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

B. Bockelman[1], D. Davila Foyo[2], K. Hurtado Anampa[3]. T. Ivanov[4], F. Khan[5], A. Kotobi[6],
K. Larson[5], J. Letts[7], M. Mascheroni[7], D. Mason[5], A Perez-Calero Yzquierdo[8]

[1]Univ. of Nebraska-Lincoln (US), [2]Aut. Univ. of Puebla (MX), [3]Univ. of Notre Dame (US), [4]Univ. of Sofia (BG), [5]FNAL (US),[6]Univ. of Malaya (MY), [7]UCSD (US), [8]CIEMAT & PIC (ES)

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.

Multi-Core Pool Scheduling

The CMS Submission Infrastructure (SI) group is responsible for glideinWMS and HTCondor operations in CMS as well as setting and communicating our priorities to the respective development teams.

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 requests a factory to submit multi-core pilot jobs to Grid and Cloud sites world-wide. These pilots instantiate HTCondor startd's that join a HTCondor pool. A HTCondor Central Manager with multiple Negotiators then matches these resources (startd's) to jobs on schedulers at CERN and Fermilab, completing the circle.

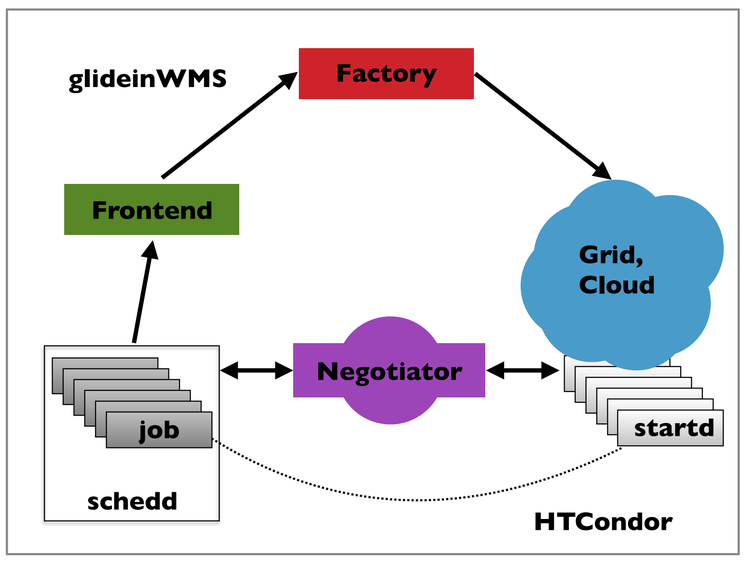

Figure 1. Elements of glideinWMS and HTCondor pools in CMS.

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.

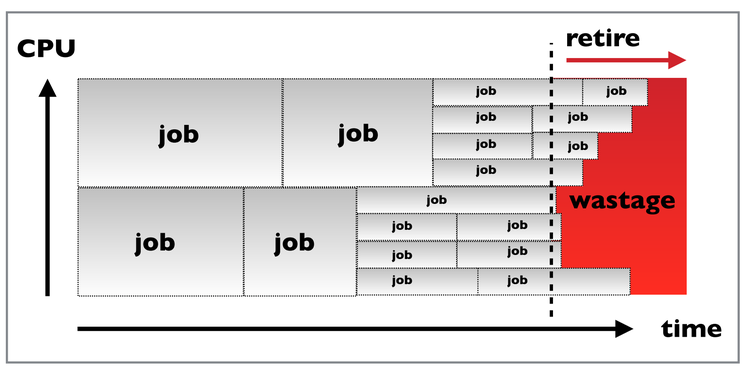

Figure 2. Evolution of multi-core pilot fragmentation and draining.

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.

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 logical CPU core count times wall clock time. This CPU efficiency is completely factorable into a contribution from the pilot infrastructure (scheduling efficiency) and from the underlying payload job.

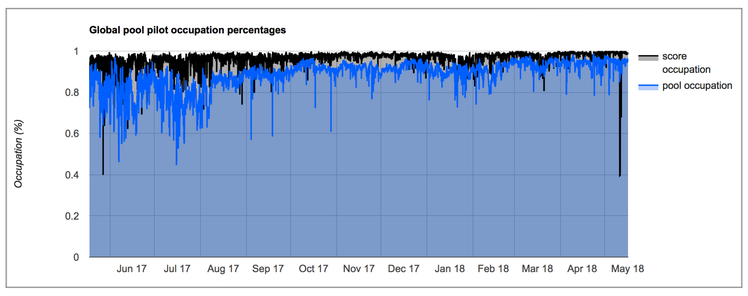

Figure 3. Improvement of the CPU scheduling efficiency over time.

