CMS readiness for multi-core workload scheduling

CMS readiness for multi-core workload scheduling

A Perez-Calero Yzquierdo[1,2], J Balcas[3], J Hernandez[2], F Aftab Khan[4], J Letts[5], D Mason[6], V Verguilov[7]

[1] Port d'Informacio Cientifica, Barcelona, Spain
[2] Centro de Investigaciones Energéeticas, Medioambientales y Tecnológicas, CIEMAT, Madrid, Spain
[3] California Institute of Technology, Pasadena, CA, USA
[4] National Centre for Physics, Quaid-I-Azam University, Islamabad, Pakistan
[5] University of California San Diego, CA, USA
[6] Fermi National Accelerator Laboratory, Batavia, IL, USA
[7] Bulgarian Academy of Sciences, Sofia, Bulgaria

E-mail: `aperez@pic.es`

Abstract. In the present run of the LHC, CMS data reconstruction and simulation algorithms benefit greatly from being executed as multiple threads running on several processor cores. The complexity of the Run 2 events requires parallelization of the code to reduce the memory-per-core footprint constraining serial execution programs, thus optimizing the exploitation of present multi-core processor architectures. The allocation of computing resources for multi-core tasks, however, becomes a complex problem in itself. The CMS workload submission infrastructure employs multi-slot partitionable pilots, built on HTCondor and GlideinWMS native features, to enable scheduling of single and multi-core jobs simultaneously. This provides a solution for the scheduling problem in a uniform way across grid sites running a diversity of gateways to compute resources and batch system technologies. This paper presents this strategy and the tools on which it has been implemented. The experience of managing multi-core resources at the Tier-0 and Tier-1 sites during 2015, along with the deployment phase to Tier-2 sites during early 2016 is reported. The process of performance monitoring and optimization to achieve efficient and flexible use of the resources is also described.

1. Multi-core jobs for the LHC Run 2

CMS has been moving towards employing multi-threaded algorithms for the processing of experimental and simulated data since the start of the LHC Run 2 in 2015 [1]. This trend is motivated by the evolution of the LHC experimental conditions towards higher luminosities, resulting in increasing data volumes and event complexity derived from the higher number of concurrent collisions per event (pile-up), as well as industry trends.

Continuing to run single-threaded applications would not be the best way to exploit current multi-core CPU architectures, as the application memory footprint would exceed the available RAM-per-core in most CPU resources pledged to CMS across the Worldwide LHC Computing Grid (WLCG) [2]. In addition, execution time per event would grow in such a way that the reconstruction of an entire luminosity section, the dataset unit for data processing, would result in excessively long running jobs, not suited to being executed in commonly shared multi-VO WLCG sites. As an added advantage, the use of multi-core jobs reduces the number of

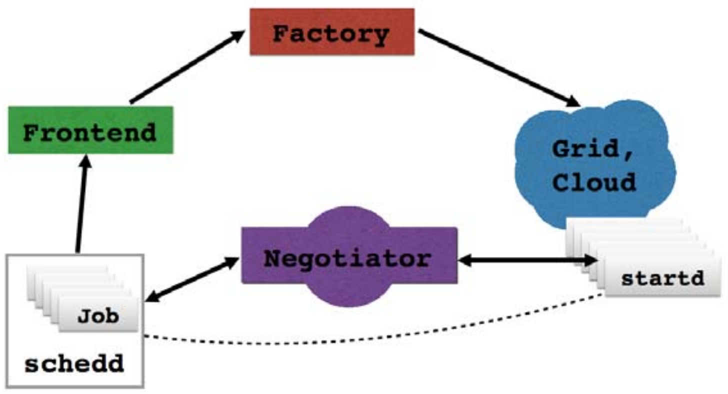

Figure 1. Schematic view of the CMS global pool including its main components.

simultaneous jobs that need to be controlled by the workflow management infrastructure down to workable limits.

In general, the preferred mode of operation for the majority of multi-threaded tasks CMS ran during 2015 and 2016 has been to employ 4 threads per job. This choice ensures high CPU usage efficiency relative to sequential execution, while it is sufficient to staying within hardware memory limitations, due to the available memory being shared by multiple threads [1].

