Future Generation Computer Systems 133 (2022) 23–38

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

DIRAC Site Director: Improving Pilot-Job provisioning on grid resources

Alexandre F. Boyer [a,b,*], Christophe Haen [a], Federico Stagni [a], David R.C. Hill [b]

[a] European Organization for Nuclear Research, Meyrin, Switzerland
[b] Université Clermont Auvergne, Clermont Auvergne INP, CNRS, Mines Saint-Etienne, LIMOS, 63000 Clermont-Ferrand, France

ARTICLE INFO

Article history:
Received 15 April 2021
Received in revised form 25 January 2022
Accepted 1 March 2022
Available online 10 March 2022

Keywords:
Grid Computing
Pilot-Job
DIRAC
Site Director
LHCb

ABSTRACT

To study the constituents of matter, CERN mainly relies on the Worldwide LHC Computing Grid (WLCG), which processes petabytes of data coming from the Large Hadron Collider (LHC). LHC experiments have adopted the Pilot-Job paradigm, and deliver tools to supply grid resources with Pilot-Jobs, to efficiently leverage the computing power offered by WLCG. This sole approach will be insufficient and will need to be complemented to meet future computing needs – of the High-Luminosity LHC – and the rise of data generated over time: national science programs are consolidating computing resources and encourage using cloud and High-Performance Computing systems. Yet, even though they have started to integrate their workflows on such infrastructures, LHC experiments still largely depend on WLCG resources. This paper lays out an approach to increase the throughput of the jobs, on grid resources, by improving the performance of the Pilot-Job provisioning tools through a case study: the LHCb-specific solution, known as "DIRAC Site Director". We propose: (i) a complete analysis of the capabilities and limitations of the DIRAC Site Director; (ii) several methods to speed up its execution, including parallel processing as well as bulk operations; (iii) a comprehensive analysis of a group of Site Directors in the LHCb production environment during 12 months. With our approach, we recorded an increase of 40.86% of the number of jobs processed simultaneously per second, enabling the simultaneous management of 80,300 LHCb jobs, while only 57,000 of them could be managed before our improvements.

1. Introduction

The Standard Model of particle physics – a theory describing the fundamental particles and their interactions – has successfully explained various phenomena and experimental results, but remains incomplete and leaves many questions open [1]. To validate and develop the Standard Model of particle physics, CERN leverages a chain of particles accelerators that speed up a beam of particles before ending in the Large Hadron Collider (LHC). Inside the Large Hadron Collider (LHC), two particle beams, traveling at close to the speed of light in opposite directions, collide and provide data about constituents of matter, which are captured by four detectors corresponding to distinct experiments: ALICE, ATLAS, CMS and LHCb. Experiments capture millions of events every second that have to be filtered, processed and stored. To better understand the impact of detector effects and experimental conditions, experiments also model events occurring in the detectors by running tens of thousands of Monte-Carlo simulation applications in parallel: they both reproduce the generation of events and the configuration of the detectors [2].

CERN does not have the financial resources to process on-site the totality of the events – simulated and real – and currently relies on the Worldwide LHC Computing Grid (WLCG) [3] to deliver nearly real-time data to physicists. This infrastructure involves 170 computing centers spread within 42 countries, 1 million of computing cores and 1 exabyte of storage. More than 50 Petabytes of data are distributed and analyzed every year. To safely execute this workload on thousands of shared and distributed heterogeneous computing nodes, the four main LHC experiments implemented the Pilot-Job paradigm [4], which overcomes many intrinsic problems of the initial push model but still depends on it.

CERN has used the computing resources of WLCG to process a continuously increasing volume of data coming from the LHC experiments as a result of their constant improvements. WLCG only is no longer sufficient and needs additional support. Therefore, the WLCG communities have found ways to exploit non-reserved CPUs, often on non-formally pledged resources [5].

* Corresponding author at: European Organization for Nuclear Research, Meyrin, Switzerland.
E-mail addresses: alexandre.franck.boyer@cern.ch (A.F. Boyer), christophe.haen@cern.ch (C. Haen), federico.stagni@cern.ch (F. Stagni), david.hill@uca.fr (D.R.C. Hill).

https://doi.org/10.1016/j.future.2022.03.002

Acronyms

ALICE	A Large Ion Collider Experiment
AliEN	ALICE Environment
ARC	Advanced Resource Connector
ATLAS	A Toroidal LHC ApparatuS
BDII	Berkeley Database Information Index
CE	Computing Element
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
CREAM	Computing Resource Execution And Management
CTA	Cherenkov Telescope Array
DIRAC	Distributed Infrastructure with Remote Agent Control
DMS	Data Management System
EGI	European Grid Infrastructure
GIL	Global Python Interpreter
HPC	High-Performance Computing
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty
LRMS	Local Resource Management System
LSF	Load Sharing Facility
PanDA	Production ANd Distributed Analysis system
SLURM	Simple Linux Utility for Resource Management
VO	Virtual Organization
WLCG	Worldwide LHC Computing Grid
WMS	Workload Management System
WN	Worker Node

On the one hand, developers have made significant efforts to integrate non-grid and opportunistic resources such as cloud systems [6], supercomputers [7] and volunteering computing [8]. They have developed novel Pilot-Job and workload provisioning mechanisms to benefit from, and exploit, these new types of resources such as the Vac model [9] to spontaneously produce Pilot-Jobs instead of pushing each of them on a Site. On the other hand, there has been less focus on Pilot-Job provisioning tools dealing with shared and distributed heterogeneous clusters, such as grid resources and the push model. Yet, many Virtual Organizations (VOs) such as LHCb still mainly depend on such resources.

In this study, we want to explore whether improving the Pilot-Job provisioning mechanism bound to the push model could speed up the Pilot-Job submission frequency and, thus, could increase the throughput of the jobs on grid resources. We propose to test this hypothesis by analyzing and improving the "DIRAC Site Director" - the Pilot-Job provisioning utility used by LHCb on WLCG - and assessing the impact of the changes on WLCG over 12 months. DIRAC has been adopted by various experiments such as Belle II [10] and CTA [11], in different environments such as the European Grid Infrastructure (EGI) [12] and France Grilles [13]. Thus, if successful, the results of our study could be directly applied into these contexts, and could deliver insights to any VO dealing with a grid architecture through the Pilot-Job paradigm in a broader sense.

After the presentation of the fundamental concepts involved, such as the push model, the Pilot-Job paradigm, we describe some of the Pilot-Job provisioning tools in Section 2, as well as the DIRAC Site Director and its current limitations in Section 3. Then, in Section 4, we describe the solution proposed to increase the Pilot-Jobs submission rate and the throughput of the jobs. Results are finally assessed in Section 5 and discussed in Section 6.

2. Pilots provisioning on grid resources

2.1. Pilot-Job paradigm: the answer to the Push Model inefficiency

Despite their diversity, middleware programs interacting with many shared and distributed heterogeneous resources deal with similar physical architectures and a prevalent abstract model known as the push model. It implies transfers at many levels to dispatch the jobs: (i) from a Workload Management System (WMS) to different Computing Elements (CEs)(Fig. 1 Step 1); (ii) from a CE to a Local Resource Management System (LRMS) within Sites geographically distributed (Fig. 1 Step 2); (iii) from a LRMS to an available Worker Node (WN). In this architecture, Sites are composed of single or multiple CEs, and WNs are mostly grouped and bound to LRMS queues homogeneously. A CE provides a uniform service to submit jobs to a LRMS. ARC [14], CREAM [15] and HTCondor [16] are a popular choice of CEs among Sites. LRMSs, such as SLURM [17] and LSF[18], store jobs in the LRMS queues, and finally spread the workload among the WNs – able to execute the jobs – once they are available. LRMSs generally allocate access to resources to users for a certain duration depending on various parameters: estimated duration of the job, number of running and waiting jobs. In the same way, jobs are spending a varying duration in the LRMS queues depending on their priority, the number of running and waiting jobs and the quality of service policies of the Sites. Jobs going through the whole model often encounter issues due to the number of transfers to perform and the volatile nature of the resources. This model proved to be inefficient and error-prone according to Stagni et al. [19].

