Reaching new peaks for the future of the CMS HTCondor Global Pool

A. Pérez-Calero Yzquierdo[1,2,*], *M.* Mascheroni[3], *M.* Acosta Flechas[4], *J.* Dost[3], *S.* Haleem[5], *K.* Hurtado Anampa[6], *F. A.* Khan[4], *E.* Kizinevič[7], and *N.* Peregonov[4], 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 California San Diego, La Jolla, CA, USA
[4]Fermi National Accelerator Laboratory, Batavia, IL, USA
[5]National Centre for Physics, Islamabad, Pakistan
[6]University of Notre Dame, Notre Dame, IN, USA
[7]European Organization for Nuclear Research, Meyrin, Switzerland

Abstract. The CMS experiment at CERN employs a distributed computing infrastructure to satisfy its data processing and simulation needs. The CMS Submission Infrastructure team manages a dynamic HTCondor pool, aggregating mainly Grid clusters worldwide, but also HPC, Cloud and opportunistic resources. This CMS Global Pool, which currently involves over 70 computing sites worldwide and peaks at 350k CPU cores, is employed to successfully manage the simultaneous execution of up to 150k tasks. While the present infrastructure is sufficient to harness the current computing power scales, CMS latest estimates predict a noticeable expansion in the amount of CPU that will be required in order to cope with the massive data increase of the High-Luminosity LHC (HL-LHC) era, planned to start in 2027. This contribution presents the latest results of the CMS Submission Infrastructure team in exploring and expanding the scalability reach of our Global Pool, in order to preventively detect and overcome any barriers in relation to the HL-LHC goals, while maintaining high efficiency in our workload scheduling and resource utilization.

1 The CMS Submission Infrastructure

The Submission Infrastructure (SI) team runs the computing infrastructure in which processing, reconstruction, simulation, and analysis of the CMS experiment physics data takes place. GlideinWMS [1] and HTCondor [2] are employed in order to build a Global Pool [3] of computing resources where these tasks are executed. The SI team also works on uncovering and solving operational limitations while preparing for future scales and requirements, the integration of new resource types and serving as liaison between the CMS collaboration and the HTCondor and GlideinWMS development teams, to stimulate cooperation and integrate new features as needed by the experiment.

Achieving a successful scheduling of all CMS workloads requires taking in consideration their diversity and prioritization, along with their resource requirements. What constitutes

*e-mail: aperez@pic.es

successful scheduling is however subject to interpretation, namely ensuring that all available resources are efficiently used, a fair share of resources between groups of workload submitters, or the completion of tasks according to their prioritization, and with minimal job failures and manual intervention. The SI team is dedicated to ensuring that these potentially diverging goals are reasonably covered simultaneously.

The infrastructure employed by the SI team has evolved over the recent years towards a model of multiple interconnected HTCondor pools of resources (see Fig.1). Specialized submission nodes (*schedds*), are deployed at CERN and FNAL. While being primarily attached to one of the pools, these schedds can however submit work into other federated pools, when resource demands are not covered by the primary one (a strategy known as *flocking*). For each pool, a Central Manager node is configured, running the *collector* and *negotiator* processes, services aggregating the latest status information of all slots in the pool and processing the matchmaking of job resource requests to open pool slots, respectively.

The center piece of the model, the Global Pool, mainly acquires resources via the submission of GlideinWMS pilot jobs to WLCG sites [4]. Pilot jobs, equivalent to resource requests on remote grid Compute Elements, are submitted by a number of distributed pilot *factories* on behalf of the Front-End (FE) service, which processes the workload queue content to determine where resources should be acquired. The main pool can also include locally instantiated processing nodes (*startds*), for example via DODAS [5] or BOINC [6] (for the CMS@Home [7] project), and opportunistic resources such as the CMS High Level Trigger (HLT) farm [8] for offline processing when not in use for data taking. A number of CMS sites have also expanded their computing capacity by locally aggregating resources from High Performance Computing (HPC) facilities in a transparent way for CMS, as exemplified by the CNAF [9] and KIT [10] cases, where pilot jobs arriving at the sites' compute elements are in turn rerouted to the HPC cluster batch system. This approach follows the CMS strategy to employ HPC resources whenever available [11]. A second HTCondor CERN pool includes the on-site CMS resources, as well as opportunistic BEER [12] and Azure [13] cloud slots. The HEPCloud [14] pool, joining cloud and HPC resources in the USA (e.g. at NERSC [15]), is managed from the FNAL site. Finally, additional external pools can be federated into the SI by enabling flocking from the CMS schedds.

