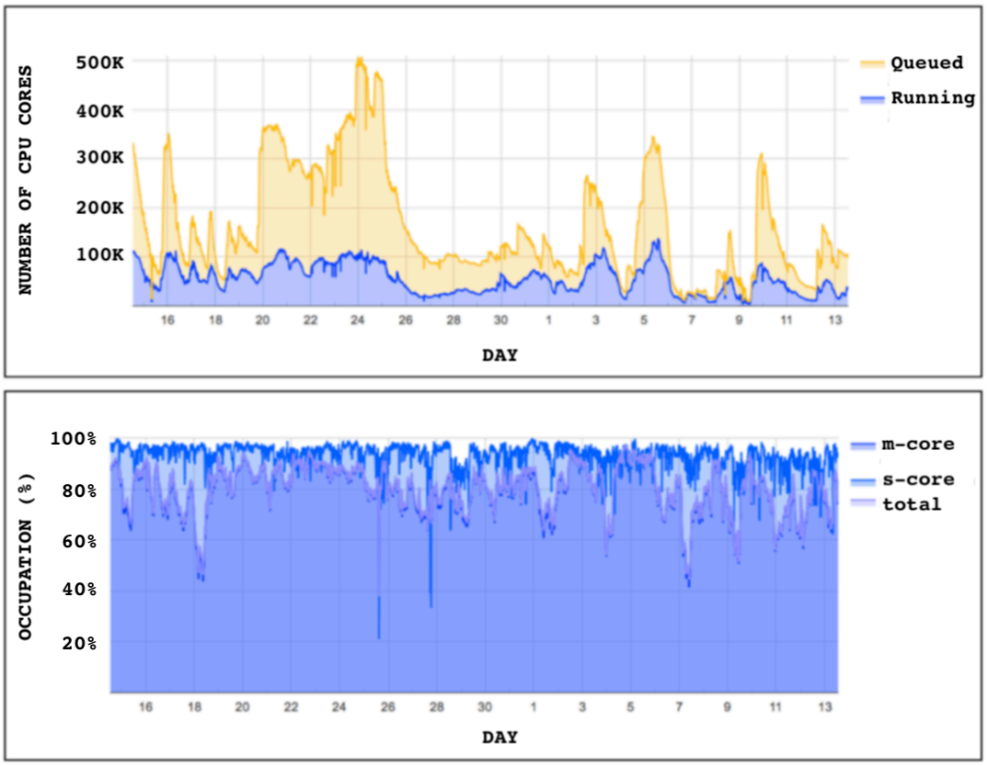

Fig. 5. Bursts of job submission and draining effect on the Global Pool. Note that multi-core scheduling efficiency (lower plot, lower shaded distribution) drops due to pilot draining during periods when the number of queued jobs drops significantly (upper plot, upper shaded distribution).

wastage while draining the glideins) is driven by the accuracy to which we know the job wall clock time. If job wall clock run time is poorly estimated at submission time, then many jobs may overrun their time limits and result in failed workflows when the pilots themselves hit time limits.

In studies we found that 99% of centralized production jobs finish within their estimated wall clock time, while analysis jobs, which are subject to much more uncertainty due to user code contributions, were found to finish within three hours of their estimated time with 99% confidence. This 3 hour uncertainty in analysis job run time estimation came after many improvements during the year [7]. These improvements allowed the SI group to shorten the retirement time to 4h from 10h, improving scheduling efficiency by several percent, and allowing >99% of jobs to complete within the pilot lifetime.

5 HTCondor Improvements

Multi-core pilots can be filled by jobs in different ways. Two distinct concepts are to fill them breadth-wise, i.e. as sparsely as possible so as to spread load, or depth-wise,

