Exploitation of network-segregated CPU resources in CMS

C. Acosta-Silva[1,2,*,**], *A.* Delgado Peris[3,5], *J.* Flix[2,3,5], *J.* Frey[4], *J.M.* Hernández[3,5], *A.* Pérez-Calero Yzquierdo[2,3,5], and *T.* Tannenbaum[4]

[1]IFAE, The Barcelona Institute of Science and Technology, 08193 Bellaterra (Barcelona), Spain
[2]PIC, 08193 Bellaterra (Barcelona), Spain
[3]CIEMAT, Scientific Computing Unit, 28040 Madrid, Spain
[4]University of Wisconsin-Madison, USA
[5]On behalf of the CMS Collaboration

Abstract. CMS is tackling the exploitation of CPU resources at HPC centers where compute nodes do not have network connectivity to the Internet. Pilot agents and payload jobs need to interact with external services from the compute nodes: access to the application software (CernVM-FS) and conditions data (Frontier), management of input and output data files (data management services), and job management (HTCondor). Finding an alternative route to these services is challenging. Seamless integration in the CMS production system without causing any operational overhead is a key goal. The case of the Barcelona Supercomputing Center (BSC), in Spain, is particularly challenging, due to its especially restrictive network setup. We describe in this paper the solutions developed within CMS to overcome these restrictions, and integrate this resource in production. Singularity containers with application software releases are built and pre-placed in the HPC facility shared file system, together with conditions data files. HTCondor has been extended to relay communications between running pilot jobs and HTCondor daemons through the HPC shared file system. This operation mode also allows piping input and output data files through the HPC file system. Results, issues encountered during the integration process, and remaining concerns are discussed.

1 Integration of HPC resources into CMS computing

The CMS experiment is aiming towards an increased usage of High Performance Computing (HPC) resources to help cover their growing computing demands while also gaining access to the best available computing technologies, usually employed at HPC sites. In the current international landscape of ever larger scientific projects, growing funds are being committed to HPC centers, and funding agencies are encouraging their LHC national communities to make use of such resources in order to satisfy, at least partially, their computing power demands. This is even more relevant due to their expected increase in the mid to long term future (Run 3 and High-Luminosity LHC) [1], and the projected continuation of current levels of funding.

The current and future contribution from HPC sites is regarded as an integral part of WLCG [2] strategy towards the future High Luminosity LHC. CMS is committed to make

*The authors acknowledge support provided by Spanish funding agency SEIDI, grant PID2019-110942RB-C21, and support provided by NSF under Cooperative Agreement OAC-2030508 as part of the PATh Project.
**The authors thank the BSC and the Spanish Supercomputing Network (RES) for the contributed resources.

the best possible use of non-Grid opportunistic CPUs, including the HPC ones, however is not yet ready to transition effortlessly from traditional computing resources at WLCG sites into HPCs. In particular, network access restrictions, often imposed by HPC centers, are hard to overcome by the current CMS computing model and employed technologies [3].

1.1 Network requirements for HPC resource exploitation by CMS

One of the main hurdles when trying to incorporate HPC resources into the CMS distributed computing system resides in the need to access external services at run time, such as the conditions Database (CondDB) via the Frontier system [4], CMS software via CernVM-FS [5], or input samples using Xrootd [6]. Additionally, the use of pilot jobs for resource allocation in a late-binding model, widely employed by CMS, requires communication between the compute nodes and the central HTCondor [7] payload job scheduler services at CERN.

Often HPC centers do not allow outgoing connectivity from the computing nodes, effectively limiting the potential resource base that CMS can directly use. Multiple R&D efforts are thus active in CMS in order to overcome these limitations [3, 8]. Some of these rely on the existence of edge services or special nodes at the HPC that can bridge network connectivity to the outer world. In this paper we address one of the most restrictive scenarios of HPC centers, where privileged nodes are not available, and present the solutions that have been developed to enable its exploitation by CMS [9].

2 The Barcelona Supercomputing Center

The Barcelona Supercomputing Center (BSC) [10] is the largest HPC center in Spain, and will host one of the most powerful HPC facilities in the EU, expected to reach pre-exascale capabilities in the near future [11]. The biggest general-purpose cluster at BSC is MareNostrum (currently in its fourth phase, MN4), which comprises 3,456 nodes, each with two Intel Xeon Platinum chips fitted with 24 CPU cores, amounting to a total of 165,888 cores and a total memory of 390 TB. The MN4 peak power is 11.15 Petaflops.