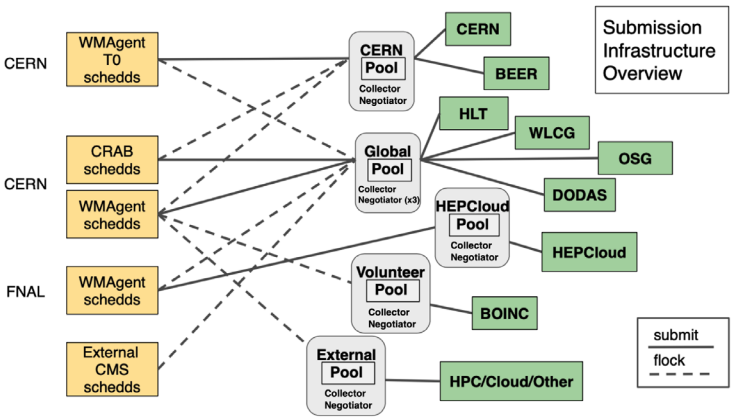

Figure 1. CMS SI current configuration, including multiple federated pools of resources allocated from diverse origins (green boxes) and sets of distributed job schedulers (schedds), handling the Tier 0 and centralized production workloads (WMAgent), as well as analysis job submission (CRAB).

2 Scalability requirements for the Global Pool

The Global Pool has been continuously growing in size over the last years, driven by the increasing resource requests during the LHC Run 2 (2015 to 2018), and through the Long-Shutdown 2 period (from 2019 to the present date), with the progressive incorporation of opportunistic and non-standard resources. Indeed, opportunistic, HPC and Cloud resources have been added to the Global Pool, currently aggregating over 300,000 CPU cores routinely, in an increasing proportion compared to the standard Grid sites slots,

Considering the growing scales of data to be collected by CMS in the LHC High Luminosity (HL-LHC) phase, driven by increasing detector trigger rates and event complexity, CMS published in 2020 its estimated computational needs in the future [16] (see Fig. 2). As shown, a dramatic increase is anticipated for the second half of the 2020s compared to the current capacity and the moderate growth expected for the Run 3 years.

As presented also in Fig. 2, the CMS CPU demands will be at least a factor 4 or 5 higher than a simple extrapolation of the current capacity into the HL-LHC phase at fixed cost (i.e. from continuously evolving technologies at a rate of 10% to 20% yearly performance improvement). In addition to the expansion of the CMS infrastructure to new resource types and providers, the gap is also foreseen to be reduced by a number of R&D projects currently being developed at the CMS software and data formats level [17].

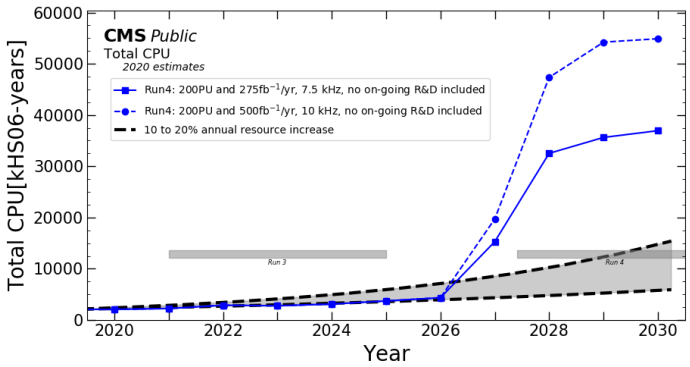

Figure 2. CMS CPU resource needs projections into the HL-LHC era, computed according to two scenarios of detector trigger rates and assuming a value of 200 for the pile-up (PU, the number or proton-proton collisions per LHC bunch crossing event).