The Pilot-Job paradigm has been devised and implemented mostly to support computations across multiple distributed machines, aggregated into high-performance clusters, computing grid or virtualized in cloud infrastructures. It has been quickly adopted in the Grid Computing context as an answer to the inefficiencies of the push model. The paper, given by Casajus and his colleagues of the LHCbDIRAC team [20], defines Pilot-Job objects, also known as pilots, as "nothing more than empty resource reservation containers that are sent to the available computing resources with the final aim of executing the most appropriate pending payload in the central WMS queue" (see *Pilot* in Fig. 2). Pilots can perform basic sanity checks of the running environment (Fig. 2 Step 3) before any binding with a given payload (Fig. 2 Step 4) to effectively run jobs on well-behaved and well-adapted resources (Fig. 2 Step 5). They create an overlay network that masks the central WMS components from the heterogeneity of the underlying resources.

DIRAC proposes its implementation of the Pilot-Job paradigm where pilots, considered as simple jobs, can run on WNs. A pilot performs a DIRAC installation and checks properties of the work environment such as CPU, memory and available disk space. Then, it gets the most suitable jobs from the central DIRAC WMS server. It retrieves and checks the availability of the input data and software, executes the payload, reports the success or the failure of the execution, and uploads output data if required.

2.2. Pilot-Job provisioning tools

Pilot provisioning tools aim at automating the submission of Pilot-Jobs to the resources, ensuring high-availability and maximizing the throughput of the jobs (see *PilotManager* in Fig. 2). A lot of WMSs integrate such tools to supply WNs with pilots. As

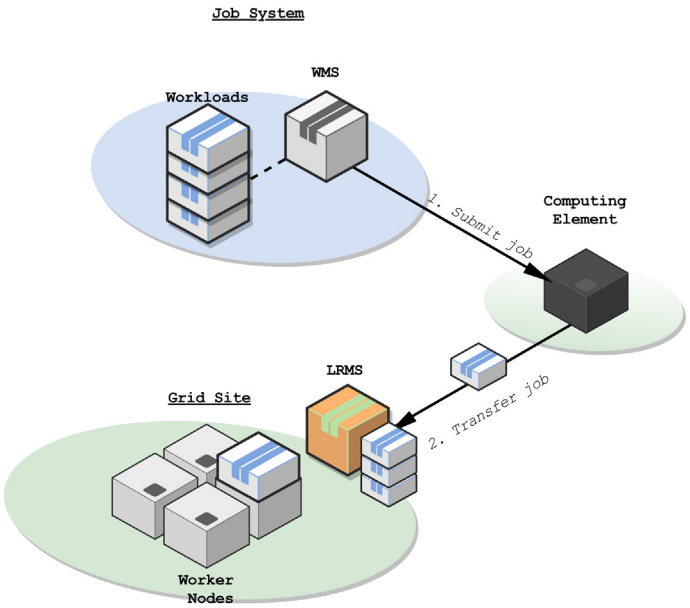

Fig. 1. Interactions between a Workload Management System (WMS) and a grid Site to execute a workload via the push model.

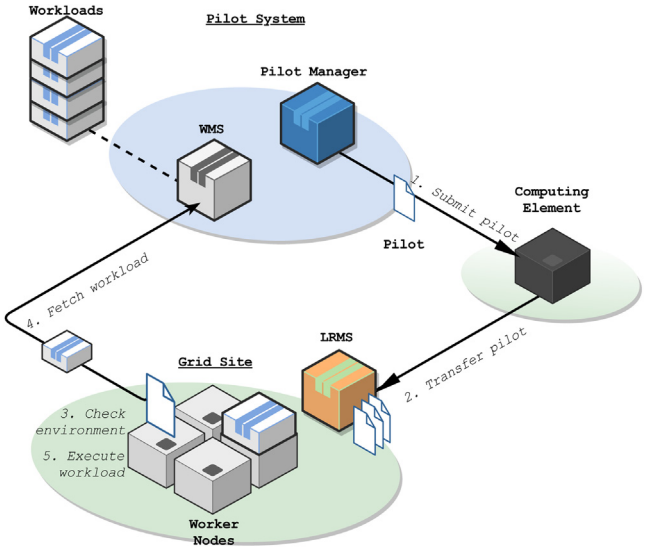

Fig. 2. Interactions between a Workload Management System (WMS) and a grid Site to execute a workload via the push model applying the Pilot-Job paradigm.

Turilli et al. underline [4], the Pilot-Job paradigm appeared as a real solution for solving the inefficient push model. We have seen an immediate development not grounded on any analytical understanding of underpinning abstractions, architectural patterns or computational paradigms, which led to a variety of Pilot-Job implementations and thus of Pilot-Job provisioning tools.

Condor [21], originally designed to allow users to execute tasks on a resource pool, was one of the first software to implement the Pilot-Job paradigm, under the name of Glideins [22], to employ the grid resources via resource placeholders. It has been quickly complemented by GlideinWMS [23] to automate and optimize the provisioning of the Glideins. In the meantime, WMSs such as DIRAC [20], PanDA [24], AliEN [25], and Coaster [26] have been developed and provide similar pilot deployment features despite slight variations.

Most of the Pilot-Job provisioning mechanisms aim at maximizing the throughput and minimizing the number of wasted resources by keeping a fixed number of pilots in the Grid pool and continuously instantiating them while there are jobs to process. The tools usually generate pilots that take the form of scripts, sent to WNs via the grid architecture and the push model. They also monitor pilots to identify failures and adjust the number of pilots to meet the demanding pressure. The characteristics and the priorities of the jobs are matched with the attributes of the resources to achieve the best binding. Rubio-Montero et al. [27] and Turilli et al. [4] emphasize the commonalities but also the differences between several WMSs in further details.

The emergence of clouds and other opportunistic resources has encouraged the development of new deployment methods. We can mention the Vac model [9], designed to leverage the advantages of clouds, or the efforts made in the Next Generation Executors to exploit High-Performance Computing (HPC) resources from an edge node and orchestrated by PanDA [28]. These approaches do not suffer from the push model of the grid to submit pilots as they do not depend on it, but grid resources cannot benefit from these developments. As a workaround, Bagnasco et al. that propose AliEN in the context of the ALICE experiment [29], give another means of provisioning grid resources [25]. The authors installed Computing Agents on several Sites that generate and submit pilots to the inner CEs without passing through the WMS.

To the best of our knowledge, Boyer et al. have been the only researchers to study the limitations of the Pilot-Jobs provisioning tools to face up the growing amount of data coming to WLCG [30]. They proposed a first analysis of the DIRAC Site Director, a few approaches to improve its performances, as well as preliminary results. Through the following sections, we will provide an extended analysis of the DIRAC Site Director limitations, novel approaches to speed up the execution of the Pilot-Jobs delivery, and comprehensive analysis of a group of Site Directors within the LHCb production environment during 12 months.