2. CMS global pool and multi-core pilot model

The CMS Submission Infrastructure Global Pool is a single HTCondor [3] pool aggregating CPU resources accessible to CMS (pledged and opportunistic) and handling both centralized production workflows and analysis tasks. As outlined in Figure 1, a GlideinWMS [4, 5] layer manages a transient pool of computing resources by means of pilot job submission to execute nodes matching job resource requests. The HTCondor Negotiator then connects the submit nodes (schedds) to matching pilots running at execute nodes (startds).

The CMS computing infrastructure needs tools to allocate multi-core jobs to its CPU resources. However, diverse single-core and multi-core jobs must coexist during LHC Run 2, therefore the ability to continue scheduling both types of jobs is mandatory.

The main ingredient to enabling the common allocation of single and multi-core jobs has been the adoption of multi-core pilots with internal dynamic partitioning of resources, which employ the HTCondor native concept of partitionable slot. As summarized in Figure 2, and described in more detail in previous reports [6, 7, 8], multi-core pilots dynamically rearrange their internal distribution of resources into fractional slots, capable of matching diverse resource requests simultaneously. The fragmentation of the pilot internal resources by single-core jobs limits the pilot's ability to continue pulling new multi-core jobs. However, the renewal of the finite-lifetime pilots provides a mechanism that supplies the pool with non-fragmented pilots continuously [7].

The CMS model for job to resource allocation therefore relies on multi-core pilot jobs exclusively. The objectives of this model are, as described, enabling the successful scheduling of multi-core and single-core jobs, together with high efficiency in CPU usage by means of minimizing any inefficiencies deriving from the scheduling of payload jobs.

The main advantage of the fully multi-core partitionable pilot model is that it allows CMS control of the scheduling of diverse jobs types (single-core, multi-core, high-memory, etc) without requiring the intervention from the grid sites (e.g. no specialized queues are required, no

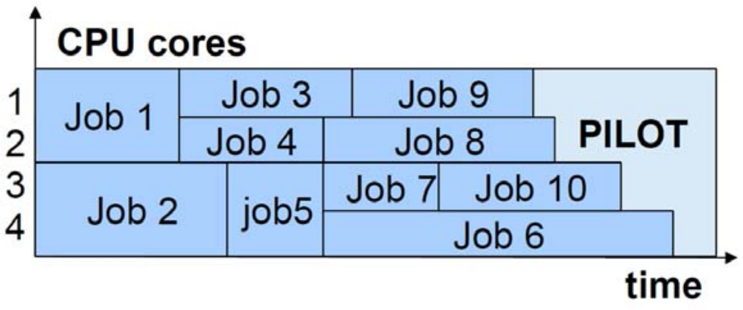

Figure 2. Multicore partitionable pilot running multiple payloads of diverse core count.

production versus analysis quotas are needed). Also, compared to using a combination of single and multi-core pilots, it removes unwanted single-core and multi-core pilot types competition for resources at the sites and for matching to jobs once running. A reduced number of pilots is then required to manage the whole CMS global pool, which contributes to the scalability of the submission infrastructure.

There are however disadvantages of employing multi-core partitionable pilots. There are causes of inefficiency in the internal scheduling of payload jobs to pilots, which are otherwise external to the VO, but detected and accounted for at the sites batch system. A more complex monitoring and accounting of allocated and used resources is required in comparison to a single payload per pilot model. Finally, a slower ramp-up of remote resource allocation is observed when compared to single core pilots in multi-VO sites, as the local site resources may need to be drained and de-fragmented in order to allocate CMS multi-core pilots. This is mitigated however by employing a common core count size with ATLAS in those sites which are shared between the two experiments, namely pilots from both VOs currently request 8 CPU cores.

3. Multi-core pilot deployment to CMS sites

The CMS priority before the beginning of the LHC Run 2 was to have multi-core job allocation enabled at Tier-1 resources by 2015, hence the first phase of deployment focused on such sites. The first tests were performed in 2014, while the full deployment was achieved along 2015, as the submission of single-core pilots to Tier-1s ceased by late 2015. Figure 3 shows the increasing use of multi-core pilots during 2015 and 2016, nearly doubling its capacity in this period. Monthly average values for allocated cores depend on the level of activity, and by the second half of 2016 over 30,000 CPU cores were regularly employed, with peaks of up to 40,000 CPU cores observed.