Uncertainties about the future CMS CPU needs propagate into the SI domain, with basic matters such as the total amount of resources that SI will need to manage, how many simultaneous tasks would be executed or how many job schedulers will we have to employ in order to execute future workloads and make efficient use of such resources. Making predictions even harder, compute resources growth will unlikely come from increasing capacities at Grid sites, therefore further heterogeneity and specialization have to be considered as well.

3 Scalability tests

In order to preventively detect and solve potential scalability limitations, the SI group regularly conducts dedicated tests on the capacity for our infrastructure to grow while maintaining high efficiency and stability. Given the continuous evolution of our underlying tools, these

tests have been executed recurrently over the last years (original tests described in [20]), involving a testbed HTCondor pool which is stressed to the limits of its capacity. As a canonical test, the number of simultaneously running tasks is progressively enlarged until the pool's performance degrades, for example in terms of reduced matchmaking efficiency or matchmaking processing time. These experiences are subsequently presented and discussed with the HTCondor and GlideinWMS development teams.

In previous test cycles, it was typically found that the collector of the pool was the limiting factor. Specifically, the collector capacity to process slot information updates saturates when the pool and the number of running tasks grow (determined by the collector daemon core duty cycle reaching 100%). The collector performance limits have been again directly put to the test in the present scalability evaluation. Moreover, in addition to testing the capabilities of the latest HTCondor version available, we also want to compare test setups where the central manager of the pool is hosted on virtual versus physical machines. Finally, a test of the complete infrastructure was launched, which allows us to explore any limitations also on the GlideinWMS layer and in the available capacity of the scheduling nodes. These tests will be described in the following sections.

3.1 Testing setup

Our tests have explored the capacity and performance of a setup based on HTCondor 8.9.10, latest available version in the development series, while the test FE and factory services (requesting and submitting pilots) were running GlideinWMS 3.7.2.

In order to estimate the potential impact of the virtualization layer in degrading the performance of our collector, two hosts were employed:

- vocms0809: a virtual machine (VM), with 24 CPU cores, hosted on a node based on the Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz processor, with 256 GB of RAM and 4 SSDs of 1.2 TB each. This node, in spite of being a VM, was not co-hosted with any other service at the time of the tests, so any performance reduction should come from the hypervisor technology (Openstack).
- vocms0803: a physical machine (PM), with 32 cores, based on Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz processor, with 187 GB of RAM and 2 SSDs of 1.8 TB each.

Collector daemons were deployed on both nodes with an identical configuration. Finally, in order to provide job submission capacity to the test pool, a number of schedds (initially 10, later on 15) were installed on VMs running with 32 cores, 57 GB of RAM and 512 GB of High IO disk.

3.2 Collector performance comparison with condor_advertise

In the first stage of the tests, *condor_advertise* commands [18] were launched, which create a stream of slot update messages (*ads*) aimed at each of the two collectors. The total rate of slot updates submitted in parallel could be tuned by selecting the number of simultaneous advertising schedds, the number of parallel updates per schedd, the number of cycles and the sleep time between cycles. Communication was performed via UDP packets, which would accumulate at each node up to a total buffer size of 2 GB in both cases. Both collectors were tested, gradually increasing their stress, with slot update rates ranging from 60 to 180 ads/s.

The results of these tests, comparing the collector performance on the VM versus the PM, are summarized in Fig. 3. A total of 4 stress test rounds were performed, corresponding to 60, 120, 180 and 90 ads/s respectively. The performance of the PM hosted collector is clearly superior, as saturation is only observed in the most extreme test, at 180 ads/s, in contrast to

the VM hosted collector, reaching saturation in all cases, leading to UDP packets processing errors, which results in missed slot updates. At 120 ads/s, the PM collector is still viable, its duty cycle having reached around 75%. UPD packet losses could only be measured in the case of 180 ads/s.