In 2020, a collaboration agreement was signed between BSC and the Spanish LHC community, making LHC computing a strategic project, with access to a stable share of BSC CPU. Up to 7% of the MN4 slots, with additional opportunistic use, are expected to be dedicated to LHC activities. The aim is to provide, in the long term, a significant fraction of the CPU budget pledged by the Spanish LHC community, enough to fully cover the Spanish share of the LHC experiments data simulation needs.

The MN4 execute nodes have no external connectivity though, making their use by CMS a difficult task. A shared file disk storage system (GPFS [12]) is mounted on the execute nodes and also on the user login machines, and can be made available to the outside via SSHFS. However, although the MN4 login machines (only entry points) allow incoming connection through SSH, they provide no outgoing network connectivity, and only short-duration processes can be run on them. Therefore, no edge or proxy services can be run on those nodes. Additionally, the BSC does not provide dedicated access points to internally allocated disk storage areas which can be directly integrated in the data management infrastructure used by CMS for its distributed storage. Substantial integration work is therefore required to make BSC capable of running CMS jobs.

3 The HTCondor split-starter model

Considering the challenging scenario for BSC resources exploitation by CMS, in particular the implications derived from the use by CMS of a central service for task distribution relying on a late binding model, a collaboration was formalized between the HTCondor development team and CIEMAT members at the Spanish WLCG Tier-1 center at Port d'Informació Científica (PIC, in Barcelona). A model was discussed, based on the use of the shared filesystem as a communication path for HTCondor daemons, replacing the standard network communication between them. Such development aims not only at solving the BSC case, but to provide a potential solution for the general case of network-restricted resources. The proposed *split-starter* model [13] has been implemented to solve the BSC case, as described in Figure 1.

For the model to work, a bridge node is installed at PIC, running HTCondor startd daemons. This bridge node mounts a pre-determined area of the GPFS shared filesystem at BSC via SSHFS. Additionally, this node also submits jobs to the BSC Slurm workload manager [14], via the BSC login node, in order to acquire resources in the MN4 machine. Slurm jobs act as pilot jobs, instantiating HTCondor starter processes on the BSC nodes. These BSC starter processes employ a rendezvous area in GPFS to connect back to the respective mirror starter processes running at PIC's bridge. Once this happens, startd processes at PIC's bridge, managing the resources allocated at BSC nodes, can join the CMS Global Pool [15] of resources. As the CERN-PIC connection is established, job wrapper scripts and input sand-boxes are passed as tar archive files from the bridge to the BSC node, via SSHFS and GPFS file copy. Likewise, execution status files and produced results are copied back from BSC to PIC.

This is all transparent for the CMS central workload management system, and, from a functional perspective, the resources allocated at BSC can be regarded as standard execution nodes at PIC. In order to enable their late-binding matchmaking to the central task scheduling processes at CERN (schedds), the BSC slots are tagged as an extension of PIC, which also facilitates their monitoring, commissioning and accounting as part of the PIC contribution to CMS.

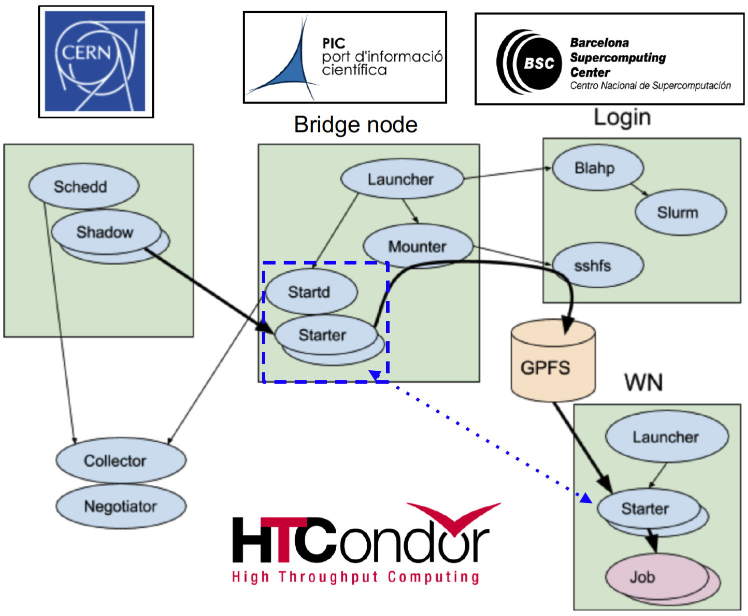

Figure 1. HTCondor split-starter model applied to the BSC case, described in the main text.