The second deployment phase, focused on the majority of CMS pledged CPUs at the Tier-2 sites, continued in Spring 2016. This required the integration in the model of resources managed by a wide variety of CE and batch system technologies (see Table 1), which was performed thanks to the close collaboration with CMS supporting sites and GlideinWMS Factory Operations team. Building from the experience gained in the Tier-1 deployment phase, a total of 26 Tier-2 sites were reconfigured to accept the new pilot model in a period of two months. Whenever possible, in most cases, a standard pilot size definition was configured, requesting 8 CPU cores, 16 GB of minimum memory and up to 48h of pilot running wall-clock time.

Figure 4 presents the evolution of the global pool composition during 2016. Once the deployment to the major Tier-2 sites was finished by April, close to 90% of the total global pool resources was on average being managed by multi-core pilots. A multi-core global pool was built, whose capacity continued growing and by late 2016 comprised on average around 140,000 CPU cores.

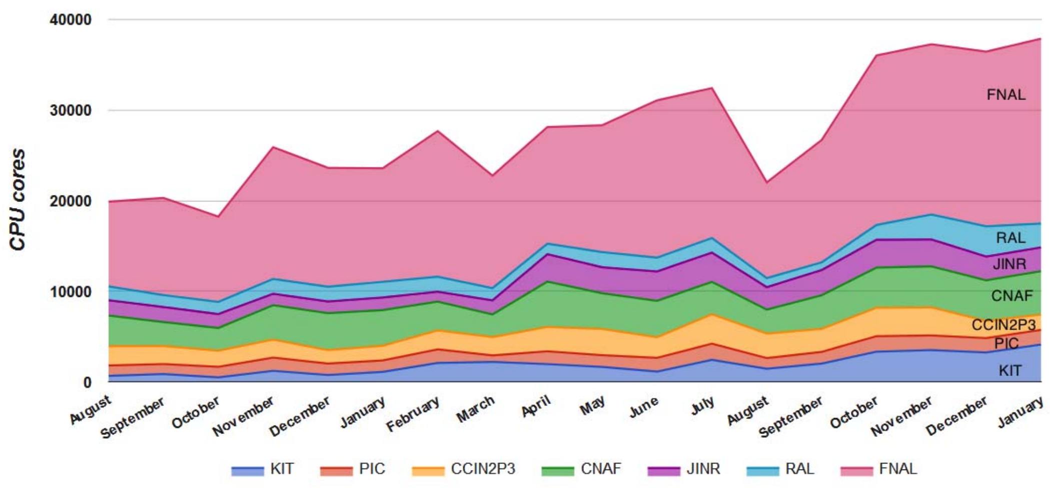

Figure 3. Aggregated number of CPU cores allocated to multicore pilots running at all CMS Tier-1 sites (monthly averages 2015 and 2016).

Table 1. Compute Element and batch system technologies of CMS supporting sites where multicore pilot submission has been enabled.

Compute Element	Batch system	Example sites (CMS naming convention)
CREAM	PBS	T1_ES_PIC, T1_RU_JINR, T2_ES_CIEMAT
CREAM	Grid Engine	T1_DE_KIT, T1_FR_CCIN2P3, T2_UK_London_IC
CREAM	LSF	T1_IT_CNAF, T2_IT_Pisa, T2_CH_CERN
CREAM	HTCondor	T2_IT_Bari, T2_FR_GRIF_LLR
ARC	SLURM	T2_CH_CSCS, T2_EE_Estonia
ARC	HTCondor	T1_UK_RAL, T2_FR_GRIF_IRFU
HTCondor	HTCondor	T1_US_FNAL, T2_CH_CERN, T2_US_Nebraska
HTCondor	PBS	T2_US_Florida, T2_US_Purdue
HTCondor	SLURM	T2_US_Vanderbilt

4. Results and performance

4.1. Running multi-threaded jobs

Multi-threaded applications were first used in the execution of Tier-0 prompt data reconstruction at the beginning of LHC Run 2 [9]. Proton-proton data reconstruction jobs have run on 4 CPU cores during the 2015 and 2016 data taking periods, while 6 and 8-core jobs were employed on heavy-ion collision runs for the reconstruction of such events at the end of 2015 [6].