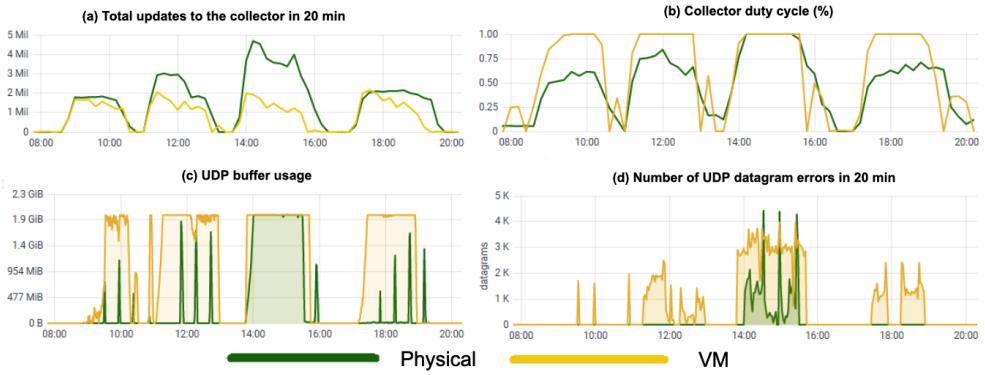

Figure 3. Four rounds of condor_advertise tests on the VM and PM collectors: (a) total number of updates processed over a 20 minutes integration window, (b) core duty cycle, (c) UDP buffer usage and (d) number of UPD packet errors in a 20 minutes integration window.

3.3 Full infrastructure test

In order to explore increasingly larger scales, test pools can be simulated by running multiple multi-core startd daemons for each GlideinWMS pilot job running on the Grid (*uberglideins*, in contrast to one startd per glidein employed in the production pool [19]). This strategy has been successfully employed in previous iterations of our tests [20].

Having analysed the results of the condor_advertise tests, it was decided to explore the scalability limitation of the infrastructure focusing on the setup with the central manager of the test pool being hosted by the physical node, configured identically to the production Global Pool (e.g. running 3 negotiator instances in parallel). An initial set of 10 schedds (later on expanded to 15) were utilized to provide the job submission pressure on the pool. As in previous iterations, millions of jobs were injected in the schedd queues as single-core tasks. The stress on the collector is maximum when the pool is completely fragmented into single-core dynamically partititoned slots, hence the job parameter choice. Other job specifications, such as requested memory, disk, and site whitelist, are randomly generated within realistic values, to maximize job diversity and explore the matchmaking cycle length (negotiation time) on which the pool could efficiently operate. Job length is also selected with a realistic value of 8 hours with a gaussian dispersion of 2 hours.

Several scale test rounds were launched, as Fig. 4 shows. In the first phase, 10 schedds were employed, which was however determined to be insufficient in order to reach the target mark of 500,000 simultaneous running jobs. Indeed, memory usage of the submit nodes increases with the number of *shadow* processes (one per running job to keep track of the its status on the remote execution slots), which in our setup limited the capacity of the schedds to manage running jobs to a maximum of 40k to 45k each. Thus, in the second phase, five more schedds were added to the pool for an increased job submission capacity. With the growth in running jobs and the subsequently enlarged pool, the maximum number of TCP sockets available in the condor connection broker (CCB) service was also reached. This caused that

even if pilots were being submitted to grid sites, their slots could not be registered into the pool. Finally, in the third test round, having corrected all the previous limitations, and with 15 operative schedds, the 500,000 running jobs milestone was achieved, reaching a maximum of 590,000 jobs recorded at peak.

