Integrating LHCb Offline Workflows on Supercomputers

State of Practice

Alexandre F. Boyer[1,*], *Federico* Stagni[1], *Christophe* Haen[1], *Christopher* Burr[1], *Vladimir* Romanovskiy[2], and *Concezio* Bozzi[3]

[1]CERN, EP Department, Geneva, Switzerland
[2]NRC Kurchatov Institute, IHEP, Protvino, Russia
[3]INFN Sezione di Ferrara, Ferrara, Italy

Abstract. To better understand experimental conditions and performances of its experiment, the LHCb collaboration executes tens of thousands of loosely-coupled and CPU-intensive Monte Carlo simulation workflows per hour. To meet the increasing LHC computing needs, funding agencies encourage the collaboration to exploit High-Performance Computing resources, and more specifically supercomputers, which offer a significant additional amount of computing resources but also come with higher integration challenges. This state-of-practice paper outlines years of integration of LHCb simulation workflows on several supercomputers. The main contributions of this paper are: (i) an extensive description of the gap to address to run High-Energy Physics Monte Carlo simulation workflows on supercomputers; (ii) various methods and proposals to maximize the use of allocated CPU resources; (iii) a comprehensive analysis of LHCb production workflows running on diverse supercomputers.

1 Introduction

The High Luminosity Large Hadron Collider (HL-LHC) is an upgraded version of the LHC designed to achieve instantaneous luminosities approximately ten times greater than the original values of the LHC, improving the potential for discoveries after 2029 [1]. The upgrade will enable the LHCb experiment to enlarge its data sample by an order of magnitude and the physicists to observe rare processes. Running Monte Carlo simulations is essential for preparing such an upgrade within the LHCb collaboration. These simulations aim to understand the experimental conditions and performance of the experiment, as well as support the analysis of past, present, and future datasets.

The simulation of proton-proton collisions, as well as the hadronization and decay of the resulting particles, are controlled by Gauss [2, 3], a CPU-intensive application. In 2022, MC simulations consumed more than 90% of the CPU hours available to the collaboration, and a larger number of complex simulations are expected with the HL-LHC.

The LHCb collaboration focuses on two strategies to cope with the lack of pledged computing resources: (i) reducing the memory and CPU footprint of Gauss and adapting it to further computing environments; (ii) exploiting opportunistic computing resources, and especially supercomputers, by improving the LHCb Workload Management System (WMS).

*e-mail: alexandre.boyer@cern.ch

In this paper, we are going to describe the latter approach and mainly the integration of a MC simulation application on supercomputers, which offer a significant amount of computing resources, but present higher integration challenges.

The LHCb collaboration has exploited supercomputers for a few years and has relied on wealth of literature about similar experiments. To the best of our knowledge, the current literature focuses on specific cases and does not provide abstract models that would help communities conducting analogous operations in different contexts, implying other WMS, workloads, and/or constrained supercomputers. The objective of this paper is to provide a generic plan relying on different concrete experiments to guide similar organizations in this process.

We first describe LHCb requirements and known constraints related to supercomputers (Section 2). We then introduce the outcome of our literature review on the integration of some LHC workload in supercomputers (Section 3). We also describe solutions that we implemented within the LHCb WMS to extend its capabilities (Section 3). Finally, we present several use cases that consists in applying the solutions we developed to different supercomputers (Section 4).

2 Analysis of Constraints

2.1 Software Architecture

The software architecture challenge refers to the ability of an application to efficiently use the computing power of a given environment. Nowadays supercomputers typically include multi-core and many-core x86 CPUs, although non-x86 and RISC CPUs are becoming more common. Most integrate accelerators such as GPUs. Besides, supercomputers are primarily designed for parallel processing with inter-process communication and fast inter-node connectivity. They tend to favor a small number of multi-node allocations over many single-core allocations. Many-core nodes generally cannot support one instance of Gauss per core due to the memory footprint of the application.

Gauss, on the other hand, is currently a CPU-intensive single-process (SP), single-threaded (ST) application, with an average memory footprint of 1.4 GB. It takes a simple configuration file as input and typically produces outputs of a few MB-GB. It supports only CISC x86 architectures and CERN CentOS compatible environments.