Since late 2015, CMS has been successfully employing multi-threaded jobs for standard work in certain tasks, such as data and Monte Carlo (MC) reconstruction running at Tier-1 and Tier-2 sites. Still, the majority of the workload executed by CMS in 2016 has been run in single-core mode, for MC generation and analysis jobs. Figure 5 shows the weekly average number of cores allocated to running jobs at Tier-1 and Tier-2 sites during 2016, as a function of the number of cores requested per job. It can clearly be observed that multi-core workload has been steadily increasing during 2016.

Once 2016 the data taking was ended, the full accumulated dataset was re-reconstructed in

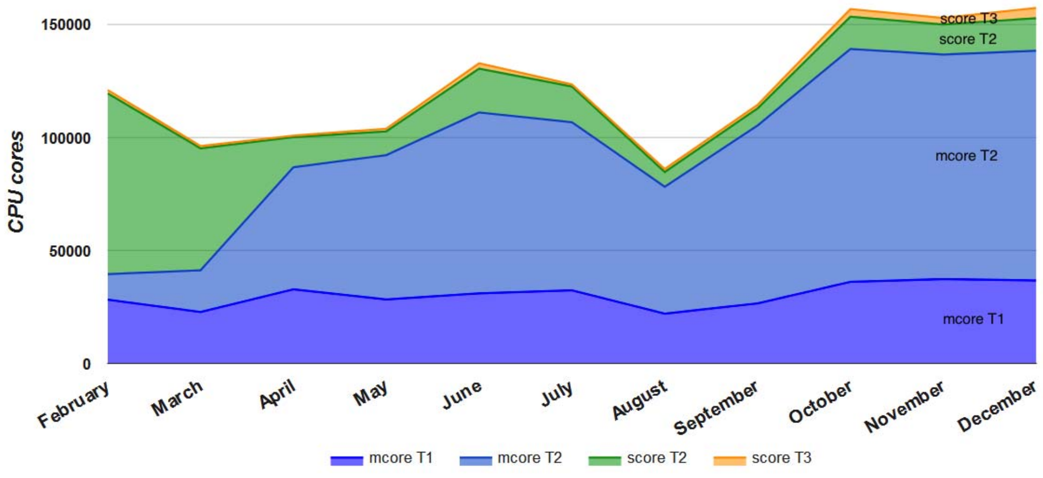

Figure 4. Aggregated number of CPU cores allocated to running CMS global pool pilots as a function of pilot type (single core or multicore) and site tier level (monthly averages in 2016).

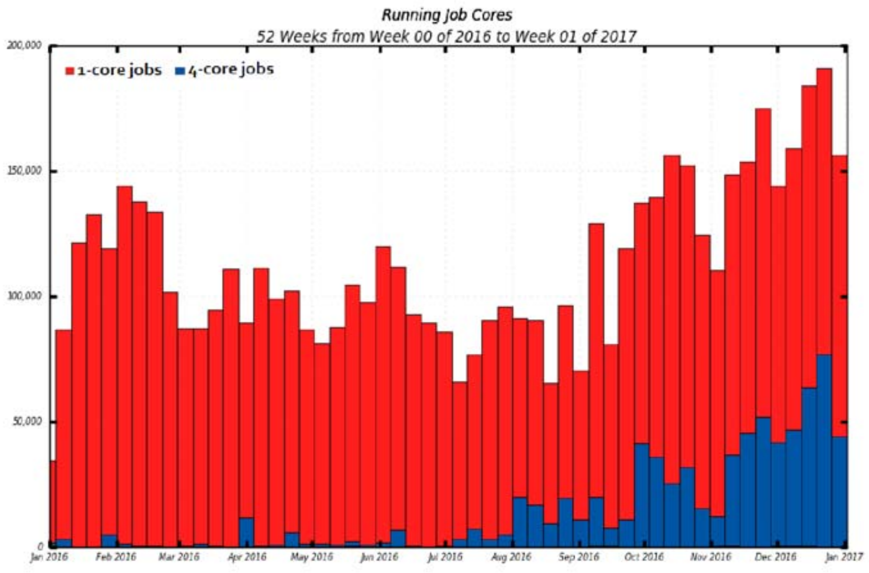

Figure 5. Weekly average number of CPU cores employed by CMS running jobs at Tier-1 and Tier-2 sites as a function of core count during 2016.