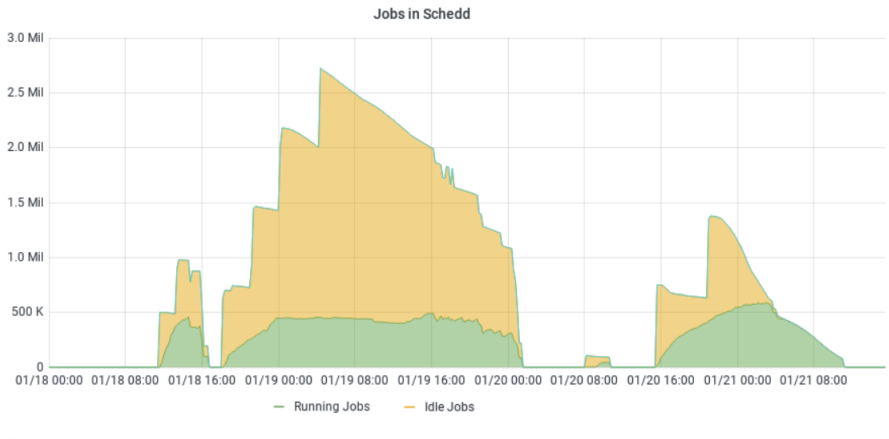

Figure 4. Number of simultaneously running and queued jobs for the scalability tests performed during January 2021 on the CMS HTCondor Global Pool. Successive attempts at reaching the 500,000 running jobs goal were performed under varying conditions, as described in the main text.

3.4 Overall evaluation of the test results

Having determined the clear advantage of the physical node over the VM as host to the collector and negotiator services, and with sufficient capacity to sustain a high enough jobs submission rate, the previously unattainable mark of 500,000 slots and jobs was surpassed.

The plots in Fig. 5 allows us to analyse the role of the collector as a limiting factor in the growth of the pool. We can observe how the collector core daemon duty cycle is correlated with both the total number of dynamic slots in the pool (each corresponding to one running task) and with the number of slot updates that need to be processed. The collector duty cycle saturates around 500,000 slots and with 1 million slot updates (integrated in a 20 minute window).

Beyond that point, even though the collector remains active, its performance is severely degraded, which has a a number of consequences to the performance of the pool. One is immediately visible in Fig. 5 (and also Fig. 6), namely the noticeable monitoring discontinuities and irregular patterns of the metrics displayed. This is caused by the saturated collector failing to reply in part or completely to external monitoring queries used to build these graphs.

Due to the excellent performance of the collector and negotiators during the expansion of the pool, an efficient utilization of the slots can be observed in Fig. 6. However, once the collector starts operating beyond its duty cycle saturation, which represents its point of failure, accumulating missing slot updates resulted in growing missing matchmaking opportunities, producing a relative decrease in slot utilization.

The physical node hosting the central manager presented no stress in absolute terms on the CPU load, which is therefore not considered a constraint to the pool growth, up to the limits presently studied. This node executes multiple essential services in parallel, namely the main collector daemon of the pool, about 100 secondary collectors (which act as intermediary agent between the slots and the main collector), and 3 negotiator processes (as mentioned,

