EPJ Web of Conferences **245**, 09007 (2020) https://doi.org/10.1051/epjconf/202024509007
CHEP 2019

Exploiting network restricted compute resources with HTCondor: a CMS experiment experience

Carles Acosta-Silva[3], *Antonio* Delgado Peris[2], *José* Flix Molina[1,2],[*], *Jaime* Frey[4], *José M.* Hernández[2], *Miron* Livny[4], *Antonio* Pérez-Calero Yzquierdo[1,2], and *Todd* Tannenbaum[4]

[1]Port d'Informació Científica (PIC), Barcelona, Spain
[2]Centro de Investigaciones Medioambientales y Tecnológicas (CIEMAT), Madrid, Spain
[3]Institut de Física d'Altes Energíes (IFAE), Barcelona, Spain
[4]University of Wisconsin-Madison, Madison WI, USA

Abstract. In view of the increasing computing needs for the HL-LHC era, the LHC experiments are exploring new ways to access, integrate and use non-Grid compute resources. Accessing and making efficient use of Cloud and High Performance Computing (HPC) resources present a diversity of challenges for the CMS experiment. In particular, network limitations at the compute nodes in HPC centers prevent CMS pilot jobs to connect to its central HTCondor pool in order to receive payload jobs to be executed. To cope with this limitation, new features have been developed in both HTCondor and the CMS resource acquisition and workload management infrastructure. In this novel approach, a bridge node is set up outside the HPC center and the communications between HTCondor daemons are relayed through a shared file system. This conforms the basis of the CMS strategy to enable the exploitation of the Barcelona Supercomputing Center (BSC) resources, the main Spanish HPC site. CMS payloads are claimed by HTCondor *condor_startd* daemons running at the nearby PIC Tier-1 center and routed to BSC compute nodes through the bridge. This fully enables the connectivity of CMS HTCondor-based central infrastructure to BSC resources via the PIC HTCondor pool. Other challenges include building custom singularity images with CMS software releases, bringing conditions data to payload jobs, and custom data handling between BSC and PIC. This report describes the initial technical prototype, its deployment and tests, and future steps. A key aspect of the technique described in this contribution is that it could be universally employed in similar network-restrictive HPC environments elsewhere.

1 Introduction

One of the main challenges for the LHC experiments is to integrate High Performance Computing (HPC) resources into their Workload Management (WM) systems, to help cover the expected increase in computing resources needs in the mid to long term future stages of the LHC program (Run3 and HL-LHC), while coping with the projected continuation of current levels of funding. This is part of the WLCG strategy [1], since many of the existing

[*]e-mail: jose.flix.molina@cern.ch

HPC infrastructures are evolving their systems towards the Exascale domain, and the national funding agencies are encouraging communities like LHC to include a fraction of these resources into their computing exploitation plans.

This is an opportunity to integrate HPC resources into a typical Grid-based worldwide infrastructure, which, in addition to being extremely useful to cope with the expected increasing need for resources, would also provide LHC collaborations with the opportunity to access new types of resources. In particular, it might be as well interesting to exploit GPUs, installed in HPC clusters, by adapting part of the LHC experiments applications.

The ATLAS experiment has pioneered the integration of HPC centers into their system, their usage of HPC centers becoming significant in the last years. CMS is as well committed to make the best possible use of non-Grid opportunistic CPUs, including the HPC ones. However, there is a number of elements in the CMS computing model, from data handling to workload management, that need to be adapted to cope with the diversity of HPC environments. Therefore, CMS would not be at this point ready to transition from traditional computing resources at WLCG sites into a scenario dominated by HPCs.

2 Integrating HPC resources: the CMS case

The CMS Computing community has produced a technical document addressing the CMS requirements to successfully exploiting HPC systems[2]. Technically, the deployment of services such as singularity[3], CVMFS[4], data cache services[5], grid job gateways (computing elements, or CEs), storage access for input and output data files, as well as connectivity from the compute nodes to the central CMS WM services would be needed for a complete and easy integration of HPC resources.

Typically, the approach to HPC centers is based on the national and local CMS teams doing the handshaking with HPC representatives from their respective nations or regions. In these discussions, many of these technical constrains need to be negotiated, and some of the functionalities are often not met, which means that ad-hoc solutions need to be explored for each of the cases. For example, in case of CVMFS or run-time access to conditions data not being accessible, CMS would need to prepare suitable images in order to run a containerized version of the CMS applications, bringing the conditions data in a SQLite file, among other constrains that are described in [6] and in the following sections.