3. Analysis of the DIRAC Site Director

3.1. Presentation of DIRAC

DIRAC [31,32] is the solution adopted by the LHCb collaboration to interact with the large number of distributed computing

resources offered by WLCG. Initially developed at CERN for the needs of the LHCb experiment, this middleware benefits from a rich experience of exploiting complex computing infrastructures. It combines a WMS to handle and orchestrate jobs among distributed and heterogeneous resources, and a Data Management System (DMS), which includes automated data replication with integrity checking, needed to deal with large volumes. DIRAC pioneered the Pilot-Job paradigm within the LHC context [20]. Developers have built it as a general-purpose framework and offered it to various user communities that would also need to manage such amount of resources like Belle II [10], CTA [11], the European Grid Infrastructure (EGI)[12], and others.

3.2. Overview of the Site Director

The Site Director is a DIRAC agent performing Pilot-Job submission with the so-called push model to install pilots – mostly – on grid resources. It works in cycles, executing the same logic at each iteration. An iteration consists in:

- Getting information about LRMS queues and the WNs bound to them from the *Configuration* service first (Fig. 3 Step 1): OS installed, architecture of the WNs, maximum number of waiting pilots allowed in the queues, maximum number of pilots allowed in the Site. The *Configuration* service fetches information from the Berkeley Database Information Index (BDII), a service composed of multiple agents installed on the Sites, collecting data at the WN, Site, and Grid level according to Osman et al. [33]. The *Configuration* service also provides the Site Director with details about the CEs to connect to them: name, credentials if required.
- Querying the *Matcher* service, for each valid LRMS queue. Given a LRMS queue configuration – namely details about the architecture of the WNs and the OS installed on them – the *Matcher* service delivers a list of n_j jobs that could be executed on the underlying WNs (Fig. 3 Step 2).
- According to the number of jobs that match the configuration n_j and the slots available in the LRMS queue S, generating a certain number of pilots n_p as scripts to run on the WNs: $n_p = \min(n_j, S)$ (Fig. 3 Step 3). S the number of slots available is determined by the limits set by the Site administrators minus the number of pilots submitted in previous iterations that are still waiting or running. Pilots previously submitted are registered in the *PilotsDB* database.
- Pushing these scripts through the multiple components of the grid to reach the WNs. A Site Director only submits the necessary number of pilots, according to the jobs waiting in the queues, to avoid congesting the Sites with empty pilots. To submit the scripts to a CE, the Site Director calls a DIRAC communication interface providing the necessary tool to interact with it (see Fig. 3 Step 4).
- Registering the pilots submitted in the *PilotsDB* database on which next iterations will draw upon (Fig. 3 Step 5).
- Monitoring pilots to spot failures and provision resources accordingly. The Site Director calls the DIRAC communication interfaces to get the status of the pilots (Fig. 3 Step 6).
- Reporting the status to the *PilotsDB* database and the *Accounting* service (Fig. 3 Step 7). The *Accounting* service collects and stores data about DIRAC activities that can then be used to build reports.

It is worth noting that a Site Director is highly and dynamically configurable. Administrators can set up multiple instances that can run in parallel and manage specific Sites, CEs, and types of resources to share the workload. Additionally, they can tune parameters to modify the functioning of a Site Director: execute

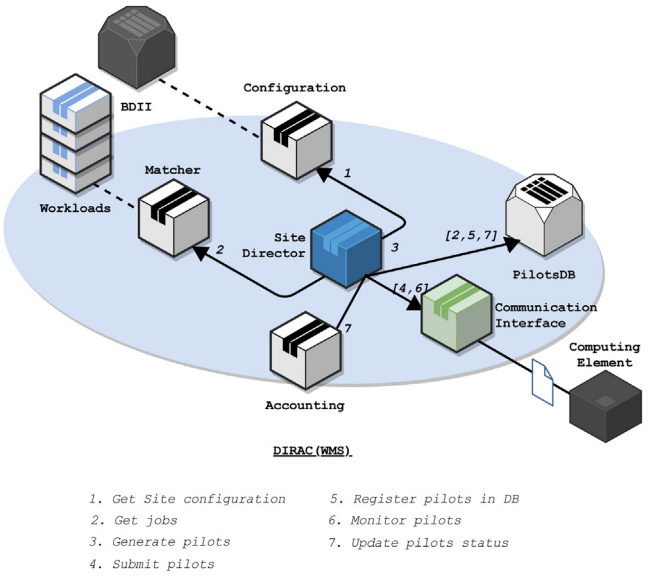

Fig. 3. An iteration of a Site Director: steps to manage Pilot-Jobs on grid computing resources.

the monitoring process every n_{update} cycle, wait n_{fail} cycles before submitting in a LRMS queue that failed, get the outputs of the pilots, etc.

By default, in LHCbDIRAC - the DIRAC instance of LHCb -, Site Directors are configured to monitor pilots every 10 cycles. They also wait 10 cycles before submitting in a LRMS queue that failed, and do not fetch the outputs of the pilots. Additionally, one cannot control the submission operations, despite the many parameters a Site Director contains. Indeed, a Site Director stops generating and submitting pilots for 10 cycles in the LRMS queues that have no more slots available. Lastly, LHCbDIRAC administrators have configured the minimum duration of the Site Director cycles to 120 s.

3.3. Limitations due to the grid architecture

We carried out an analysis in DIRAC for the LHCb experiment to emphasize the different limitations inherent to the grid architecture that could cause latencies and prevent to submit as many pilots as needed to run jobs. Since we cannot profile the production environment, we draw on the DIRAC command-line interface, the web application, as well as the log files to get insight into the Site Directors. Raw data and results from the analysis are publicly available [34].

The *Accounting* service of the web application provides the average number of jobs processed by pilot per day and per CE during a month (Fig. 4). In the context of LHCb, most of the pilot handles a single job, despite pilots have been designed to fetch and run multiple jobs. Indeed, getting an accurate value of the time left allocated to a pilot is a complex operation due to the grid heterogeneity. Site managers work with various LRMS types and versions and adjust specific features differently. Therefore, LHCbDIRAC administrators prefer to limit the number of jobs that a pilot can process, to avoid aborting the jobs that would run out of time. Thus, a Site Director generally submit a pilot per waiting job.

The web application also presents the time that pilots take from their submission to their installation on a WN. According to the records of 3000 pilots installed on 33 Sites, this duration is not immediate, and generally vary from 165 s (1-quantile) to 1719 s (3-quantile) (Fig. 5). Indeed, many VOs are competing for

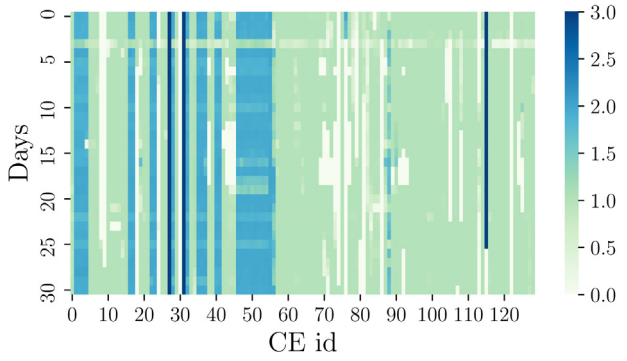

Fig. 4. Average number of jobs processed per pilot during a month, classified by the CE that was used to submit them.