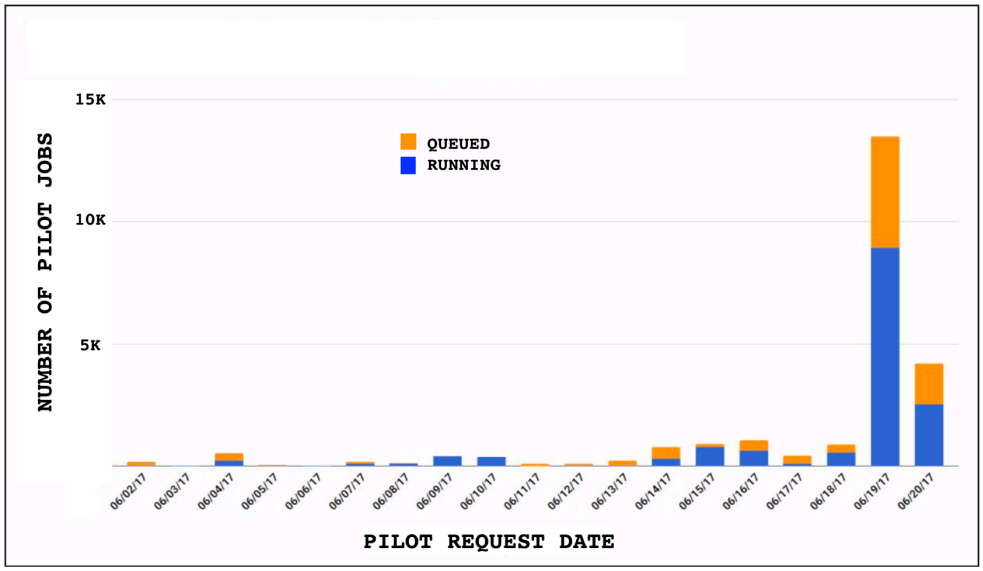

Fig. 6. Up to two-week-old glideins in the queues and running at sites on June 20, 2017 in the Global Pool. The date on the horizontal axis indicates when the pilots were first queued at the sites by the GlideinWMS factories.

concentrating jobs into as few pilots as possible. While breadth-wise is desirable from the point of load balancing, depth-wise filling of pilots results in better overall scheduling efficiency by minimizing the number of idle CPU cores.

Depth-wise filling of multi-core pilots was made available from HTCondor version 8.7.5 and integrated in the Global Pool in May 2017. Before that, it was possible for pilots to be filled randomly or even breadth-wise, which effectively maximized idle CPU in the pool. In depth-wise filling, nearly full pilots are favored for matches over sparsely-occupied glideins.

6 Conclusions

While the individual contributions of all of the fixes we implemented are difficult to quantify, the overall effect was to improve the scheduling efficiency in multi-core pilots to be comparable to single-core glideins, after legitimate use cases such as over-committing memory are taken into account. We typically had over ~98% multi-core scheduling efficiency in 2018, on par with single-core. Remaining sources of inefficiency in the submission infrastructure are largely irreducible.

This work was partially supported by the U.S. Department of Energy and the National Science Foundation, and by Spain's Ministry of Economy and Competitiveness grant FPA2016-80994.

We thank our partners in the GlideinWMS and HTCondor development teams, the OSG, our colleagues at CERN, and in other experiments such as ATLAS for their collaboration, all of which makes the shared computing infrastructure a success.

References

1. I. Sfiligoi et al., "The Pilot Way to Grid Resources Using GlideinWMS", Proc. of the 2009 WRI World Congress on Computer Science and Information Engineering, Vol. 2, pages 428-432. http://glideinwms.fnal.gov
2. D. Thain, T. Tannenbaum, and M. Livny, "Distributed Computing in Practice: The Condor Experience", Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005. https://research.cs.wisc.edu/htcondor/index.html
3. J. Balcas et al., "Connecting restricted, high-availability, or low-latency resources to a seamless Global Pool for CMS", J. Phys. Conf. Ser. 898 (2017) no. 5, 052037
4. B. Holzman et al., "HEPCloud, a New Paradigm for HEP Facilities: CMS Amazon Web Services Investigation", Comput. Softw. Big. Sci. (2017) 1: 1. https://doi.org/10.1007/s41781-017-0001-9
5. L. Field, D. Spiga, I. Reid, et al. Comput. Softw. Big. Sci. (2018) 2: 2. https://doi.org/10.1007/s41781-018-0006-z
6. M. Mascheroni et al., "Recent developments in GlideinWMS: minimizing resource wastages", CHEP18 Poster #513, in these proceedings.
7. T. Ivanov et al., "Improving efficiency of analysis jobs in CMS", CHEP18 Oral Presentation #379, in these proceedings.