Given the above definition, the application would not be adapted for supercomputers, but there are many ongoing efforts to optimize and extend the application. The developers are actively working on a multi-threaded version of Gauss called Gaussino [4] to reduce the memory footprint of the application and improve its efficiency. In parallel, they are also attempting to prepare Gauss for RISC architectures such as ARM.

2.2 Distributed Computing

The distributed computing challenge refers to the ability of the WMS to provide allocations and compatible environments including all the dependencies - data and software - for the application to run successfully. This can be a complex task as supercomputers can be very different. Supercomputers may embed different GNU/Linux distributions and impose their own security models on users. For instance, administrators may prohibit outbound connections from Worker Nodes (WNs) and/or require users to connect to a Virtual Private Network (VPN) to access computing resources. Similarly, supercomputer administrators generally prevent users from mounting file systems such as CVMFS [5] on the WNs, a distributed file system that allows LHC experiments to access their software from any distributed grid

site. The uniqueness of supercomputing resources demands specific solutions for each one, generally expensive in terms of development and operational efforts.

In the context of LHCb, the WMS refers to the DIRAC Workload Management System [6]. DIRAC can interact with different types of Computing Elements (CEs) and Local Resource Management System - via ssh - to supply grid resources with Pilot-Jobs [7]. The developers chose to rely on the Pilot-Job paradigm, which is much more efficient than pushing jobs to distributed computing resources, but requires ubiquitous external connectivity. DIRAC was initially designed to request single-core allocations but has recently been enhanced to support multi-core allocations too. Pilot-Jobs provide a controlled and compatible environment by encapsulating tasks in Singularity/Apptainer [8] containers whenever possible. Pilot-Jobs can also rely on CVMFS to provide tasks with their sotware dependencies.

The current distributed computing strategy of LHCb consists in (i) exploiting x86 CPU supercomputer partitions; (ii) collaborating with the local system administrators and performance experts, which has proven to be mutually beneficial and has helped to address many specific issues; (iii) adapting the DIRAC WMS to constrained environments. The latter point is essential given the substantial amount of computing power that supercomputers can provide and the growing computing needs to process future LHC data.

3 Software blocks

We analyzed several use cases involving teams integrating their HTC workloads on supercomputers. Most of the solutions required immediate development without an analytical understanding of the underlying abstractions. We developed an abstract model in the form of an activity diagram, shown in Figure 1 and described below. It remains relatively tied to the LHC experiments and does not address data movement problems but could serve as a basis for further projects.

The model is designed to question the reader about the constraints of a given supercomputer. The following pathways allow communities to test the capabilities of the system and build the necessary components to overcome the obstacles that might hinder the integration of their workloads. Overall, the solutions on the left side of Figure 1 are more generic, but less efficient, than those on the right side.

3.1 Outbound connectivity

The connectivity policy implies a choice of provisioning model. Getting external connectivity from the WNs allows them to contact external services, thus employing the Pilot-Job paradigm.

Having partial connectivity - WNs only have access to the edge nodes that can contact external services - is similar, but requires a gateway service. The role of a gateway service is to redirect network traffic from Pilot-Jobs and tasks running on WNs to external services. A gateway service consists of a client installed within the allocation on a WN, and a service hosted on an edge node [9]. The service works provided that system administrators allow WMS administrators to install their applications on an edge node, which is not always the case. Otherwise, jobs have to be pushed to the supercomputer.

In this case, the WMS needs to embed or communicate with a Job Manager, which is installed outside of the machine and works as a Pilot-Job. A Job Manager pre- and post-processes jobs - including interactions with external services -, and solely submits the tasks to a remote middleware. It also monitors jobs and controls the throughput. The best known example is the ARC Control Tower (aCT), which aims to provide a generic interface to the WMS [10].