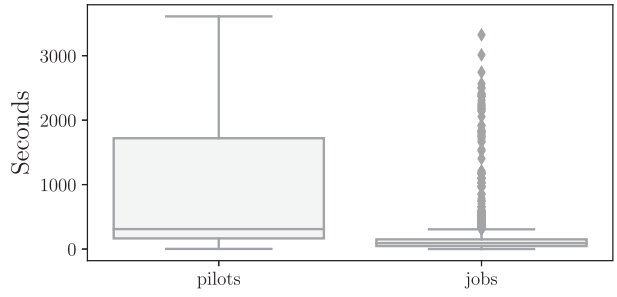

Fig. 5. Duration, in seconds, from the pilot generation to the pilot installation on a WN at the left; Duration, in seconds, from the job arrival to the job matching at the right.

limited computing resources on different Sites. LRMS of the Sites may put the pilots on hold when they arrive, while other VOs are using WNs. Fig. 5 also contains the time that 3000 jobs take from their arrival in DIRAC to their installation on a WN via a pilot. The median duration to effectively bind a waiting job to a pilot is about 92 s, while the median duration to send and execute a pilot on a WN is 309 s. The medians demonstrate that jobs are rarely processed by pilots that were generated for this purpose, outlining the importance of always having waiting pilots in the LRMS queues.

The web application contains a configuration page with the parameters of LRMS queues. LRMS queues limit the number of pilots, running and waiting, by means of two parameters: (i) *max pilots* the maximum number of pilots coming from a given queue, and bound to a VO, an LRMS can handle simultaneously; (ii) *max waiting pilots* the maximum number of pilots, bound to a VO, that a LRMS can hold in a given queue simultaneously. The number of waiting jobs in LHCbDIRAC is often significantly superior to the *max pilots* values of the LRMS queues.

Issues bound to the infrastructure remain unsolvable, as modifying the architecture in place is not a possible option. Therefore, Site Directors cannot submit as many pilots as necessary to quickly process the jobs, nor reach and maintain *max pilots* in the LRMS queues. Thus, we should focus on continuously submitting pilots to maintain *max waiting pilots* in the LRMS queues.

3.4. Limitations due to the Site Director itself

Through this part, we analyze whether the Site Director limit the production of Pilot-Jobs by itself and the reasons of such limitations if they exist.

The DIRAC command-line interface allows us to get a summary of the status of the pilots, classified by CE, at a certain moment.

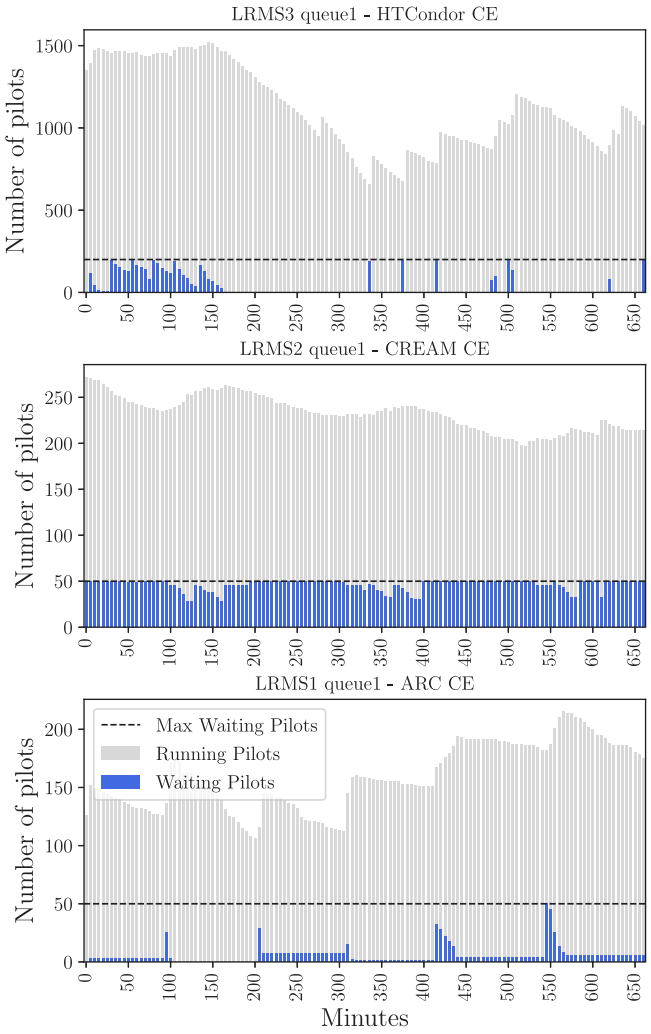

Fig. 6. Status of the pilots supervised by three specific LRMS queues for 12 h.

There are keys for every active CE, and each of them contains a list of status associated with the number of pilots currently in this state. By repeating the process every 5 min, we get plots describing the activities of the pilots associated with a specific CE or LRMS queue through time (Fig. 6). Plots only describe the activities of the pilots at a certain point in time, but we consider this sufficient to get a grasp of the limitations of the Site Director. In the same way, pilots can pass from *Waiting* to *Running* in less than 5 min, meaning some of them can only appear as *Running* on the plots, but this should not significantly impact the plots.

We notice that *max waiting pilots* is reached but rarely maintained in most cases. The Site Director bound to *LRMS3 queue1* did not submit any pilot for 2 h, whereas no pilot was queued, and running pilots were decreasing through time. We can observe similar behavior in *LRMS2 queue1* and *LRMS1 queue1* even if the latter one is less noticeable. The web application can provide information about errors that could have occurred during the submission process, but nothing was reported for the studied queues during this period. Thus, the limitation must come from the execution of the Site Director.

In the LHCb context, each Site Director is bound to specific Sites and to a specific CE type to minimize the number of LRMS queues to manage per Site Director. Its execution is recorded in a distinct log file where we can extract additional information. Each file consists of a suite of logs relative to the execution of

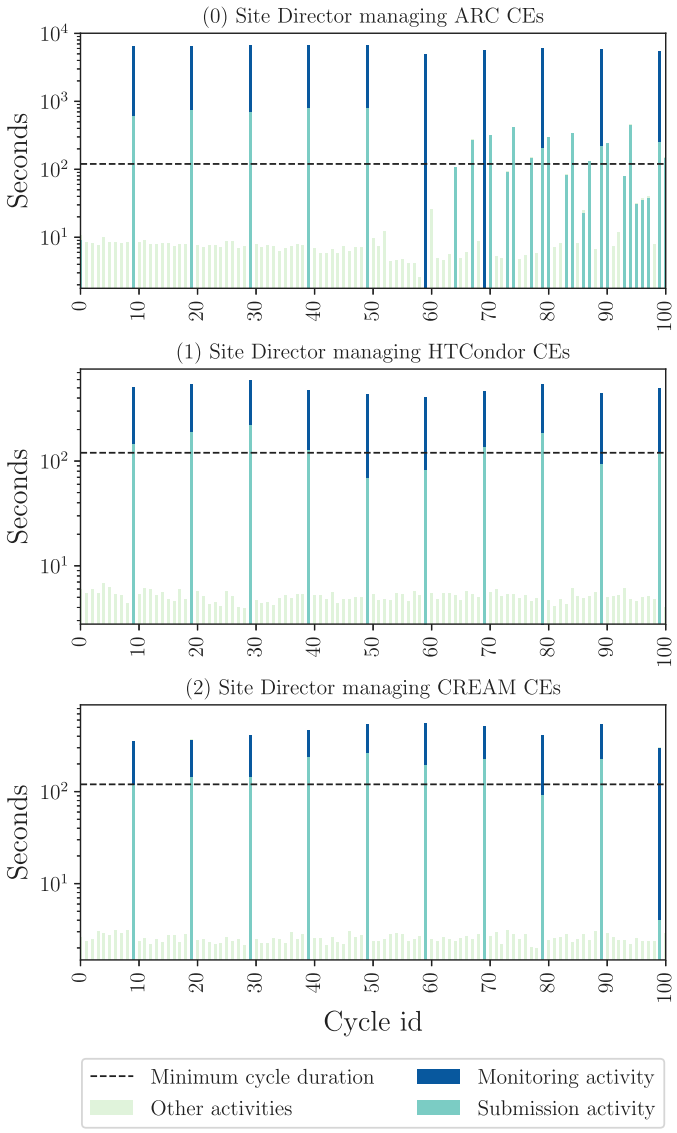

Fig. 7. Duration of the cycles and activities of three Site Directors (120 s minimum.

