EPJ Web of Conferences **337**, 01210 (2025) https://doi.org/10.1051/epjconf/202533701210
CHEP 2024

Optimization of distributed compute resources utilization in the CMS Global Pool

Marco Mascheroni[1*], *Antonio* Pérez-Calero Yzquierdo[2,3**], *Hyunwoo* Kim[4], *Florian* Von Cube[5], *Vaiva* Zokaite[5], and *Frank* Würthwein[1] on behalf of the CMS Collaboration

[1]University of California San Diego, La Jolla, CA, USA
[2]Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
[3]Port d'Informació Cientifica (PIC), Barcelona, Spain
[4]Fermi National Accelerator Laboratory, Batavia, IL, USA
[5]European Organization for Nuclear Research (CERN), Geneva, Switzerland

Abstract.
The CMS Submission Infrastructure is the primary system for managing computing resources for CMS workflows, including data processing, simulation, and analysis. It integrates geographically distributed resources from Grid, HPC, and cloud providers into federated pools managed by HTCondor and GlideinWMS, for a total of around 500k CPU cores. This system dynamically manages workloads based on priorities defined by the collaboration. Additionally, CMS scheduling strategies must be flexible to handle multiple concurrent workloads while considering changing processing demands and resource availability from various providers.
Efficient utilization of vast amounts of distributed compute resources is a key element for the success of the scientific programs of the LHC experiments. Optimizing the system is essential to maximize resource efficiency and fully utilize the distributed computing power. The CMS Submission Infrastructure team thus systematically investigates sources of inefficiency in workload scheduling to reduce their impact. In addition, a strategy of pilot overloading has been introduced to compensate for other inefficiency sources, thereby optimizing resource utilization and enhancing computational throughput.

1 Introduction

The Compact Muon Solenoid (CMS) experiment [1] is one of the major experiments at the Large Hadron Collider (LHC) at CERN. Designed to study a wide range of physics phenomena, including the Higgs boson, dark matter candidates, and other beyond-the-Standard-Model particles, CMS relies on an extensive computational infrastructure to process and analyze the vast amounts of data collected by the detector. The CMS computing model is structured around a distributed system, leveraging computing resources from multiple providers for its data processing and Montecarlo simulation needs.

The CMS Submission Infrastructure (SI) group is part of the CMS Offline and Computing coordination area and is responsible for acquiring and managing computing power for CMS

*e-mail: marco.mascheroni@cern.ch
**e-mail: aperez@pic.es

operations from over 70 Grid sites from the Worldwide LHC Computing Grid (WLCG) [2], supplemented by non-Grid resources such as those from high-performance computing (HPC) centers and cloud providers. The team operates a set of federated resource pools (Figure 1), including the Global Pool [3], a critical infrastructure where essential computing tasks such as data reconstruction, simulation, and analysis are performed. This infrastructure, which is managed through HTCondor [4], plays a key role in ensuring that computing resources are effectively utilized to meet the needs of the CMS collaboration. The federation of these resources is largely achieved through GlideinWMS [5], a pilot-based workload management system that dynamically provisions worker nodes on demand, enabling flexible and scalable compute provisioning.

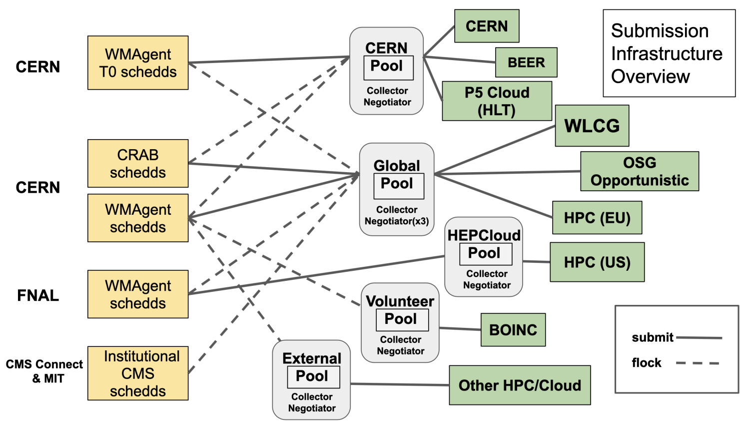

Figure 1. CMS SI current configuration, including multiple federated pools of resources allocated from diverse origins (green boxes) and sets of distributed job schedulers (schedds).