approximately one month, starting in late September 2016, with contributions from all multi-core enabled sites (including the HLT farm used as opportunistic resource [10]). Following that, MC datasets were produced in the later months of 2016 and into early 2017. Peaks of 100,000 CPU cores dedicated to these 4-core tasks have been observed during these periods of intense activity, when the global pool resources were mostly being successfully employed to execute multi-core tasks.

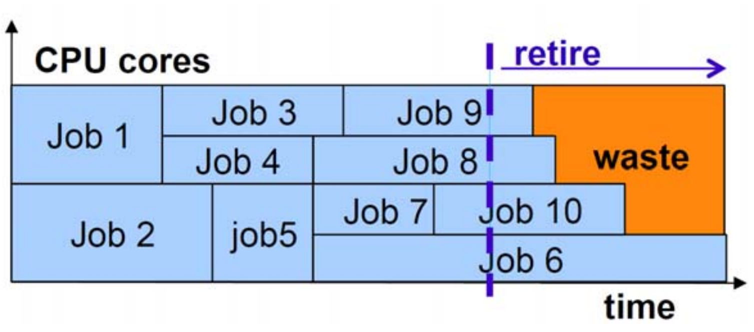

Figure 6. Wastage in CPU usage due to multicore pilot draining after it stops accepting any new payloads (retire mode).

4.2. Observed Scheduling Performance

During these months of operation, the main sources of scheduling inefficiency of this multi-core pilot model have been identified. Inherent to multi-core pilots internally executing single-core payloads is the inefficiency related to draining glideins in retire mode. After a fraction of their maximum allowed lifetime, pilots stop accepting new jobs to avoid the pilot itself, and subsequently all the running payloads, being killed by the local batch system due to exceeding running time limitations. As payloads finish, unused cores are observed while the pilot completes its draining, see Figure 6. This effect has been quantified to correspond to approximately 5% of the total pool cores.

Another source of CPU inefficiency comes from memory constrained slots. As partitionable slots, and upon payload request, pilots may create internal slots with higher than average assigned memory. This flexibility in resource allocation may result however in remaining available CPU cores idling, as they can not be used by regular average jobs due to their insufficient memory (typically below 2GB). This has been determined not in general an issue, except for some particular periods of time and at some sites, when, for example, a fraction of the resources were dedicated to running heavy-ion collision data reconstruction jobs.

Finally, a third category has been identified, corresponding to usable but unmatched slots. The effect of certain slots not being matched to any payload is caused by a variety of reasons, such as decreasing workload and restrictive site white-listing, which results in a lack of payloads which request these resources. The slow response from the job-submit components may also cause this starvation of workload capable of running in these empty slots. Finally, when the pool is saturating all its available resources, some scaling limitations may be observed, linked to slow slot status updates and long negotiation cycles, which also prevent these empty slots from being discovered and matched by suitable payloads. All of these effects combined correspond to an average additional wastage of approximately 5% of the pool cores.

Some of the causes of CPU usage inefficiency from the third group were also affecting the former single-core pilot model, still present for the remaining 10% of the CMS Global Pool resources managed by single-core pilots. For example, an idle single-core pilot will remain alive for a certain amount of time (an adjustable parameter usually set at 20 minutes) before returning the allocated slot, in order to allow the negotiator to run several matchmaking cycles looking for a suitable payload. The net effect is very small, with CPU wastage for single-core pilots being below 5% of the total managed cores on average.