3.1 Integration with CMS workload management systems

The split-starter model described was implemented by the HTCondor team, and a prototype was deployed at PIC and BSC. As indicated, startd daemons with a BSC-aware split-starter configuration run at PIC, joining the CMS Global Pool, and making themselves available to receive payload jobs.

The initial prototype employed in the tests described in this report followed approximately the so-called vacuum operation model [16], where startd daemons are launched autonomously by the resource center (PIC, in this case), rather than submitted by the experiment (CMS). Moreover, for the testing phase, the resource slot start-up and clean-up operations were manually performed. A production vacuum model scenario would involve an additional component at PIC handling the automated startd launching and clean-up, based on both the availability of resources at BSC and the existence of suitable workloads in the CMS Global Pool scheduler queues.

The suitability of CMS workloads to be executed at BSC could be determined according to diverse criteria, such as workflows not requiring any input data, tasks belonging to a certain campaign or study, or even containing only simulation jobs (i.e. not involving for example experimental data reprocessing tasks). Technically, such restrictions would be enforced by using a similar approach to that of CMS site-customizable pilots [17]. For the current tests, only payload jobs known to work a priori (described in the next section) were allowed matchmaking the BSC slots.

An alternative mode of operation, and our ultimate goal, is to achieve a tighter integration with standard CMS operations, in which pilots are launched and managed by the GlideinWMS system [15]. This will require that GlideinWMS pilot jobs, submitted centrally to claim BSC resources via PIC, are configured appropriately, so that startds employ the split-starter mode of operation, customized for the BSC case.

4 BSC functional and scale integration tests

In September and October 2020, a series of integration tests were run to validate the described setup. For these tests, realistic, but manually injected CMS simulation jobs were executed. The need to access external services (other than the HTCondor system) was circumvented by pre-placing any data required by the jobs at the BSC shared GPFS. This included a CondDB file, with the appropriate CMS conditions data for these jobs, input data files with generator-level collisions, and a custom singularity [18] container image containing all necessary CMS software libraries. A number of alternative solutions to the problem of accessing these external services are being explored, some already being tested. They will be discussed in Section 6.

Having performed the initial functionality validation of the deployed prototype, including its integration with CMS systems, various scale tests were also performed. From a diversity of configuration options, the schema adopted relied on each submitted Slurm job requesting the resources of a full BSC node, with 48 CPU cores and about 1.8 GB/core of RAM. Moreover, each Slurm job was paired with a single HTCondor startd daemon at PIC, which in turn was configured to advertise the allocated resources (i.e. a full BSC node) as a dynamically partitionable slot. When payload jobs are matched to this resource, the partitionable slot is fragmented into multiple starter processes, each one executing a single payload job and managing the fraction of resources such job requires. Simulation jobs requiring 48, 16 and 8 CPU cores, respectively, were successfully employed. As an example, in the case of jobs requesting 8 cores, 6 starter daemons would be instantiated from the parent partitionable slot, saturating the 48 cores available in a single node. Moreover, in order to facilitate a fast allocation of resources by the BSC scheduler, aiming at a quick and steady ramp up of BSC slots for CMS, the maximum CPU time requested for each Slurm job was kept relatively short (from 4h to 12h).

As shown in Figure 2, the scale achieved by the prototype during these tests slightly exceeded 7,000 CPU cores at peak. This scale was however constrained by the actual amount of resources acquired at BSC during the tests, rather than any inherent limitation observed in the prototype. Indeed, the bridge node at PIC was carefully monitored during the tests, and no sign of saturation was detected in either network, CPU or memory metrics, even at the maximum scale, as shown in Figure 3. In any case, the bridge service would be easily deployed in multiple nodes in parallel, therefore the individual host capacity should never be considered a limitation to higher exploitation scales.