2 The CMS SI multicore pilot model

The CMS SI operates in a late-binding model, where resources are primarily acquired via *pilot jobs* [6] submitted to WLCG sites' Computing Elements (CEs). The minimal configuration is based on 8-core 16 GB of memory and 48h long slots. However, this model is capable of also utilizing non-standard slots, such as other fixed slot sizes, high-memory allocations or even allocation of whole worker nodes, depending on the workload's requirements and resource providers policies, as shown in Figure 2.

Internally, pilot jobs configure HTCondor an Execution Points (EP) [7], configured as partitionable slots. Within a pilot, multiple payload jobs of diverse types can be scheduled and executed, both in parallel and consecutively, maximizing allocated resource utilization. Figure 3 shows an example of this, highlighting inefficiencies in resource usage, which can stem from two broad categories: *scheduling* and *payload* inefficiencies.

Scheduling inefficiencies stem from system constraints and assumptions about resource usage built into the allocation system. For example, workloads may exceed the usual ratio of 2GB RAM per core, leaving CPU cores idle. Additionally, the limited pilot job execution time introduces inefficiencies during the draining phase, as running jobs must finish before new ones start. Despite these challenges, the CMS SI model maintains and average 95% scheduling efficiency at major WLCG sites, including Tier-1 centers and large Tier-2s.

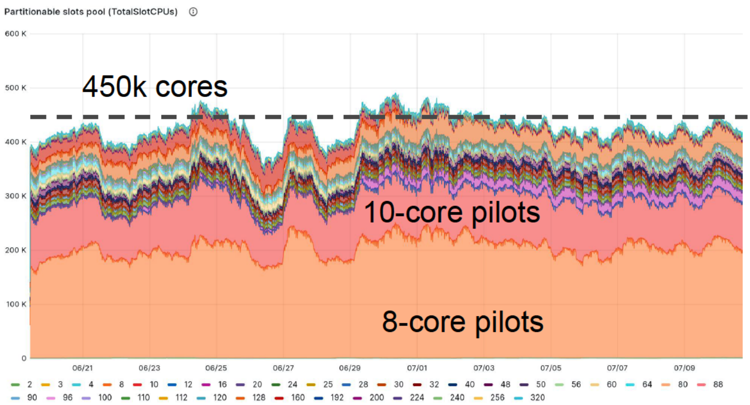

Figure 2. Composition of the CMS Global and CERN pool in terms of pilot sizes.

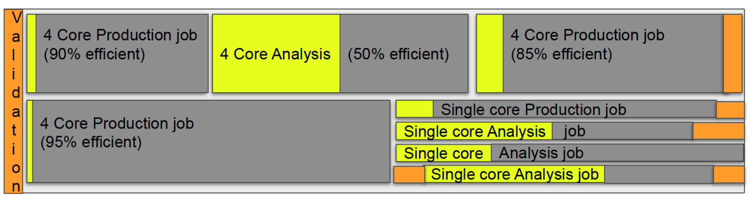

Figure 3. A diagram of a typical CMS pilot job (8-core and 48h long) executing multiple payloads, which are broadly categorized into *analysis* (user driven) and *production* (centrally managed simulation or data processing campaigns). Scheduling inefficiencies are indicated in orange, while inefficiencies originated from the payload jobs internal utilization of the processor are shown in yellow.

Payload inefficiencies arise when allocated CPU cores remain underutilized. Causes include task bootstrapping and data stage-in/out phases, which introduce idle time, as well as I/O-bound jobs that rely on heavy I/O operations or remote data access. CMS analysis jobs may also be inefficient due to diverse, sometimes unoptimized for their resource request, code. Additionally, centrally driven simalution campaigns in the form of StepChain workflows [8], which link multiple executables into a single job, improve turnaround time and reduce intermediate storage but can lead to inefficiencies due to varying resource demands across execution steps.

From the perspective of WLCG sites as resource providers, inefficiencies are not categorized by type or cause. Instead, they measure overall CPU efficiency for each CMS task as the fraction of time the CPU cores were actively used relative to the total slot allocation time. Each site records and reports total CPU time and allocated walltime to a common accounting portal [9]. Based on this data, an average monthly CPU efficiency metric is calculated and published. In the next section, a *pilot overloading* strategy, designed to recover unused CPU cycles and further enhance overall resource utilization, will be described.