Also, it is important to properly negotiate the resource availability for each HPC center exploitation, since the standard procedure goes through allocation grants, a way which is not suitable for LHC experiments, including CMS, as they need steady and reliable resource allocations to plan for the use of the computing resources. The key element to consider when discussing HPC integration is that each center is different, which generates the need to reach agreements on technical and policy questions in each case. This is perceived as a potential cause for big integration efforts by CMS.

3 HPC integration example: BSC case

The Barcelona Supercomputing Center (BSC) is the main HPC site in Spain. The biggest general-purpose cluster at BSC, the MareNostrum 4 (MN4, with over 150.000 processors, with a peak power rated at 11.15 Petaflops), is managed with slurm[7] as batch system, its compute nodes running SUSE Linux Enterprise as operating system.

The BSC site was selected in 2019 for the deployment of a new near-Exascale cluster (MN5, designed for 200 Petaflops by 2021), which becomes a quite sizeable amount of resources, making its integration to CMS computing certainly worth exploring. In particular,

all of the simulation needs for the LHC experiments in Spain represents about ~7% of the MN4 resources. The Spanish CMS community and BSC management have been working into a collaboration agreement, that has been recently approved, turning the LHC computing activities as an strategic project for BSC. This will allow LHC collaborations to exploit BSC resources in a continuous mode along the year.

However, the BSC case is one of the most difficult to integrate into CMS computing infrastructure, since the compute nodes do not have open ports that allow essential connectivity. The login machines only allow ssh incoming connections, and there is no support for edge services at the site (e.g. squids[8], CVMFS, or data transfer services). The CMS software is mostly written and compiled for x86_64 CPU architectures, and for some operating systems, which are not compatible to the one provided by the BSC center. Therefore, dedicated singularity images need the be built for the CMS applications to be executed into suitable containers in the BSC compute nodes. There is no storage element as used in WLCG standards: a shared disk storage area (managed by GPFS[9]) is available and mounted on compute nodes and login machines, and externally available via sshfs. Also, CMS tasks are not optimised for large parallelization (no use of fast node interconnects), and software is not yet fully prepared to use accelerators (GPU, FPGA).

Considering these strict circumstances, the Spanish CMS team at PIC and CIEMAT have joined the HTCondor[10] development team on a dedicated project to enable access to BSC network-isolated nodes for CMS, using the filesystem (GPFS) for the HTCondor control signal path.

3.1 Special HTCondor developments: jumping the network barrier

In order to allocate CPU resources, CMS central Submission Infrastructure (SI) submits GlideinWMS pilot jobs on to the CEs at WLCG computing centers. These pilot jobs interact with the CMS SI to fetch configuration and validation scripts in order to instantiate and run *condor_startd* that join the centrally managed CMS HTCondor Global Pool, acting as execute nodes where CMS applications run. CMS pilots need to connect to the Global Pool collector and negotiator in order to get payloads to execute (late binding model). In this scenario, compute nodes need a small number of ports opened. Moreover, CVMFS is employed to distribute CMSSW versions as well as dedicated singularity images, used by CMS pilots to containerize the execution of jobs. Additionally, payload jobs get experimental conditions from local or remote squid caches (port 80). On top of this, simulation DIGI workflows typically need to read multiple events in order to build the underlying event for LHC collisions (pile-up), which are taken from remote or local files via XRootD. Executing payloads under circumstances where the aforementioned connections can't be established requires non-negligible development and integration work.

The concept of using a shared filesystem as control path for HTCondor, to replace WAN connectivity, was already introduced by HTCondor developers in several conferences [11][12], already having in mind use-cases such as the one discussed here. In this context, the BSC case was indeed used to move forward with all of the required code developments and initial tests.

The main elements of the working model under discussion are described in Figure 1. The core idea is to deploy a bridge node at PIC that allows access to *condor_starter* processes running in the BSC compute nodes, mirroring *condor_starter* processes at PIC. The *condor_startd* process remains at PIC, where it can be accessed to negotiate and keep the late binding model operative. From a functional perspective, the node at BSC can join either the HTCondor pool at PIC or the CMS Global Pool, running CMS tasks directly or via HTCondor flocking.