multiple cycles. Each log contains a date as well as a message that can constitute a landmark to extract information of interest. Information about the configuration such as the Sites, the types of CEs supervised, the number of jobs, and waiting pilots at a specific moment always appear first. Content about the submission and the monitoring activities can show up afterward. To study the logs, we developed an analysis tool that draws on repeated and common messages and their dates across the files. Its purpose is to extract useful data from a given log file and summarize them into different graphs such as Fig. 7.

Fig. 7 describes the execution's length of some Site Directors handling different CE types. The Site Director managing ARC CEs (0) can spend around 6000 s to make a cycle while it can take 500 s in (1) and (2). This difference can vary according to the number of pilots managed by the Site Directors, the type of the supervised resources, their location, and their capabilities. In this example, (1) manages slightly more pilots than (0), which indicates a potential issue in ARC resources that we are going to investigate in Section 4.2.

On all the plots, one cycle out of ten exceed the minimum cycle duration set to 120 s, despite they deal with distinct types of CEs. These specific cycles execute the submission of the pilots followed by the monitoring step, which is time-consuming, while other cycles perform short operations. The monitoring task is the longest operation according to Fig. 7, and would probably explain some of the Site Director limitation seen in Fig. 6. Indeed, while monitoring the pilots, the Site Director cannot generate and submit new pilots to fill in the LRMS queues.

One could think about isolating the monitoring part of the Site Directors into a specific agent. On the one hand, this would prevent the stops occurring in the execution of the Site Director and would ease the Pilot-Job submission. On the other hand, it would only shift the monitoring issues elsewhere and would continue affecting the Site Directors. In the same way, administrators could instantiate new Site Directors to split resources across them. Having one Site Director per CE would likely provide better results, but would partially help since it would make the maintenance part difficult. Indeed, in the context of LHCb, we have hundreds of CEs, administrators would not be able to manage so many Site Directors.

By mapping the log messages with their location within the source code, we noticed that the communication between the DIRAC server and the CEs represent the longest operations. Therefore, optimizing communication methods could probably decrease the submission and monitoring duration, prevent stops in the submission of the pilots and thus, could help to maintain *max waiting pilots* in the queues. We are going to study several approaches in Sections 4.1 and 4.2.

Better sharing the workloads between cycles could also ease the submission of pilots on a more frequent basis. In (0), the submission of pilots is more frequent for a limited time, and we observe that the submission duration decreases when shared between cycles. This only occurs when *max waiting pilots* is maintained for more than 10 cycles in some queues. We define and discuss a robust solution in Section 4.3.

The combination of both Figs. 6 and 7 suggests that the submission of pilots on a more frequent basis would help to maintain *max waiting pilots* on the short term on the one hand. On the other hand, it would likely increase the number of running pilots and thus the monitoring period that would finally stop the submission of new pilots for a while and would decrease the number of pilots available again. The main idea would be to improve the monitoring process in order to decrease its duration and submit more frequently and, consequently, reach and maintain *max waiting pilots* finding a balance between the number of pilots to submit and the monitoring time.

4. Performance improvements of the DIRAC Site Director

4.1. Parallel communication with the Computing Elements

DIRAC provides communication interfaces to communicate with different CEs. Such interfaces take the form of plugins wrapping the necessary tools to connect to a specific type of CE. They allow DIRAC services, and especially Site Directors, to interact with the underlying LRMS queues and pilots. These communication interfaces include methods to submit pilots to a given CE, kill pilots, get their outputs and/or their statuses. Operations rely on communication with remote resources and require several seconds or even minutes to get responses.

A Site Director sequentially communicates with the CEs, via the communication interfaces, to submit pilots and monitor them. Moreover, the Site Director can administer tens of CEs containing hundreds of pilots, that would involve a large number of requests to remote resources. To minimize duration first, one should privilege parallel treatments and bulk operations.

Submitting pilots involves communication with the *PilotsDB* database containing the pilot identifiers and their statuses. A Site

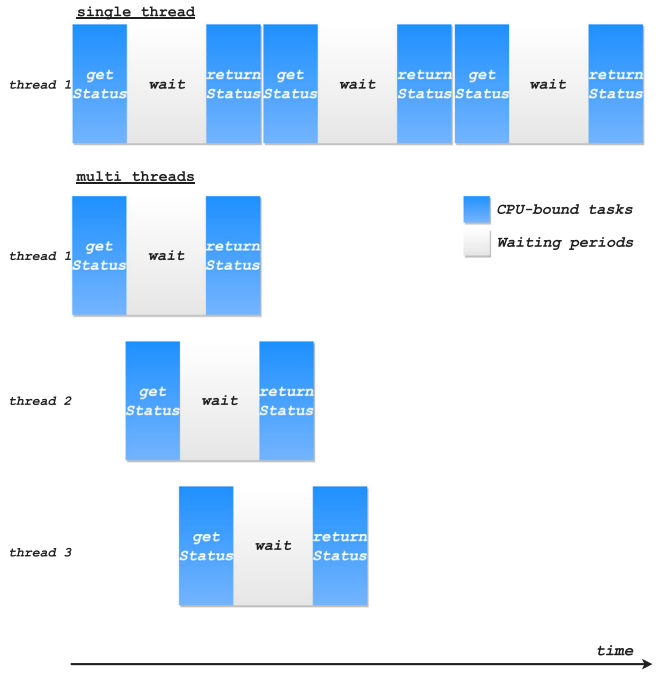

Fig. 8. Schema of a sequential execution of the monitoring task at the top; schema of a multi-threads execution of the monitoring task at the bottom.

Director reads the database before generating pilots for a given LRMS queue, and updates the pilot database after finishing a submission in this LRMS queue. Because submissions are dependent from each other, we cannot simply process submission in each LRMS queue in parallel, and thus, we focus on monitoring.

Monitoring pilots from different CEs simultaneously would probably decrease the waiting time to get remote data. As various communication interfaces exist to interact with the different types of CEs and that new types often appear, we have decided to tackle the issue at the Site Director level to preserve interfaces and avoid maintaining too many pieces of code. Classical approaches to make an application parallel include processes and threads.

Multiple processes would allow the Site Director to manage multiple CEs in parallel. However, they would mainly depend on the number of available CPUs on the DIRAC server and would not decrease the waiting time between requests. Multiple threads should, in theory, take advantage of multiple CPUs. However, as DIRAC has been written in Python, it has to deal with the Global Python Interpreter (GIL) [35]. The GIL enables concurrency by preventing multiple threads from executing Python bytecodes at once, which does not benefit CPU-bound operations. In [36], we find further details about the GIL bottleneck concerning CPU-bound threads. Nevertheless, the interpreter releases the lock on I/O operations such as reading and writing in a file or connections to external resources, which is adapted to our needs. Indeed, the monitoring task would imply IO-bound threads. Connections to the CEs would create an opportunity to switch between threads and would minimize the waiting time in the program execution.