5. Scheduling efficiency optimization

As described in the previous section, CMS has transitioned to using multi-core pilots for resource allocation. As Figure 5 shows, the overall CMS workload during 2016 has been still dominated by single-core jobs, although multi-threaded workflows are now employed for centralized data and MC reprocessing. In addition, centralized MC production and potentially a fraction of analysis jobs will be also run in multi-core mode in 2017. This means that the fraction of the total load being run as single-core is decreasing, which will help in reducing the draining wastage effect.

The wastage in usable but unclaimed cores is being reduced by means of continuously evolving to a more efficient and faster job-to-pilot matchmaking. This comes from incremental improvements on the CMS Global Pool central manager and schedd performance, removing previous scalability limitations, as a result of the Submission Infrastructure team being in close contact with HTCondor and GlideinWMS developer teams [11]. Improved workflow management and job-submit components, as well as further automation of the system in order to reduce the need for human intervention, are contributing to produce a more continuous flow of requests being injected to the pool, which produces more reliable and efficient usage of the global pool slots. Moreover, overloaded and unresponsive submit-nodes, which cause increasing negotiation cycle times, are now being detected and automatically reconfigured or removed from the pool.

An improved management of pilot pressure on sites can be achieved by better tuning resource-based response to workload pressure from the GlideinWMS frontend and factories, which is work in progress together with the GlideinWMS developers team. Removal of out-of-synchronization excess pilots (due to late job-pilot binding) before they start to run, during irregular workload submission phases, can potentially reduce the amount of unused CPU cores.

A more efficient matchmaking to running pilots is possible by implementing a depth-wise filling of the resources, opposed to breadth-wise filling pattern employed by HTCondor negotiator by default. This has been worked on in close collaboration with the HTCondor developers team, and there is currently a prototype deployed in the global pool testbed. The introduction of resource requests expressions now allowing for the possibility of running re-sizable jobs, which can be allocated a variable number of CPU cores, and could for example take all idle cores in the matching pilot (prototype in global pool test-bed), hence also contributing to increased CPU allocation and usage efficiency. Finally, a proposal for the removal of pilots in a long draining stage, profiting from automated job resubmission, has been discussed. An active *pruning* of the retiring pilot pool could also lead to improved overall CPU usage results.

6. Conclusions

The CMS experience with multi-core pilots goes back to 2014, when they were introduced to Tier-1 sites. Since then, the multi-core pilot model has been fully expanded to the majority of CMS CPU resources at its supporting Tier-1 and Tier-2 sites, forming multi-core pilot Global Pool. Multi-threaded job submission has been demonstrated to work successfully along 2016, as many centralized production tasks have been switched to be executed as multi-core jobs. A number of causes of CPU allocation inefficiency have been identified and understood. Work is currently ongoing in order to mitigate them from multiple fronts.

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 d'Informació Científica (PIC) is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universitat Autonoma de Barcelona.

References

[1] Jones C *et al*, CMS Event Processing Multi-core Efficiency Status, to be published in these proceedings
[2] Worldwide LHC Computing Grid `http://wlcg.web.cern.ch/`
[3] `http://research.cs.wisc.edu/htcondor/`
[4] `http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS/doc.prd/`
[5] Sfiligoi I *et al*, The pilot way to Grid resources using glideinWMS, Proc. WRI World Congress on Computer Science and Information Engineering vol. 2 pp. 428-432
[6] Perez-Calero Yzquierdo A *et al*, Exploiting multicore compute resources in the CMS experiment, J. Phys. Conf. Ser. 762 (2016) 012018
[7] Perez-Calero Yzquierdo A *et al*, Evolution of CMS workload management towards multicore job support, J. Phys. Conf. Ser. 664 (2015) 062046
[8] Perez-Calero Yzquierdo A *et al*, CMS multicore scheduling strategy, J. Phys. Conf. Ser. 513 (2014) 032074
[9] Hufnagel D *et al*, The CMS Tier0 goes Cloud and Grid for LHC Run 2, J. Phys. Conf. Ser. 664 (2015) 032014
[10] Colling D *et al*, Using the CMS High Level Trigger as a Cloud Resource J. Phys. Conf. Ser. 513 (2014) 032019
[11] Balcas J *et al*, Stability and scalability of the CMS Global Pool: pushing HTCondor and GlideinWMS to new limits, to be published in these proceedings