The significance of the maximum scale achieved is evidenced by the fact that the PIC contribution as a Tier-1 site to CMS is currently about 2,000 CPU cores on average. Moreover, it is worth mentioning that the utilization of the obtained slots by CMS payload jobs was excellent. The integrated compute power allocated for this initial testing phase, corresponding to 250,000 hours of CPU time, was in fact exhausted in a couple of days.

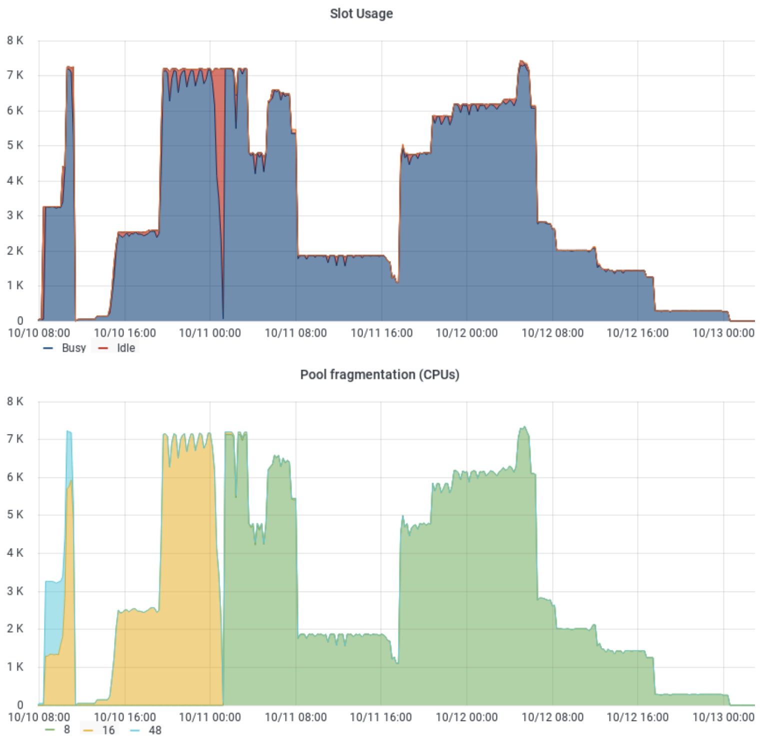

Figure 2. HTCondor split-starter model being tested as prototype for BSC exploitation: total number of CPU cores allocated to CMS during the execution of the tests, in idle or busy states (top), and total number of CPU cores employed by running jobs, successively configured to run as 48, 16 and 8-core tasks (bottom). Excellent resource utilization is observed in the three cases.

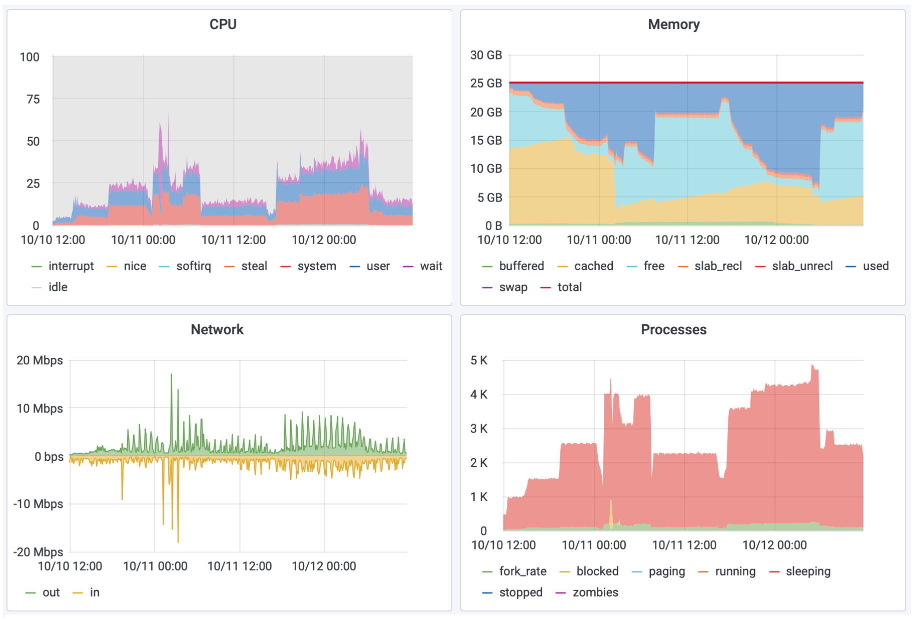

Figure 3. Bridge node host metrics during the execution of the BSC resources integration tests.