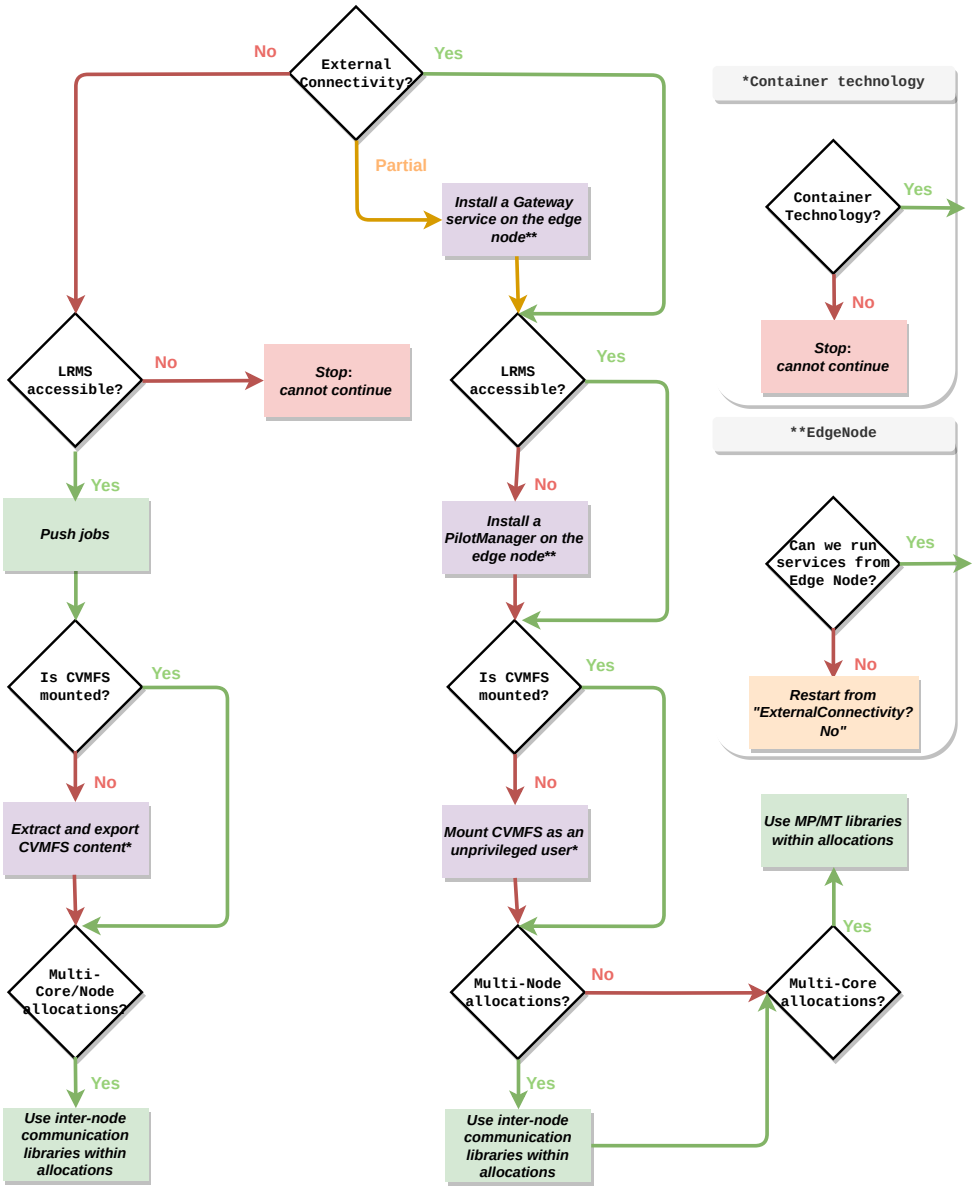

Figure 1. Activity diagram to better understand capabilities of a constrained cluster and existing solutions to cope with integration obstacles.

To address external connectivity issues while preserving the current architecture, we introduced two components within DIRAC: *PushJobAgent*, which fetches jobs from a DIRAC server, and *RemoteRunner*, which submits and monitors the workload on the supercomputer. The first response of the activity diagram leads to significant changes in the approach to adopt and leads to two distinct paths in the diagram.

3.2 LRMS access

It is generally more convenient to interact with the LRMS from an edge node rather than from outside a supercomputer, especially if it is protected by a VPN. WMS administrators can install a Pilot Manager directly on an edge node to locally submit Pilot-Jobs to the LRMS. The service works as long as the system administrators permit WMS administrators to install their applications on an edge node, although this permission is not always granted. Moreover, if there is no outbound connectivity within the system, then the solution is simply not viable.

The DIRAC Pilot Manager, called *Site Director*, could run on an edge node if allowed, and provided a host certificate to interact with external DIRAC services.