Fig. 8 presents a Site Director requesting the status of the pilots from three different CEs, first sequentially, then using multi-threads. Each communication interface performs little CPU tasks before and after the connection, while the central part represents the waiting time due to the connection. We expect threads to switch during I/O operations to avoid the program to stop, which would result in better execution time.

4.2. Optimizations in the communication interfaces

Even though getting pilot status in each CE simultaneously would ease the monitoring of the pilots and, thus, allow the submission of a larger number of them, it remains incomplete. Indeed, CEs may interact with hundreds or even thousands of pilots, and some of the communication interfaces could be better optimized. Some of them do not exploit all the features of the underlying CEs. We have been focused on ARC, CREAM and HTCondor resources that LHCbDIRAC mainly leverages to deal with inner LRMSs.

In Section 3.4, we noticed an issue in the Site Directors dealing with ARC resources. Some of them were taking up to 6000 s to monitor a small number of pilots. The communication interface of the latter does not involve bulk methods and creates one request per pilot, which can result in a long execution time. Yet, the ARC documentation mentions the presence of such methods grouped into a module named `JobSupervisor` [37], able to gather pilot identifiers and perform a single connection to cancel and clean them, renew their credentials, get their status and their outputs. While its integration within the interface would remove the overhead generated by the sum of several single requests, a too large number of pilots supervised would also generate timeouts. Thus, we split the pilots into mid-size chunks as input of the `JobSupervisor` to efficiently use it.

Additionally, we have worked on the CREAM communication interface and especially the proxy renewal frequency. Indeed, a pilot requires a proxy to interact with DIRAC, mainly to fetch a job to run. A proxy has a limited lifetime and can expire while the pilot waits for available resources in a LRMS queue, which can lead to its abortion. To address this issue, before getting the status of a pilot, most of the communication interfaces perform a check of the proxy lifetime left and renew it if necessary. The communication interface attached to CREAM does not perform this checking and renews the proxies of chunks of pilots in multiple requests every cycle involving the monitoring, which remains unnecessary. Renewing them every n cycles, n being larger than m the number of cycles to wait before invoking the monitoring would reduce the amount of time spent to monitor the pilots on this kind of CEs occasionally. We set m to 600 by default, and we assume it would be always sufficient.

Finally, we have focused on the way DIRAC gets pilot outputs from HTCondor CEs. HTCondor CEs require an output location prior to the pilot execution, contrary to CREAM and ARC CEs. The communication interface has to define this location before the submission of a job. Moreover, HTCondor provides outputs asynchronously, which prevents the communication interface to control the number of outputs arriving at the defined location. To avoid getting too many files in the same directory, and because DIRAC may interact with multiple HTCondor CEs sharing the same job identifiers, DIRAC developers decided to fully randomize the output location at every submission but did not store these locations. Therefore, the communication interface has to perform a `find` command to get pilot outputs on the server, which can be cumbersome. To address this issue, we have decided to build the output location based on deterministic attributes that we can retrieve from the pilot identifier: the CE name, the HTCondor job identifier, and a unique pilot stamp. A deterministic path would ease the research of specific outputs and would be faster than a `find` call in most of the cases.

4.3. Pilot-Job submission pace

A Site Director regulates the number of pilots to submit in a given LRMS queue according to: (i) the number of running and waiting pilots, related to this queue, at a given time (*pilots*); (ii)

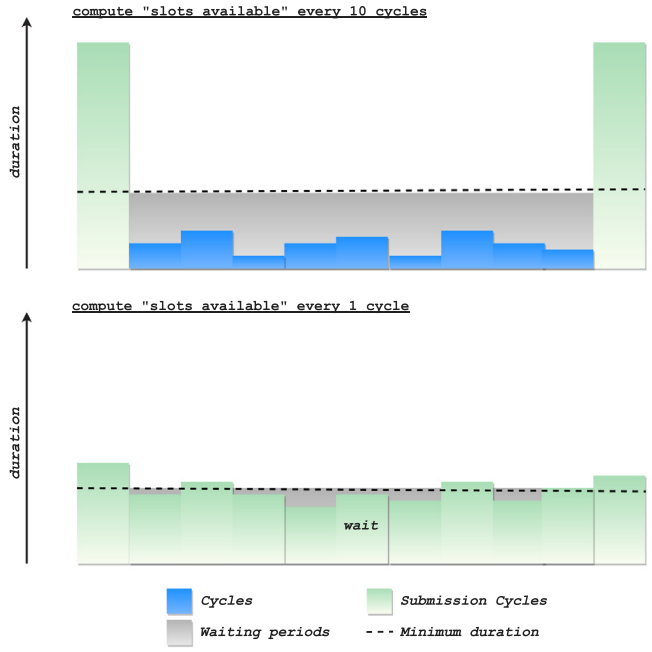

Fig. 9. Schema of the duration of the cycles when the number of slots available is computed every ten cycles at the top; schema of the duration of the cycles when the number of slots available is computed every cycle at the bottom.

the limit values it can support, namely *max pilots* and *max waiting pilots*. Thus, we have:

$$\begin{aligned} \textit{pilots to submit} = \textit{min}((\textit{max pilots} - \textit{pilots}), \\ (\textit{max waiting pilots} - \textit{waiting pilots})) \end{aligned} \quad (1)$$

The Site Director computes this number before each submission in a given queue to fill in every slot. To avoid having too many submissions of pilots that could slow down the monitoring process afterward, DIRAC developers chose to stop computing the number of slots available in a given LRMS queue for 10 cycles once slots have been filled. Thus, a Site Director waits for 10 cycles, a minimum of 1200 s, before submitting to the given LRMS. There are two main problems in this approach: (i) there is no mechanism to balance the submissions of pilots between different cycles and they often occur before the monitoring operation, which creates an overused cycle; (ii) LRMS queues can quickly install pilots on WNs and could get new ones in less than 1200 s. These problems probably explain the lack of submitted pilots sometimes, previously seen in Section 3.4. Therefore, we have introduced a new configuration option that intends to tune the number of cycles to wait before computing the number of slots available in LRMS queues. This would allow us to split the submission operation between the different cycles and, combined with the monitoring optimizations, would better meet the demanding pressure. Fig. 9 emphasizes the benefits of such an approach.

5. Performance assessment of the DIRAC Site Director

5.1. Assessment of the individual changes

We measured the changes that we described in Section 4.2 to assess their distinct contribution. From a DIRAC client, we wrote programs involving multi-threads and communication interface developments to get the pilot statuses.

First, we study the benefit of multi-threads integrated within the Site Directors. A program computes the monitoring process,

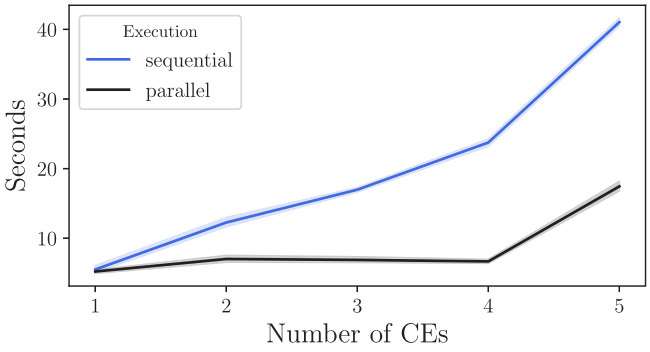

Fig. 10. Mean duration, in seconds, that a Site Director spends to monitor tens of pilots managed by a range of CEs: from 1 to 5; along with error bars representing the standard deviation.