5 Handling of input and output data files

The management of input and output event data files, in order to be consumed and produced by CMS jobs at BSC, and exposed to CMS, remains to be implemented. Input files need to be made accessible to jobs running at BSC and output files have to be transferred to a CMS storage element and registered in the CMS data book-keeping and location services, DBS [19] and Rucio [20]. Since BSC jobs cannot directly access any external CMS storage, data will be staged in and out with an intermediate step at PIC storage, using either SSHFS or the BSC's GridFTP service (also subject to a very restrictive access configuration). Initially, matchmaking could be restricted to workflows with no input data, with a simulation task preceded by an event generation step, thus delaying the solution to input data handling to a second production phase.

Two main options are presently being explored for input and output datasets management. One possibility would involve adding pre- and post-execution tasks for each job run at BSC, taking care of in/out data staging and output datatsets registration in the CMS catalogue. This could be regarded as a synchronous operation, which would occur within the lifetime of each individual job. Jobs duration would be longer from the CMS point of view, but no additional CPU would be consumed at BSC during these data management phases. In this case, all data would be registered at PIC's Rucio Storage Element (RSE), and file replicas at BSC would be temporary. The split-starter code already supports configurable hooks that can be used to implement the described data staging strategy. This model, with additional data manipulation steps, would be transparent to the CMS workload management system (WMAgent [21]), which would only be aware of the PIC RSE. In the case of merge tasks, which produce final datasets out of the multiple output files generated from different jobs (unmerged files), they would be exclusively run at PIC, as unmerged data files would always appear as registered in PIC's storage.

An alternative choice would be to define an additional BSC RSE, where files produced and stored at BSC would be registered. In this case, merge jobs would also run at BSC, consuming previously produced unmerged output files stored there. Final merged files would be later (asynchronously) moved to PIC's RSE, having their registration updated in the CMS catalogue. The indicated asynchronous transfers from BSC to PIC would be performed using Rucio, if possible, or a custom transfer service, otherwise.

In order to estimate the achievable data transfer throughput between PIC and BSC, initial tests were also performed. Both BSC and PIC storages (GPFS and dCache [22]) were simultaneously mounted on the aforementioned bridge node at PIC, using SSHFS and parallel NFS, respectively. A cp command could thus be used to cause a double data transfer, from BSC to the bridge node and from there to the dCache disk servers. Transfer rate results are shown in Figure 4. With this simple setup, and running several copy operations in parallel, 1 Gbps throughput was achieved. This boundary is administratively imposed by BSC configuration for a given user and connection to the BSC login node, and can thus be increased by using several connections in parallel. Moreover, BSC also offers an independent GridFTP endpoint, which serves GPFS data. The possibility of running direct GridFTP transfers from BSC to dCache is being explored. In any case, it is worth mentioning that 1 Gbps throughput should be already enough to stage out the data produced by CMS simulation jobs consuming ~7,000 cores at BSC (the maximum instantaneous BSC resource allocation up to now). Indeed, a job roughly produces 1 simulated event per core every minute, whose size is around 1 MB. Thus, 7,000 cores would produce around 930 Mb of data each second, compatible with the observed transfer rate. Nevertheless, the potential interference resulting from a data transfer service running at the bridge node at PIC concurrently with the HTCondor split-starter operations described in the previous section remains to be explored.

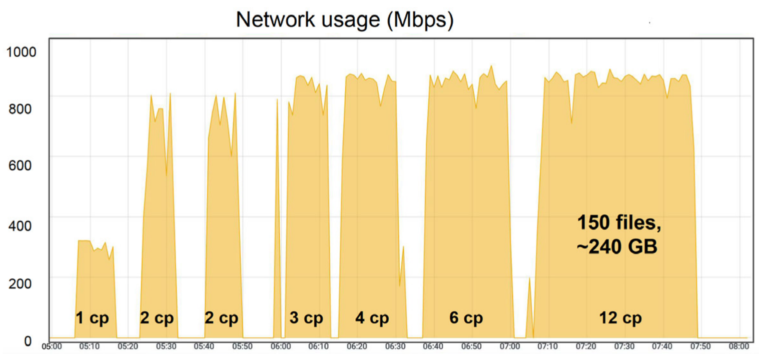

Figure 4. Data transfer test results for the integration of the BSC resources into CMS via PIC described in the main text.