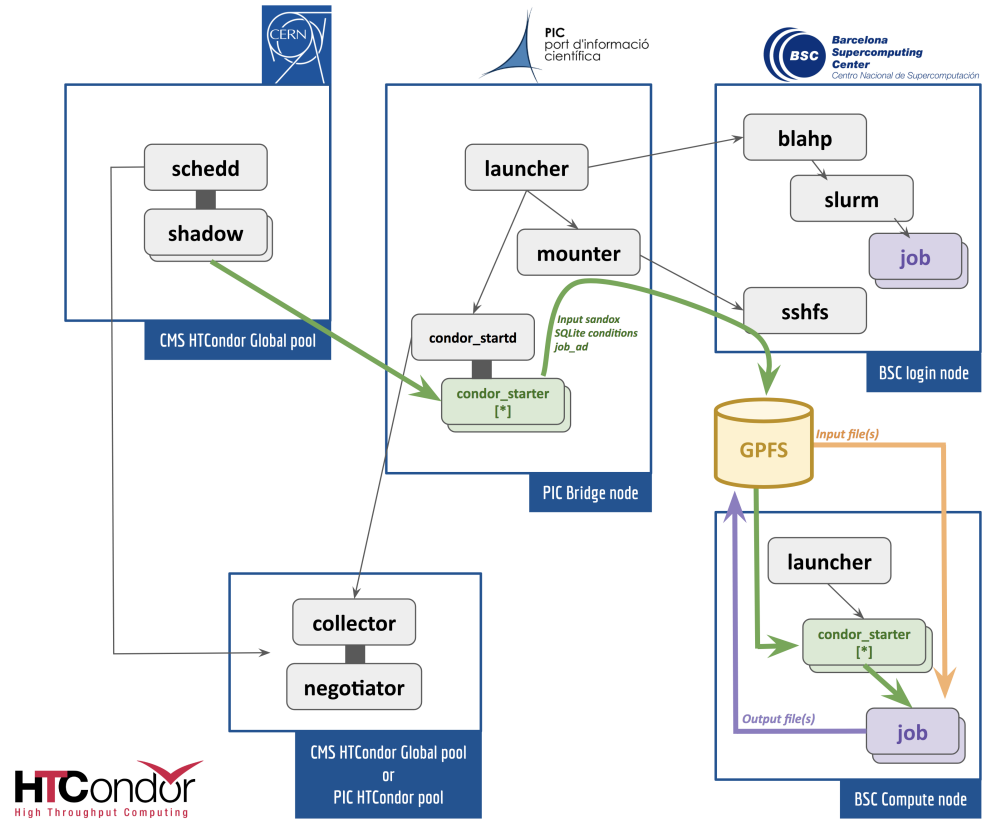

Figure 1. HPC resources at BSC integration model into CMS computing using communication via filesystem to replace WAN.

3.2 HTCondor via shared filesystem at work

A bridge node installed at PIC submits slurm jobs to BSC queues using blahp over ssh (a setup known as Bosco[13]) and handles file transfers over ssh to BSC GPFS (e.g. CMS payloads input sandboxes). For each compute node running the slurm job, a *condor_startd* daemon is spawned on the PIC bridge server. These *condor_startds* are configured to act as if they hold the hardware resources (cpus, memory, disk) of a single BSC compute node. Each *condor_startd* sets up an sshfs mount on the PIC server of BSC's GPFS file system (via BSC login node), creating a series of rendezvous directories in the filesystem, one per BSC node. The *condor_startds* join the HTCondor pool collector but advertise themselves as being unavailable for matchmaking. Once the slurm jobs start running, *condor_startds* switch to re-advertise as ready to be matched and accept payload jobs. The *condor_starters* are instantiated on the BSC compute nodes, and these processes update their status to the mirror *condor_starter* processes running at PIC, at the other side of the network barrier. In order to enable tasks execution, each *condor_starter* copies the job_ad and input sandbox of the payload job to the corresponding rendezvous directory of its mirror *condor_starter* in GPFS. Input and output sandboxes and status signals are regularly synchronized between the bridge to the compute node at BSC, passing .tar files via GPFS.

The *condor_startd* process at PIC opens to matchmaking by CMS workload scheduling submit nodes (schedds) at CERN, being allocated resources at from the startd, subsequently passing payload jobs to the *condor_starter*, and via filesystem synchronization, to the BSC compute node. PIC *condor_starters* wait for the output sandbox and a job_ad summarizing the execution to appear in the rendezvous directory. These files are copied back out of filesystem and normal post-job-execution steps can then proceed (transfer the output sandbox to the submit machine, etc). From a functional perspective, in this model, the compute nodes at BSC can be regarded as having joined an external HTCondor pool, for example the HTCondor pool at PIC, which can be federated to CMS Global Pool to be used for CMS job execution.

3.3 Current status and next steps

The basic setup which includes the main HTCondor elements has been developed and deployed as a prototype, and it is up and running, functionally interconnecting PIC and BSC as described. The HTCondor bridge has been built and tested, although jobs are still manually submitted to BSC slurm. The setup has been integrated into a test bed CMS HTCondor pool and test jobs (no real CMS payloads yet) have been sent to validate the new HTCondor functionalities.

Current activities are centered in integrating the CMS pilot jobs infrastructure into this new schema. The CMS GlideinWMS frontend and factory nodes provide CMS with advantageous functionalities that should be incorporated in this project, including the capability to trigger the resource allocation mechanism when suitable workload pressure is detected in the schedds queues (a first matchmaking stage), as well as holding the information required to configure and validate an arbitrary HTCondor startd into one suitable for CMS workload execution.