3 Pilot overloading

Given the irreducible nature of some of the CPU inefficiency causes described in the previous section, in particular concerning payload inefficiencies, a different approach to improving overall CPU utilization was conceived by CMS SI. The new strategy, named pilot overloading, involves modifying pilots to accept additional payload jobs within the same allocated resources. Since the CMS SI team directly manages pilot configurations, this strategy was straightforward to implement and test.

It is important to emphasize that as a key principle, pilot overloading is not aimed at gaining opportunistic cycles but only to recover unused CPU resources. Therefore, CMS SI introduced only a moderate level of overloading by rescaling by +25% the CPU core count and memory parameters in the internal HTCondor EP configuration. This allows pilots effectively to advertise two additional cores per pilot in the standard 8-core pilot configuration to the CMS Global Pool. These extra cores can then be matched, for example, for the execution of additional single-core analysis or production payloads.

In order to ensure that pilot overloading would not be exceeding the total memory allocated to CMS multicore pilots, an analysis of memory usage for pilots running at Tier-1 sites was conducted before enabling it. Results indicate that at least 20% of the partitionable slot memory remains unscheduled to any payload, even when pilots are fully occupied (Figure 4 left). Furthermore, considering the actual utilization by payloads during their execution in dynamic slots, an average memory utilization was found to be typically below 50% (Figure 4 right). These observations conclude that there is no memory constraint for a moderate overloading strategy such as the one proposed.

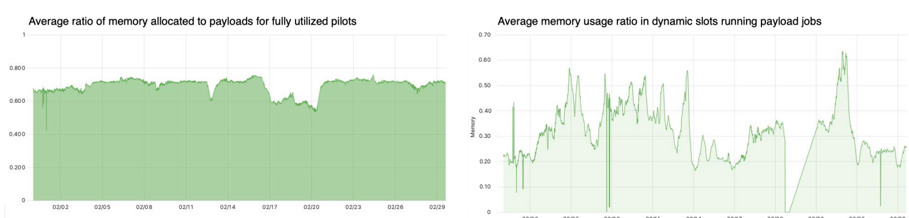

Figure 4. Memory utilization in CMS pilots in terms of memory allocated to dynamically carved internal slots in relation to total memory claimed by the pilot (left), and average actual memory utilization by payload jobs inside their dynamic slots (right).

Figure 5 illustrates an example of the application of the pilot overloading strategy to a real case, namely a CMS pilot managing a whole-node slot allocated at FNAL Tier-1 site. It can be observed that a 128-core slot is successfully enlarged by the pilot up to 160 cores, which then overloads the resources with CMS tasks from diverse CMS MC production campaigns (colors indicate the number of cores associated with running payload jobs of each type). In this example, the pink section represents a specific workflow that achieved an average CPU utilization efficiency of 75%, thus leaving 25% of the CPU cycles unused, and similarly for the rest of tasks executed concurrently. Overloading the HTCondor EP with additional payload jobs, as in this example, therefore improves the overall compute power utilization at the node level, ultimately increasing CMS throughput in terms of simulated LHC collision events.

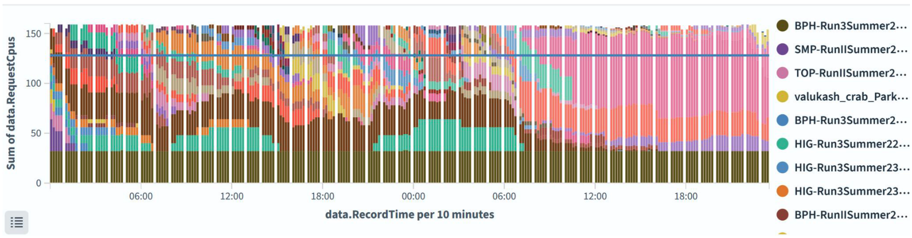

Figure 5. Example of whole node pilot overloaded from 128 to 160 cores, as described in the main text.

4 CPU efficiency improvement results

Early tests of the pilot overloading strategy, conducted in 2023 at three CMS sites, already showed promising indications of its benefits. A set of active and reliable sites with multiple Computing Element (CE) were selected, and then overloading was only applied to pilots submitted to one CE in each case, leaving the others unchanged to serve as reference. As Figure 6 shows, a noticeable increase in CPU utilization was measured in every case, while no negative impact could be observed in terms of successful job completion or pilot stability. Encouraged by these initial results, CMS expanded the deployment of pilot overloading at the beginning of 2024.