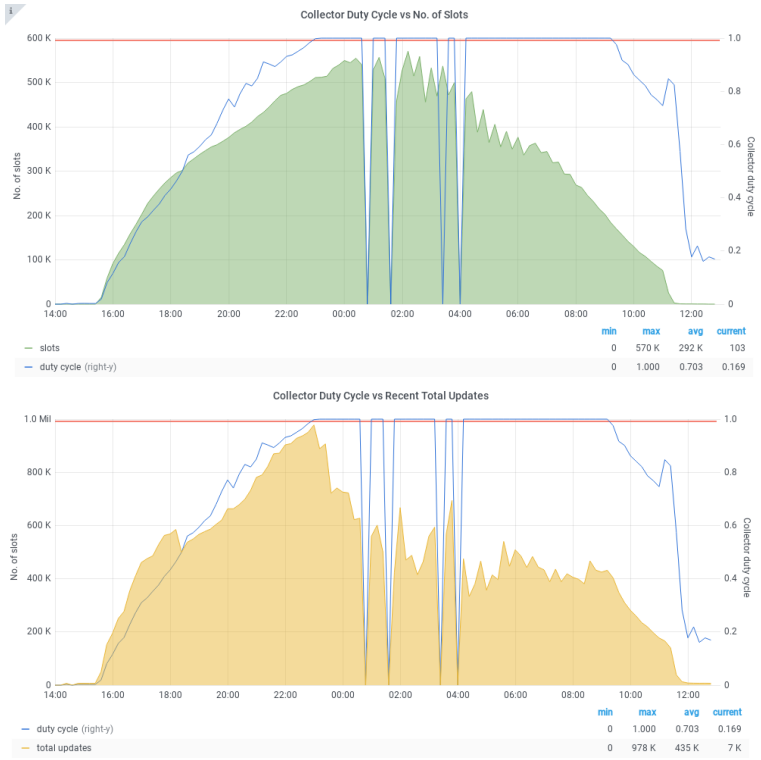

Figure 5. Collector core duty cycle progression until saturation during the third phase of the tests, in comparison to the number of slots in the pool (top), and the total number of slot updates in the preceding 20 minute window (bottom).

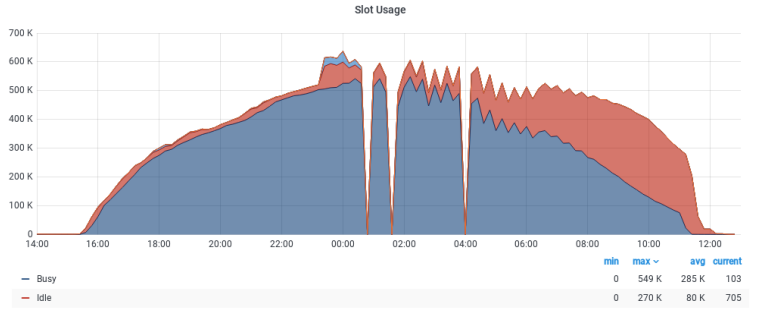

Figure 6. Busy and idle slots during the third phase of the tests, showing excellent resource utilization during the expansion of the pool, reaching the 500,000 slot target scale.

handling matchmaking in parallel). All these processes contribute to memory consumption, which grows with the size of the pool, as Fig. 7 indicates. An interesting observation is the convolution of two effects: an increasing baseline memory usage value, which in this test grows to about 120 GB, and then a very irregular pattern of short bursts of intense memory use above the baseline, each peak representing an increment of an additional 40% to 50% over the baseline value. Memory bursts are caused by forked collector workers that are spawned in order to reply to pool status queries launched by the negotiators (therefore, each peak represents a new matchmaking cycle). Overall memory usage thus nearly reaches saturation

levels for the current host memory capacity, at the explored pool size. Central manager host memory therefore represents an additional immediate limiting factor to pool growth.

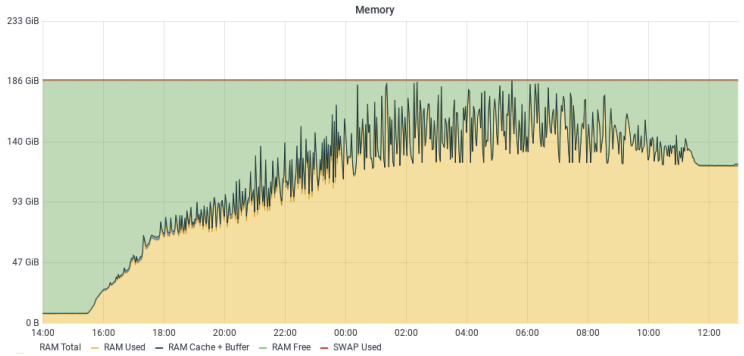

Figure 7. Memory usage of the CM node during the third phase of the tests, described in the main text.