Additionally, efforts are being dedicated at developing mechanisms for the automatized creation of CMS fat singularity images, including all relevant CMSSW versions in CentOS7, and their pre-placement in the BSC GPFS. Moreover, conditions data tags need to be extracted from payloads as soon as their configuration files land at the PIC bridge. From each tag, conditions data need to be gathered from the squid servers at PIC in order to build SQLite files that can be transferred as part of the job input sandbox. Then, payload jobs configuration needs to be modified to set the jobs to access these conditions data files from their storage at the BSC GPFS area.

Concerning job input data, in the case of certain tasks, the pile-up simulation sample, with a volume of about 500 TB, needs to be pre-placed in the BSC GPFS area. The payload configuration needs to be further modified so jobs can read local files via POSIX. Also, as we are interested in running a particular type of jobs in the BSC nodes, additional filters need to be introduced in the *condor_startd* configuration to restrict job matchmaking to those suitable tasks. Finally, we will need to automate the submission of slurm jobs from the PIC HTCondor bridge, as well as automating the post-processing steps, including the payload job output data transfer from BSC to the HTCondor bridge and into the PIC storage element.

Once the system is fully integrated, scale tests will be required, given that the scaling limitations of the mode presented here are yet unknown. The setup will probably need to be evolved in order to perform at the desired scale (approximately 4.000 CPU cores in continuous usage).

4 Conclusions

CMS is dedicating increasing efforts to the progressive integration of HPC resources, as they are expected to play an important role in the future LHC program. CMS strategy relies on local CMS teams establishing agreements with their national HPC centers, a work that started in Spain already in 2018. The process has culminated with a collaboration agreement signed up that will allow the Spanish LHC community to stably exploit a fraction of the BSC resources. However, BSC infrastructure and operations policies present some particularities and restrictions that are not suitable for an effortless integration of CMS workflows.

A special project started in 2018, with the help of the HTCondor development team, aiming at overcoming one of the major difficulties: the isolation of BSC compute nodes, which potentially breaks the late binding model of CMS. The model being developed and implemented seems promising, although a significant number of elements need to be modified to enable a successful integration. Once the proposed model is deployed and functionally operative, scale tests will demonstrate whether the proposed solution can handle the expected load, or if we need to further tune or expand some of the elements deployed.

As explained, overcoming the network barrier with the model presented in this report might be a solution that could be applied to other restrictive HPC scenarios worldwide, hence the benefits for the CMS collaboration that can be obtained from the successful completion of this work would be multiple.

This work was partially supported by MICINN in Spain under grants FPA2016-80994-C2-1/2-R, which include FEDER funds from the European Union. We would like to thanks our partners in the GlideinWMS and HTCondor development teams, and our colleagues at BSC and CERN, who are supporting this initiative.

References

[1] J. Albrecht, et al, *A Roadmap for HEP Software and Computing RD for the 2020s*, Computing and Software for Big Science volume 3, Article number: 7, 2019

[2] CMS Offline, Software and Computing, *HPC resources integration at CMS*, CMS-NOTE-2020-002 ; CERN-CMS-NOTE-2020-002

[3] Singularity webpage: https://singularity.lbl.gov/

[4] J. Blomer et al, *Distributing LHC application software and conditions databases using the CernVM file system*, J. Phys.: Conf. Ser. 331 042003, 2011

[5] X. Espinal, et al, *The Quest to solve the HL-LHC data access puzzle. The first year of the DOMA ACCESS Working Group*, to be published in these proceedings.

[6] A. Pérez-Calero Yzquierdo, *CMS strategy for HPC resource exploitation*, to be published in these proceedings.

[7] Slurm worload manager: https://slurm.schedmd.com/

[8] Frontier Squid as a distribution of the well-known squid HTTP caching proxy software that is optimized for use with applications on the Worldwide LHC Computing Grid (WLCG): http://frontier.cern.ch/

[9] IBM GPFS (General Parallel File System): https://researcher.watson.ibm.com/researcher/view_group.php?id=4840

[10] HTCondor webpage: https://research.cs.wisc.edu/htcondor/

[11] T. Tannenbaum, et al, introduced the concept: What's new in HTCondor, HTCondor Week 2018: https://agenda.hep.wisc.edu/event/1201/session/8/

[12] T. Tannenbaum, et al, development ideas: *What's new in HTCondor*, EU HTCondor Workshop 2019: https://indico.cern.ch/event/817927/contributions/3570445/8

[13] Bosco: https://osg-bosco.github.io/docs/BoscoInstall/