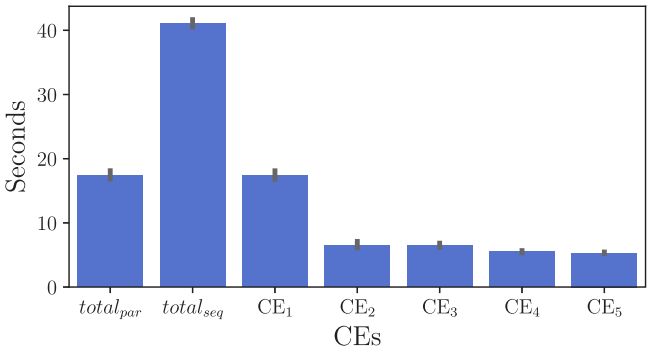

Fig. 11. Mean duration, in seconds, that a Site Director spends to monitor pilots in different Sites, managed by different CEs; along with error bars representing the standard deviation.

both in parallel and sequentially. It supervises different types of CEs, handling a diverse number of pilots. Fig. 10 summarizes 10 program executions in a plot, both sequentially and in parallel. The more CEs the program monitors, the larger the gap between both methods from what we can observe. However, the duration does not increase linearly.

Fig. 11 provides an average of the 10 program runs and details about each CE involved. The length of the sequential execution, $total_{seq}$, corresponds to the sum of every CE_n, defined as the time spent by a CE to get the pilot statuses. On the other hand, the parallel version duration, $total_{par}$, is close to CE_1, which is the longest one. The standard deviation remains low and results demonstrate the efficiency of the threads and confirm this choice in this context.

We also measured changes brought to the communication interfaces. We evaluated the `JobSupervisor` integration into the ARC interface. First, the evaluation program performs single requests as it was originally the case, and then bulk requests, leveraging the `JobSupervisor`, to get the pilot statuses. The program execution was launched three times to get an average of the results as well as a standard deviation. Three CEs from distinct Sites, each of them handling 47 pilots, were available during the experiment. The results appear in Fig. 12. We can observe a significant improvement in these CEs. Indeed, in all of these cases, it takes less than a second to monitor the pilots using the `JobSupervisor` while it can reach 17 s for the same treatment employing a request per pilot. The higher the number of pilots, the larger the gap between the processes. By computing a linear regression on a CE, we can obtain a theoretic time to process 500 pilots. For instance, based on the available data, CE_1 would spend 82 s to monitor such a number of pilots using a request per pilot while a bulk operation would take 3.7 s.

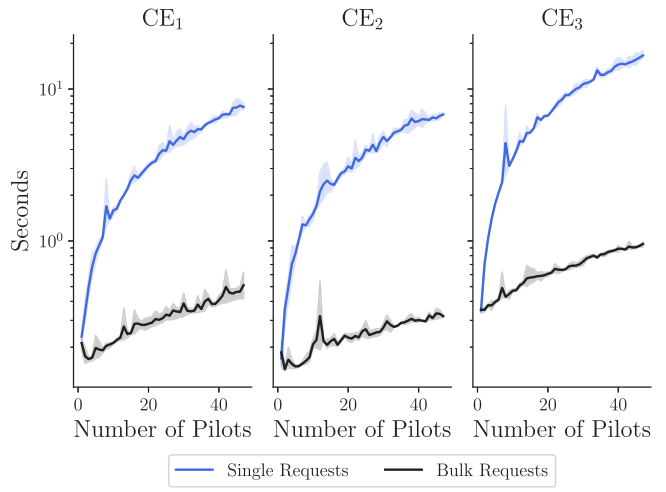

Fig. 12. Mean duration, in seconds, of the single and bulk requests in ARC resources along with error bars representing the standard deviation.

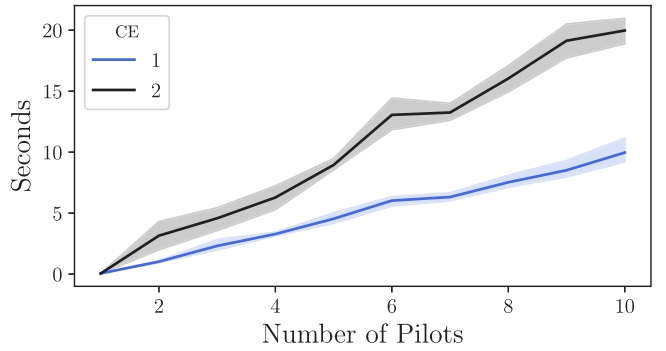

Fig. 13. Mean duration, in seconds, of the proxy renewal in CREAM resources along with error bars representing the standard deviation.

Changes related to CREAM depend on the number of pilots and the time spent to renew a proxy. However, Fig. 13 estimates the time that a Site Director would spend to renew the proxies of a certain number of pilots through CREAM resources, and offers a brief idea of the time that could be saved in some cycles. The program renews the proxies of 1 to 10 pilots on two CEs from different Sites five times. The duration scale between the CEs is varying because of their location and their capabilities. We could set up better and more accurate means to renew the proxies of the pilots, but the support of CREAM has ended and resources of this kind should progressively disappear. Furthermore, they would not be so worthwhile in terms of running time.

Getting the benefits that the deterministic path setup to retrieve HTCondor pilot outputs could bring is a complex operation that would be meaningless. Indeed, the performance of the `find` command depends on too many factors such as the disk utilization and the server capabilities, which can vary a lot through time. Moreover, administrators disabled the parameter to get the pilot outputs in production. Similarly, evaluating the gain of the option to finetune the submission pace individually would depend on too many external factors such as the underlying occupancy of the LRMS queues.

The next subsections of our work assessment aim at providing insights: (i) about the evolution of the throughput of the jobs and the pilot submission frequency over time; (ii) about the involvement of the changes. Such points are necessary to answer our initial research question: Does the improvement of the Pilot-Job provisioning tool speed up the Pilot-Job submission frequency and, by extension, the throughput of the jobs on grid resources? This will be plainly answered and discussed in the next Section 6, after the presentation of our results.

Table 1
Site Directors from the LHCbDIRAC production environment removed from the study.

Site Dir.	Reason
SD_2	Added during Phase3
SD_5	Changes of CEs (ARC to HTCondor) since Phase3
SD_8	Changes of CEs (CREAM to HTCondor) since Phase2
SD_9	Changes of CEs (CREAM to HTCondor) since Phase2
SD_{11}	Removed during Phase3
SD_{13}	Removed during Phase2
SD_{15}	Changes of CEs (CREAM to ARC) since Phase2
SD_{16}	Removed during Phase3
SD_{18}	Removed during Phase2
SD_{23}	Changes of CEs (CREAM to HTCondor) since Phase2
SD_{25}	Manage SSH CEs
SD_{26}	Manage SSH CEs
SD_{27}	Manage SSH CEs
SD_{28}	Manage SSH CEs
SD_{29}	Manage SSH CEs
SD_{30}	Added and removed during Phase2
SD_{31}	Added during Phase3

5.2. Evaluation of the LHCbDIRAC production environment: experimental conditions

We analyzed the Site Directors of the LHCbDIRAC production environment for 12 months to assess the contributions mentioned in this paper in a real use case. Raw data, results and figures are publicly available [34]. We introduced three different phases:

- Phase1: does not include any of the change (from week 1 to week 17).
- Phase2: include changes related to the monitoring task seen in Sections 4.1 and 4.2 (from week 18 to week 41).
- Phase3: include changes related to the submission pace control seen in Section 4.3 (from week 42 to week 56).