6 Access to other external services

As indicated in the first section of the paper, additionally to the communication between HTCondor daemons handling job execution and the access to input and output datasets, standard CMS jobs also require outgoing network connectivity to access various central services, namely Frontier (for the Conditions Database) and CernVM-FS, for the CMS software (CMSSW). Both services are normally accessed through a system of tiered HTTP caches, built upon the standard web caching tool squid.

Regarding CMS software, two approaches are being considered. Firstly, self-contained singularity images, centrally produced and distributed, including the relevant CMSSW releases required for major simulation campaigns, could be employed. The continuous production of such container images is currently being discussed within the collaboration, as they could be used as a solution for a number of use cases (exploitation of other HPC, or isolated resources). This is in fact the method successfully employed for the tests described in Section 4 of this report. The second option, currently being examined as well, would involve the replication of the whole CMS CernVM-FS repository at the BSC storage. In that case, jobs could simply access the required software at the local GPFS mounted on the execution nodes. For the replication and synchronization of the software repository, the cvmfs_preload tool is being evaluated. Once the repository is copied, only light-weight periodic update operations would be required.

The conditions data, required and specific for every simulation campaign, would also be packed in files for the local consumption by jobs, being replicated at BSC prior their execution. CMS has already recognized the need to centrally and routinely produce such files. For this method to work, a lookup mechanism has been implemented in the CMSSW framework in order to search for locally available conditions data files in the event that access to the external database is not available.

However, work is still required to automate the copy of these conditions data files to BSC storage, or conversely, to enable the selection of the workflows to be run at the BSC resources based on the availability of the corresponding conditions data files. A possible way forward is the distribution of the condition files via CernVM-FS, which, as discussed previously, needs to be reachable by the jobs at the BSC nodes in order to access CMSSW.

7 Conclusion and next steps

A viable operational model for the transparent integration of the available BSC resources into the CMS workflow management system has been designed, implemented and tested. It minimizes central operational efforts for CMS, while maximizing CPU resources utilization. This model overcomes the severe network restrictions imposed by BSC by providing an alternative way to access external services. In terms of job management, it makes use of the HTCondor split-starter feature. Payload execution is enabled by the use of centrally produced singularity images, containing all required CMS software, and conditions data files. The handling of input and output data files can be performed by synchronous stage-in and stage-out additional job steps.

Next stages of this BSC integration work include evaluating the replication of the whole CernVM-FS repository to the BSC shared storage as an alternative for the pre-staging of CMS software and conditions data; the transparent operational integration of BSC slots into the CMS Global Pool infrastructure, making use of its GlideinWMS component, which would submit pilot jobs capable of autonomously employing the split-starter feature; the implementation of customized matchmaking policies, to ensure that only suitable CMS jobs are launched into BSC slots; and the final design, testing and tuning of the input and output data handling routines.

The BSC management council has recently granted 18 million CPU-core hours for CMS in order to help move forward with the subsequent integration phases and start exploiting the BSC infrastructure for production of CMS simulation datasets during 2021. The authors of this paper are thus confident that the productive integration of BSC for CMS use will be achieved shortly.

References

[1] J. Albrecht, A.A. Alves, G. Amadio, G. Andronico, N. Anh-Ky, L. Aphecetche, J. Apostolakis, M. Asai, L. Atzori, M. Babik et al., Computing and software for big science **3**, 1 (2019)

[2] *Worldwide LHC Computing Grid*, https://wlcg-public.web.cern.ch/ (2021), accessed: 2021-02-08

[3] A. Pérez-Calero Yzquierdo, *CMS strategy for HPC resource exploitation*, in *EPJ Web of Conferences* (EDP Sciences, 2020), Vol. 245, p. 09012

[4] B. Blumenfeld, D. Dykstra, L. Lueking, E. Wicklund, *CMS conditions data access using FroNTier*, in *Journal of Physics: Conference Series* (IOP Publishing, 2008), Vol. 119, p. 072007