3.3 Software dependencies through CVMFS

Boyer et al. provide an overview of the existing approaches to using CVMFS from unprivileged/unconnected environments [11]. The best known one is *cvmfsexec* [12], which allows the file system to be mounted as an unprivileged user. The program requires a specific environment to work correctly: (i) external connectivity; (ii) the *fusermount* library or unprivileged namespace mount points or a setuid installation of Singularity/Apptainer. Blomer et al. provide further details on the package [13].

Communities using supercomputers that do not provide outbound connectivity cannot directly benefit from *cvmfsexec*: the package still needs to fetch updated data via HTTP, which is not available in this context. Teuber and the CVMFS developers conceived *cvmfs_shrinkwrap* in response [14]. *cvmfs_shrinkwrap* can extract specific CVMFS files and directories based on specification files, deduplicate them, and make them easily exportable in various formats such as squashfs or tarball. Boyer et al. developed an open source and generic command-line interface, called *subcvmfs-builder*, around *cvmfs_shrinkwrap* to assist WMS administrators in this process [11]. It allows the dependencies of a given application to be identified, extracted, tested and deployed on constrained computing resources.

To deliver Gauss dependencies on supercomputers without external connectivity, we developed *subcvmfs-builder-pipeline*, a GitLab CI that relies on *subcvmfs-builder* to produce a subset of CVMFS. The CI pipeline runs every day and typically takes about 1 hour and 30 minutes. The resulting subset of CVMFS represents 0.24% of the LHCb-CVMFS repository and occupies around 6 GB: 3.2 GB of space for the dependencies and 2.8 GB for the *cvmfs_shrinkwrap* metadata. This is sufficient to run most of the current Gauss workloads.

3.4 Allocations

Supercomputers embed many-core nodes and typically propose a limited number of allocations. This approach encourages users to exploit multiple cores and even multiple nodes within the same allocation. To take advantage of multiple nodes in parallel, teams use inter-node communication libraries such as MPI and OpenMP to develop wrappers around pilots and/or jobs [15]. In the case where WNs do not have access to the external network, wrappers embed a fixed number of jobs, which means that they should be of similar duration to avoid wasting resources. Otherwise, wrappers can embed Pilot-Jobs and better control the allocated space.

DIRAC currently relies on a specific solution based on `srun`, a Slurm command that allows running MPI pilots involving many nodes. DIRAC requests a variable number of nodes - between *min nodes* and *max nodes* and submits a single pilot that is copied and executed in parallel by `srun` on each allocated node. They all run under the same pilot identifier and status.

To leverage a maximum number of cores in a node, we also designed a flexible and generic fat-node partitioning mechanism to simultaneously run independent jobs from the same Pilot-Job [16]. Once installed on the node, the Pilot-Job checks the number of processors available and sequentially fetches jobs accordingly to execute them in parallel. The fat-node partitioning mechanism supports complex descriptions of jobs and resources. It allows users to submit single-core and multi-core jobs.

4 Use Cases

4.1 Piz Daint

Piz Daint, ranked 28^{th} in the June 2023 Top500 list [17], has been installed at the Swiss National Supercomputing Center (CSCS) in 2012 and updated three times since then. It consists of 387,872 nodes with Intel Xeon E5-2690v3 12-cores at 2.6GHz chips.

We are able to closely collaborate with the administrators of the supercomputers, who provide us with pledged computing resources and allow us to address the issues that are commonly found on such infrastructures. We can request single-core allocations by submitting pilots via an ARC CE. CVMFS is mounted on the WNs that have external connectivity.

4.2 Marconi A-2 and Marconi 100

Marconi Intel Xeon Phi [18] is a supercomputer at CINECA, available to the Italian and European research community. Ranked number 22^{nd} in the June 2020 Top500 [17], Marconi Intel Xeon Phi provides its compute capacity through several independent partitions. The Marconi-A2 partition consisted of nodes equipped with one Xeon Phi 7250 (KNL) at 1.4 GHz, with 96 GB of RAM. System administrators installed CVMFS and allowed jobs to interact with certain external services from CERN. Approximately 300MB of RAM was available per core when all 272 logical cores were in use, much less than the 1.4 GB required by Gauss.

We submitted pilots via an HTCondor CE and experimented with the fat-node partitioning mechanism to request one fat node per allocation. Each pilot was able to fetch and run one job per physical core, resulting in 68 jobs per allocation [16].