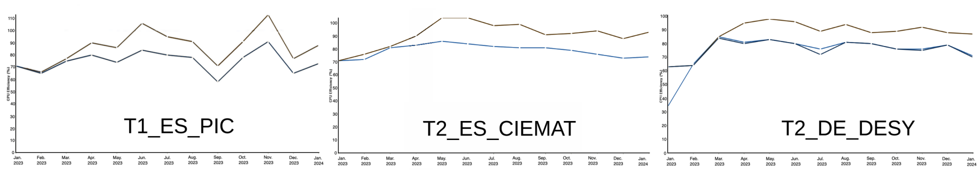

Figure 6. Evolution of CPU efficiency measured monthly in 2023 for CMS tasks, segregated by CE, at PIC (left), CIEMAT (center) and DESY (right) sites. Results for CEs receiving pilots with overloading enabled (in brown) present a clear improvement in all cases.

Starting in January 2024, CMS SI enabled pilot overloading at a broader set of resource providers, including all Tier-1 sites and a selection of high-performing Tier-2 sites. The Tier-2 sites were chosen based on their demonstrated scheduling efficiency, at or above the 95% level. Approximately 50% of the resources in the Global Pool remained unchanged, allowing for performance comparisons between standard and overloaded pilots at each site.

To assess the impact of pilot overloading, total walltime and used CPU time data were collected from pilot logs, which allows the comparison with CPU efficiency reported by resource providers, in both cases for overloaded and non-overloaded pilots. Results consistently show a noticeable improvement in CPU utilization across all sites. Importantly, moderate pilot overloading did not lead to an increase in job failure rates, indicating that this strategy successfully improves efficiency without introducing instability. Figure 7 illustrates the observed efficiency improvements from diverse sites. Measured performance gains validate pilot overloading as an effective method for recovering underutilized CPU cycles, supporting also a broader adoption of this strategy within the CMS SI.

Site Name	Normal Efficiency	Overloaded Efficiency	Efficiency increase
T1_FR_CCIN2P3	84.57%	95.78%	11.21%
T1_IT_CNAF	79.81%	84.62%	4.81%
T1_UK_RAL	72.41%	85.00%	12.60%
T2_BE_IIHE	77.21%	89.17%	11.96%
T2_DE_RWTH	55.68%	78.01%	22.33%
T2_EE_Estonia	73.81%	85.47%	11.66%
T2_ES_CIEMAT	72.65%	88.40%	15.75%
T2_IT_Bari	74.48%	78.87%	4.39%
T2_IT_Legnaro	75.77%	85.88%	10.11%
T2_IT_Rome	60.25%	73.38%	13.13%
T2_UK_London_IC	66.14%	72.94%	6.80%
T2_US_MIT	63.74%	70.61%	6.87%

Figure 7. Example of CPU efficiency gains for overloaded CMS pilots at multiple sites (overall results for the month of September 2024).

5 Impact on event rates

Maximizing event processing throughput for a given set of compute resources is the ultimate goal for CMS SI. The effect of pilot overloading on event processing rates was analyzed using performance data from a diverse set of workloads executed across multiple sites over several months.

One large-scale example studied was a StepChain simulation workflow, consisting of approximately 450,000 jobs. The analysis of event processing rates for these jobs (Figure 8) revealed significant variability, both across different sites and within the same site for jobs of the same workflow. It can be observed that pilot overloading may induce a slight decrease in throughput when considering individual jobs, however this effect is in general smaller than the intrinsic dispersion observed between jobs at a given site.

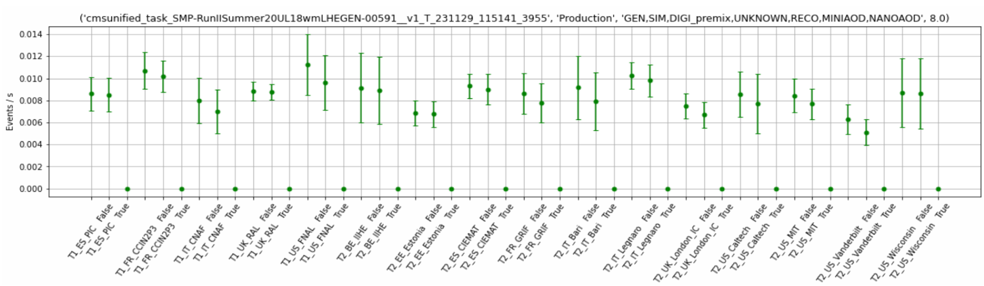

Figure 8. Average event processing rate for a CMS simulation workflow executed at multiple sites via overloaded and non-overloaded pilot jobs.