[5] C. Aguado Sanchez, J. Bloomer, P. Buncic, L. Franco, S. Klemer, P. Mato, XII Advanced Computing and Analysis Techniques in Physics Research p. 52 (2008)

[6] A. Dorigo, P. Elmer, F. Furano, A. Hanushevsky, WSEAS Transactions on Computers **1**, 348 (2005)

[7] D. Thain, T. Tannenbaum, M. Livny, Concurrency - Practice and Experience **17**, 323 (2005)

[8] D. Hufnagel, B. Holzman, D. Mason, P. Mhashilkar, S. Timm, A. Tiradani, F.A. Khan, O. Gutsche, K. Bloom, *HPC resource integration into CMS Computing via HEPCloud*, in *EPJ Web of Conferences* (EDP Sciences, 2019), Vol. 214, p. 03031

[9] T. Boccali, S. Dal Pra, D. Spiga, D. Ciangottini, S. Zani, C. Bozzi, A. De Salvo, A. Valassi, F. Noferini, L. dell'Agnello et al., *Extension of the INFN Tier-1 on a HPC system*, in *EPJ Web of Conferences* (EDP Sciences, 2020), Vol. 245, p. 09009

[10] *Barcelona supercomputing center*, https://www.bsc.es/ (2021), accessed: 2021-02-08

[11] *Digital single market: Europe announces eight sites to host world-class supercomputers*, https://ec.europa.eu/commission/presscorner/detail/en/IP_19_2868 (2021), accessed: 2021-02-18

[12] F.B. Schmuck, R.L. Haskin, *GPFS: A Shared-Disk File System for Large Computing Clusters.*, in *FAST* (2002), Vol. 2

[13] C. Acosta-Silva, A. Delgado Peris, J. Flix Molina, J. Frey, J.M. Hernández, M. Livny, A. Pérez-Calero Yzquierdo, T. Tannenbaum, *Exploiting network restricted compute resources with HTCondor: a CMS experiment experience*, in *EPJ Web of Conferences* (EDP Sciences, 2020), Vol. 245, p. 09007

[14] A.B. Yoo, M.A. Jette, M. Grondona, *Slurm: Simple linux utility for resource management*, in *Workshop on job scheduling strategies for parallel processing* (Springer, 2003), pp. 44–60

[15] J. Balcas, S. Belforte, B. Bockelman, D. Colling, O. Gutsche, D. Hufnagel, F. Khan, K. Larson, J. Letts, M. Mascheroni et al., *Using the glideinWMS system as a common resource provisioning layer in CMS*, in *Journal of Physics: Conference Series* (IOP Publishing, 2015), Vol. 664, p. 062031

[16] A. McNab, F. Stagni, M. Ubeda Garcia, Journal of Physics: Conference Series **513**, 032065 (2014)

[17] A. Pérez-Calero Yzquierdo, M.A. Flechas, D.D. Foyo, S. Haleem, K.H. Anampa, T.T. Ivanov, F.A. Khan, E. Kizinevič, K. Larson, J. Letts et al., *Evolution of the CMS Global Submission Infrastructure for the HL-LHC Era*, in *EPJ Web of Conferences* (EDP Sciences, 2020), Vol. 245, p. 03016

[18] G.M. Kurtzer, V. Sochat, M.W. Bauer, PloS one **12**, e0177459 (2017)

[19] A. Afaq, A. Dolgert, Y. Guo, C. Jones, S. Kosyakov, V. Kuznetsov, L. Lueking, D. Riley, V. Sekhri, *The CMS dataset bookkeeping service*, in *Journal of Physics: Conference Series* (IOP Publishing, 2008), Vol. 119, p. 072001

[20] M. Barisits, T. Beermann, F. Berghaus, B. Bockelman, J. Bogado, D. Cameron, D. Christidis, D. Ciangottini, G. Dimitrov, M. Elsing et al., Computing and Software for Big Science **3**, 1 (2019)

[21] M. Cinquilli, D. Evans, S. Foulkes, D. Hufnagel, M. Mascheroni, M. Norman, Z. Maxa, A. Melo, S. Metson, H. Riahi et al., Journal of Physics: Conference Series **396**, 032113 (2012)

[22] P. Fuhrmann, V. Gülzow, *dCache, storage system for the future*, in *European Conference on Parallel Processing* (Springer, 2006), pp. 1106–1113