In January 2020, the A2 partition was turned off to increase the compute capacity of the center. KNLs were replaced with nodes equipped with IBM Power9 CPUs and NVIDIA Volta V100 GPUs in a new Marconi100 supercomputer, ranked 26^{th} in June 2023. We have the possibility to take advantage of the GPUs, but we do not currently have the workload to run on such WNs.

4.3 Santos Dumont

The Laboratório Nacional de Computação Científica (LNCC), in Brazil, provides opportunistic allocations to the LHCb collaboration. The Santos Dumont (SDumont) supercomputer, ranked 424^{th} of the Top500 in June 2022, was the first Petascale machine in Brazil, with a total of 36,472 CPU cores, distributed across 1,134 compute nodes, mainly composed of CPUs ([19]). Local system administrators manage computational resources and their use with Slurm. They enable outbound connections from the WNs.

Thanks to a close collaboration with LNCC, system administrators installed and mounted CVMFS on the nodes, and created an SSH bridge from an LHCbDIRAC host to an edge node of the supercomputer so that LHCb administrators and developers do not need to deal with the presence of a VPN. We have leveraged Pilot-Jobs, the fat-node partitioning and the

`srun` tool to exploit the available resources. The number of jobs processed per hour has varies considerably over time but we request and use multiple nodes per allocation and any available logical cores within the nodes.

4.4 Mare Nostrum 4

Managed by the Barcelona Supercomputing Center (BSC), Mare Nostrum is the most powerful and emblematic supercomputer in Spain [20]. Mare Nostrum was ranked 98^{th} in the June 2023 Top500 list [17]. The general purpose partition of Mare Nostrum comprises 153,216 cores distributed across 3,456 computational nodes. Each node in the general-purpose block is equipped with two Intel Xeon Platinum 8160 24-cores at 2.1 GHz chips, and at least 2 GB of RAM per core: this configuration meets the requirements of Gauss. Nodes and users share access to a shared file system, where they have access to 5 TB of storage. The WNs also host a non-CERN-compliant version of Linux: SUSE Linux Entreprise Server 12. Administrators manage the nodes using the Slurm LRMS.

Mare Nostrum is very restrictive. By default, CVMFS is not mounted on the nodes and administrators would not allow it. Worker nodes and edge nodes have no external connectivity and no service can be installed on the edge nodes. Therefore, it is impossible to deploy Gauss tasks to WNs using Pilot-Jobs.

To provide tasks with Gauss dependencies and a reproducible environment, we first created a CernVM Singularity/Apptainer container and moved it to the shared file system of Mare Nostrum. Then, we set up the LHCb *subcvmfs-builder-pipeline* to push the latest Gauss dependencies to the shared file system of the supercomputer. To facilitate the management of the workloads on Mare Nostrum, a team from the Port d'Informació Científica (PIC) installed an ARC instance on their infrastructure. We installed and configured a *PushJobAgent* to push Gauss tasks to Mare Nostrum via the ARC instance hosted by PIC. We configured the *PushJobAgent* instance to initially handle 200 single-core jobs in parallel. The current approach is a simple solution that minimizes code changes. To handle more jobs, we would need to redesign the structure of the DIRAC WMS workflows embedding Gauss.

5 Conclusion

This work promotes and relies on the research efforts of the LHC experiments which continue to develop tools to exploit supercomputers. The sections are relatively bound to LHCb and MC simulations but they could be transferred to any other similar projects depending on distributed, shared and, especially, constrained computing resources .

We specified our requirements and abilities for dealing with constrained environments (Section 2). Next, we presented a generic plan and some implementations to help communities integrate their applications on supercomputers (Section 3). We emphasized solutions to support resources without external connectivity: (i) *PushJobAgent*, a DIRAC agent to perform pre- and post- operations from LHCb servers and only submit the tasks to supercomputers (Section 3); (ii) *subcvmfs-builder-pipeline*, a generic CI pipeline to automatically build and deploy subset of CVMFS (Section 3). We also described abilities of DIRAC to manage many-core allocations in supercomputers with external connectivity (Section 3). Finally, we presented some supercomputers and followed the logic of the plan to integrate Gauss tasks on their computing resources (Section 4). It is worth noting that the examples we have presented are very different from each other in their architectures and policies.