Considering the rest of elements in the infrastructure, it is worth mentioning that the final set of 15 schedds were capable of launching increasingly more jobs in order to induce and sustain pool growth. A combined job start rate of 60 Hz was achieved (thus, 4 Hz per schedd on average). The limiting factor to schedd scalability in these tests was memory, which started to saturate, overloading and even crashing some schedds, when running around 45,000 simultaneous jobs. Negotiator performance was adequate to maintain a high utilization of the slots, with no sign of excessive cycle time observed. On the GlideinWMS side, performance of the FE and factory was also as expected, measuring a FE matchmaking cycle rate of 15 to 20 minutes.

4 Conclusions

In our latest Global Pool scalability tests, performed in January 2021 on HTCondor version 8.9.10, our setup exceeded the significant value of 500,000 concurrently running jobs. The primary bottleneck was (once again) identified to be the collector capacity to process slot updates. The total available memory in the CM of the pool represents a second potential limitation to pool growth and stability, having almost reached saturation in the present tests.

Our examination very clearly confirmed the improved performance of the collector when running on a physical node instead of a VM. An immediate consequence is that SI should be very careful in choosing the appropriate host type for each of its services. For example, schedds and factories can be reasonably deployed in parallel over VMs. On the other hand, it is clear that the best collector performance requires its deployment on a physical node.

Finally, a combined job start rate of 60 Hz was measured when employing a set of 15 schedds. Additional work will be needed in order to ponder the implication of such result when considering the workload submission capacity that will be required to ensure efficient utilization of the combined CMS compute capacity during Run 3 and beyond, towards the HL-LHC phase.

A. Pérez-Calero Yzquierdo acknowledges support by Spain's Ministry of Economy and Competitiveness grant FPA2016-80994.

References

[1] The Glidein-based Workflow Management System, https://glideinwms.fnal.gov/doc.prd/index.html, accessed February, 2021.

[2] HTCondor public web site, https://research.cs.wisc.edu/htcondor/index.html, accessed February, 2021.

[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 Worldwide LHC Computing Grid http://wlcg.web.cern.ch, accessed February, 2021.

[5] D. Spiga et al. “Exploiting private and commercial clouds to generate on-demand CMS computing facilities with DODAS”, EPJ Web of Conferences. **214**. 07027 (2019).

[6] D. P. Anderson. “BOINC: A Platform for Volunteer Computing”, J Grid Computing (2019).

[7] LHC at Home, https://lhcathome.cern.ch/, accessed February, 2021.

[8] D. Da Silva Gomes et al. “Experience with dynamic resource provisioning of the CMS online cluster using a cloud overlay”, EPJ Web of Conferences. **214**. 07017 (2019).

[9] Consorzio Interuniversitario del Nordest Italiano Per il Calcolo Automatico (CINECA) https://www.cineca.it, accessed February, 2021.

[10] M. Schnepf et al. "Dynamic Integration and Management of Opportunistic Re- sources for HEP", EPJ Web of Conferences **214**, 08009 (2019).

[11] A. Pérez-Calero Yzquierdo et al. “CMS Strategy for HPC resource exploitation”, EPJ Web of Conferences **245**, 09012 (2020).

[12] D. Smith et al. “Sharing server nodes for storage and computer”, EPJ Web of Conferences. **214**. 08025 (2019).

[13] D. Giordano et al. “CERN-IT evaluation of Microsoft Azure cloud IaaS, ”, https://zenodo.org/record/48495#.X9I7JC8rx24, accessed February, 2021.

[14] 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).

[15] National Energy Research Scientific Computing Center (NERSC) https://www.nersc.gov, accessed February, 2021.

[16] CMS Offline and Computing Public Results, https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults, accessed February, 2021.

[17] CMS Offline Software and Computing. “Evolution of the CMS Computing Model towards Phase-2”, CMS NOTE-2021/001 (2021), https://cds.cern.ch/record/2751565.

[18] HTCondor Manual Documentation, https://htcondor.readthedocs.io/en/latest/man-pages/condor_advertise.html?highlight=condor_advertise, accessed February, 2021.

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

[20] 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).