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%). SI 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.

Legitimate Use Cases

There are several legitimate cases where CPU is left idle, however. As seen in Figure 4, overcommitment of RAM in pilots for high-memory workflows is one legitimate use case. CMS' opportunistic use of CPU cores in the high level trigger (HLT) farm is another, where long-lived VM's remained idle waiting for work. CMS also leaves a certain number of CPU cores (CAF) ready for urgent calibration work at CERN during LHC data taking.

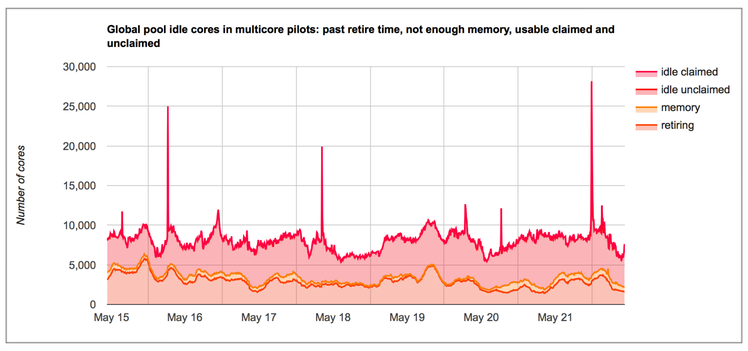

Figure 4. Sources of idle CPU cores in the CMS Global Pool.

SI sought to minimize the contributions to idle CPU from other sources, such as pilot startup, retirement, or poor occupancy.

Pilot Improvements

SI 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 [1] 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.

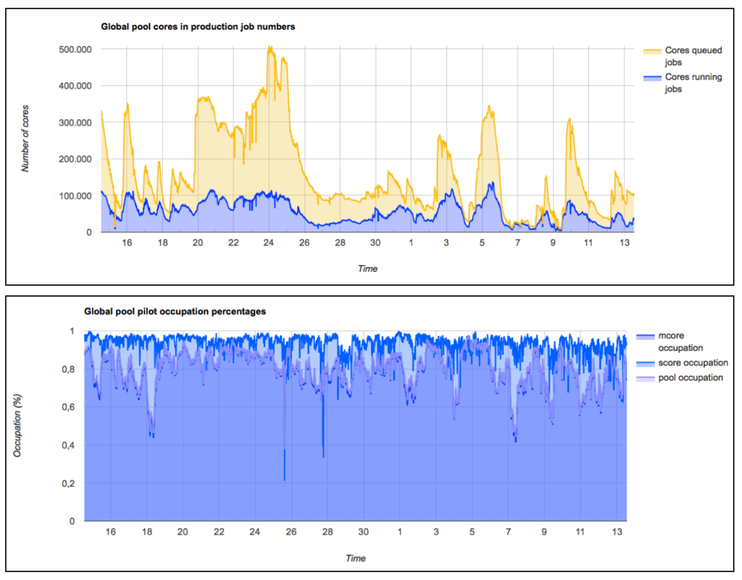

Figure 5. Bursts of job submissions and draining effect on the Global Pool.

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 wastage) is driven by the accuracy to which we know the job wall clock time. Improvements in analysis job length estimation [2] allowed us to shorten the retirement time to 4h from 10h, improving scheduling efficiency by several percent, and allowing almost all jobs to complete within the pilot lifetime.

HTCondor Improvements

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.

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 have over ~97% multi-core scheduling efficiency in 2018, on par with single-core. Remaining sources of inefficiency in the submission infrastructure are largely irreducible.

Related SI Work:

1. M. Mascheroni et al., "Recent developments in glideinWMS: minimizing resource wastages", CHEP18 Poster #513.
2. T. Ivanov et al., "Improving efficiency of analysis jobs in CMS", CHEP18 Oral Presentation #379.
3. K. Hurtado Ampara et al., ""Producing Madgraph5_aMC@NLO gridpacks and using TensorFlow GPU resources in the CMS HTCondor Global Pool", CHEP18 Poster #422.
4. A. Perez-Calero Yzquierdo et al., "Exploring GlideinWMS and HTCondor scalability frontiers for an expanding CMS Global Pool", CHEP18 Oral Presentation #438.

Contact Information: jletts@ucsd.edu

This work was partially supported by the U.S. Department of Energy and the National Science Foundation.