While the used amount of computing power is still significantly lower with respect to the current pledged resources that we can exploit, it lays the foundations for further and larger upcoming allocations of computing resources.

References

[1] G. Apollinari, I. Béjar Alonso, O. Brüning, M. Lamont, L. Rossi, *High-Luminosity Large Hadron Collider (HL-LHC) : Preliminary Design Report*, CERN Yellow Reports: Monographs (CERN, Geneva, 2015), http://cds.cern.ch/record/2116337

[2] M. Clemencic, G. Corti, S. Easo, C.R. Jones, S. Miglioranzi, M. Pappagallo, P.R. and, Journal of Physics: Conference Series **331**, 032023 (2011)

[3] I. Belyaev, T. Brambach, N.H. Brook, N. Gauvin, G. Corti, K. Harrison, P.F. Harrison, J. He, C.R. Jones, M. Lieng et al., Journal of Physics: Conference Series **331**, 032047 (2011)

[4] B.G. Siddi, D. Müller, *Gaussino - a Gaudi-Based Core Simulation Framework*, in *2019 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)* (IEEE, Manchester, United Kingdom, 2019), p. 1–4, ISSN 2577-0829

[5] P. Buncic, C.A. Sanchez, J. Blomer, L. Franco, A. Harutyunian, P. Mato, Y. Yao, Journal of Physics: Conference Series **219**, 042003 (2010)

[6] A. Tsaregorodtsev, F. Stagni, P. Charpentier, A. Sailer, M.U. Garcia, K.D. Ciba, C. Haen, C. Burr, A. Lytovchenko, R. Graciani et al., *Diracgrid/dirac: v8.0.27* (2023), https://doi.org/10.5281/zenodo.8249063

[7] M. Turilli, M. Santcroos, S. Jha, ACM Comput. Surv. **51**, 43:1–43:32 (2018)

[8] L. Foundation, *Apptainer, the container system for secure high-performance computing* (2022), https://apptainer.org/

[9] Tovar, Benjamin, Bockelman, Brian, Hildreth, Michael, Lannon, Kevin, Thain, Douglas, EPJ Web Conf. **251**, 02032 (2021)

[10] J. Nilsen, D. Cameron, A. Filipčič, Journal of Physics: Conference Series **664**, 062042 (2015)

[11] A.F. Boyer, C. Haen, F. Stagni, D.R.C. Hill, *A Subset of the CERN Virtual Machine File System: Fast Delivering of Complex Software Stacks for Supercomputing Resources*, in *High Performance Computing*, edited by A.L. Varbanescu, A. Bhatele, P. Luszczek, B. Marc (Springer International Publishing, Cham, 2022), pp. 354–371

[12] CVMFS, *cvmfsexec* (2022), https://github.com/cvmfs/cvmfsexec

[13] Blomer, Jakob, Dykstra, Dave, Ganis, Gerardo, Mosciatti, Simone, Priessnitz, Jan, EPJ Web Conf. **245**, 07012 (2020)

[14] S. Teuber, *Efficient unpacking of required software from CERNVM-FS* (2019), https://doi.org/10.5281/zenodo.2574462

[15] K. De, A. Klimentov, D. Oleynik, S. Panitkin, A. Petrosyan, J. Schovancova, A. Vaniachine, T. Wenaus, A. Collaboration et al., *Integration of PanDA workload management system with Titan supercomputer at OLCF*, in *Journal of Physics: Conference Series* (IOP Publishing, 2015), Vol. 664, p. 092020

[16] F. Stagni, A. Valassi, V. Romanovskiy, EPJ Web of Conferences **245**, 09002 (2020)

[17] Top500, *Top500 the list* (2022), https://www.top500.org/

[18] Cineca, *Cineca hpc* (2023), https://www.hpc.cineca.it/hardware/marconi

[19] LNCC, *Sdumont* (2021), https://sdumont.lncc.br/

[20] D. Vicente, J. Bartolome, *BSC-CNS Research and Supercomputing Resources*, in *High Performance Computing on Vector Systems 2009*, edited by M. Resch, S. Roller, K. Benkert, M. Galle, W. Bez, H. Kobayashi (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010), pp. 23–30, ISBN 978-3-642-03913-3