In order to measure overall event processing rates, an additional control study was conducted using pilots fully loaded with payload jobs of a single workflow (MinBias production) across multiple sites. Total event production rates were measured using both overloading and non-overloading pilots executed on the same worker nodes. The results (shown in Figure 9) indicate that pilot overloading increased the total event throughput. While individual process event rates were slightly reduced due to resource contention, this was compensated for by the additional concurrent processes, with a net overall gain in the number of events processed per pilot. It is therefore demonstrated that pilot overloading effectively improves overall CPU efficiency utilization while also increasing simulated events production for the same set of resources.

Site	Processor	On (Evt/s)	Off (Evt/s)	Evt/s increment %
T2_BE_IIHE	AMD EPYC 7452 32-Core Processor	3,14	2,82	11,35
T2_DE_RWTH	AMD EPYC 7543 32-Core Processor	2,89	2,49	16,06
T2_ES_CIEMAT	Intel(R) Xeon(R) Gold 5318Y CPU @ 2.10GHz	4,31	3,58	20,39
T2_IT_Bari	AMD EPYC 7413 24-Core Processor	2,96	2,81	5,34
T2_IT_Legnaro*	AMD EPYC 7282 16|7413 24-Core Processor	2,98	2,74	8,76
T2_UK_London_IC	Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz	1,70	1,61	5,59
T2_US_UCSD	AMD EPYC 7662 64-Core Processor	2,42	1,63	48,47
T2_US_Vanderbilt	Intel(R) Xeon(R) CPU E5-2420 0 @ 1.90GHz	1,30	1,05	23,81

Figure 9. Event processing rates for MinBias production jobs running in pilots with overloading enabled or disabled, for a variety of computing sites and processor types.

6 Conclusions and future work

The CMS SI team has introduced a moderate pilot overloading strategy, which has been measured to be an effective method for recovering idle CPU cycles, leading to a measurable increase in resource utilization. By increasing internal pilot matchmaking to simulated 25% more CPU and memory, between 5% to 20% of otherwise unused CPU cycles can be regained, depending on computing sites. This effectively has resulted in boosting computational capacity for CMS by an equivalent of approximately 30,000 additional CPU cores. No negative impact has been observed from the site perspective, nor has job error rates been increased, confirming that pilot overloading is a safe and practical optimization for improving overall efficiency.

Future work will focus on expanding this approach to a larger fraction of the compute resources supporting CMS. Expected developments on cgroups [10] anticipate new features to be available in HTCondor which will enable the CMS SI team further, more controlled, application of the pilot overloading strategy presented in this report.

Acknowledgements

This work was partially supported by the Spanish Ministry of Science and Innovation under grants PID2020-113807RA-I00 and PID2022-142604OB-C21, which include FEDER funds from the European Union, and by the US National Science Foundation under Grant No. 2121686 and under Grant No. 2030508.

References

[1] CMS Collaboration, “The CMS experiment at the CERN LHC", JINST **3** (2008) S08004, doi:10.1088/1748-0221/3/08/S08004.

[2] The Worldwide LHC Computing Grid http://wlcg.web.cern.ch, accessed February, 2025.
[3] J. Balcas et al. “Using the glideinWMS System as a Common Resource Provisioning Layer in CMS”, J. Phys.: Conf. Ser. **664** 062031 (2015).
[4] The HTCondor Software Suite public web site, https://research.cs.wisc.edu/htcondor/index.html, accessed February, 2025.
[5] The Glidein-based Workflow Management System, https://glideinwms.fnal.gov/doc.prd/index.html, accessed February, 2025.
[6] I. Sfiligoi et al. “The Pilot Way to Grid Resources Using glideinWMS", 2009 WRI World Congress on Computer Science and Information Engineering, Los Angeles, CA, USA, 2009, pp. 428-432, doi: 10.1109/CSIE.2009.950.
[7] HTCondor Execution Point, https://htcondor.readthedocs.io/en/latest/admin-manual/introduction-admin-manual.html, accessed February, 2025.
[8] A. Malta et al. “TaskChain vs StepChain”, https://github.com/dmwm/WMCore/wiki/TaskChain-vs-StepChain, accessed February, 2025.
[9] EGI Accounting Portal, https://accounting.egi.eu/, accessed February, 2025.
[10] Control Group v2 https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v2.html, accessed February, 2025.