Getting the overall benefit of the changes is a complex operation. Indeed, Site administrators can add, replace, remove computing resources, LRMS queues and CEs over time. They can also modify the scheduling policies; grant a varying number of slots to VOs. Besides, DIRAC administrators can tweak parameters related to the Site Directors such as the Sites and CEs they manage.

Over the analysis period, Sites, as well as all the Site Directors, were modified. We removed data related to the Site Directors instantiated, largely modified, or deleted during the three phases, as it would skew the study (Table 1). This mainly concerns Site Directors managing deprecated CREAM CEs. We also removed Site Directors managing pilots via SSH, as we did not observe such a use case through this paper, and they deal with a minor part of computing resources.

Thus, the study includes 13 out of 31 Site Directors managing pilots within a total of 65 out of 77 Sites: five of them manage ARC CEs, five others interact with HTCondor CEs, and the last ones supervise CREAM CEs. We configured the pilot submission pace according to the type of CEs that Site Directors deal with: ARC Site Directors submit every cycle while CREAM and HTCondor Site Directors submit every 5–6 cycles. Selected Site Directors were present during the three phases and received small adjustments over time such as LRMS queues added or removed. Table 2 classifies the number of LRMS queues managed by the Site Directors and the changes that occurred during the different phases.

Table 2
Selected Site Directors from the LHCbDIRAC production environment and their evolution over the different phases.

Site Directors	Phase1	Phase2		Phase3	
		Added queues	Removed queues	Added queues	Removed queues
SD_1	17	6	0	0	0
SD_3	5	9	0	8	0
SD_4	2	1	0	0	1
SD_6	3	0	0	0	0
SD_7	2	1	1	0	0
SD_{10}	1	0	0	0	0
SD_{12}	9	0	0	0	0
SD_{17}	29	4	0	0	0
SD_{14}	2	0	0	0	0
SD_{19}	4	0	0	0	0
SD_{20}	5	0	0	0	0
SD_{21}	4	0	0	0	0
SD_{22}	8	10	0	3	1

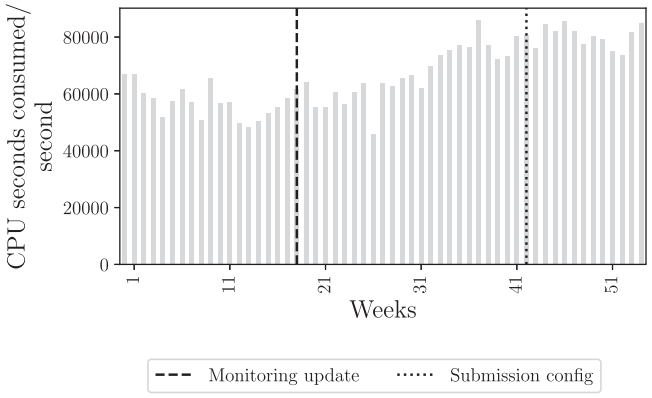

Fig. 14. CPU seconds processed per second by LHCb jobs on selected Sites over 12 months, averaged per week.

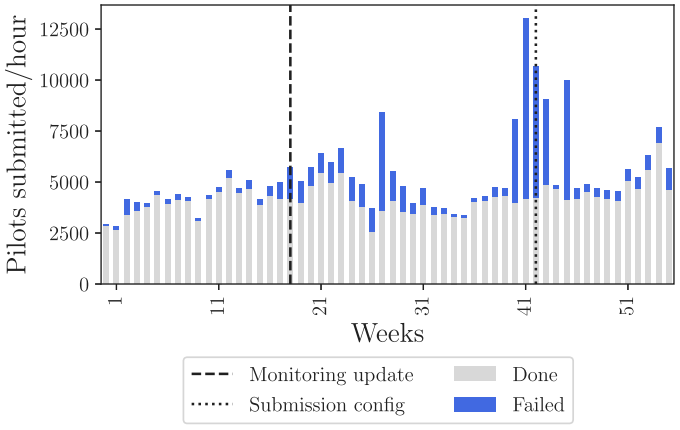

Fig. 15. Number of pilots submitted per hour, averaged per week.

5.3. Evaluation of the LHCbDIRAC production environment: evolution of the throughput of the jobs and the Pilot-Job submission frequency

First, we assessed the number of CPU seconds processed per second (Fig. 14). The metric corresponds to the number of jobs running simultaneously within the Sites observed. It also represents the number of CPUs that the LHCb VO can exploit in parallel on grid resources to process the workload.

In Fig. 14, we averaged values per week. Dashed and dotted lines designate the limits between the phases. We also grouped and averaged the values by phase. LHCbDIRAC processed 40.86% more CPU seconds per second in Phase3 (80 306) than in Phase1 (57 010). The largest increase occurs between Phase2 and Phase3 (21.64%) but the gap between Phase1 and Phase2 remains meaningful (15.81%). Yet, we cannot notice a clear distinction between the different phases. While the standard deviation is relatively small in Phase1 (5917) and Phase3 (3675), it is larger in Phase2 (9121). Values from Phase2 almost linearly increase from about 55,000 CPU seconds/second in the first weeks to about 70,000 in the last ones.

Moreover, we studied the evolution of the number of pilots submitted – by the selected Site Directors – per hour, averaged per week. Fig. 15 illustrates the number of successfully submitted pilots per hour along with the pilots that Site Directors failed to submit.

Dashed and dotted lines delimit the phases. We grouped and averaged the values by phase. Site Directors intended to submit 60.23% more pilots per hour in Phase3 than in Phase1, with an increase of 33.10% between Phase2 and Phase3, and 20.38% between Phase1 and Phase2. Nevertheless, the evolution of the number of successfully submitted pilots per hour is much lower: values remained constant between Phase1 and Phase2 (+1.29%) and rose between Phase2 and Phase3 (16.90%). We can also see peaks in Phase2 and Phase3 that were not reached in Phase1. The evolution of failed submission per hour is much more noticeable in the figure (+671.41% between Phase1 and Phase3), but remains highly variable within the phases: the standard deviation is about 725 in Phase1, 1325 in Phase2 and 2693 in Phase3. As we can observe, there is no clear association between errors and phases, and most errors seem concentrated at the end of Phase2 and the beginning of Phase3. We provide further details about errors in Section 5.4.

Additionally, to get a more accurate idea of the involvement of the changes, we focus on the status of the pilots for small periods during the phases. Using the command-line interface, we got the status of every pilot on all the observed Sites between 144 and 432 times per phase. Fig. 16 presents the distribution of waiting pilots per Site Director over time, classified per phase.

We summed the median value of each Site Director per phase. LRMS queues contain 15% more waiting pilots between Phase1 and Phase2, and 53% more between Phase2 and Phase3. We can also observe less variability in Phase3 than in Phase1, indicating that values are relatively stable in general. Most of the ARC Site Directors in (0) manage many more waiting pilots in Phase3 than in Phase1 (between +144% to +805%) except SD_4 (−76%). HTCondor Site Directors in (2) propose similar results: more waiting pilots in Phase3 than in Phase1 with less variability (between +14.60% and 62.88%). SD_{22}, which has handled a growing number of queues in Phase2 and Phase3 according to Table 2, monitors

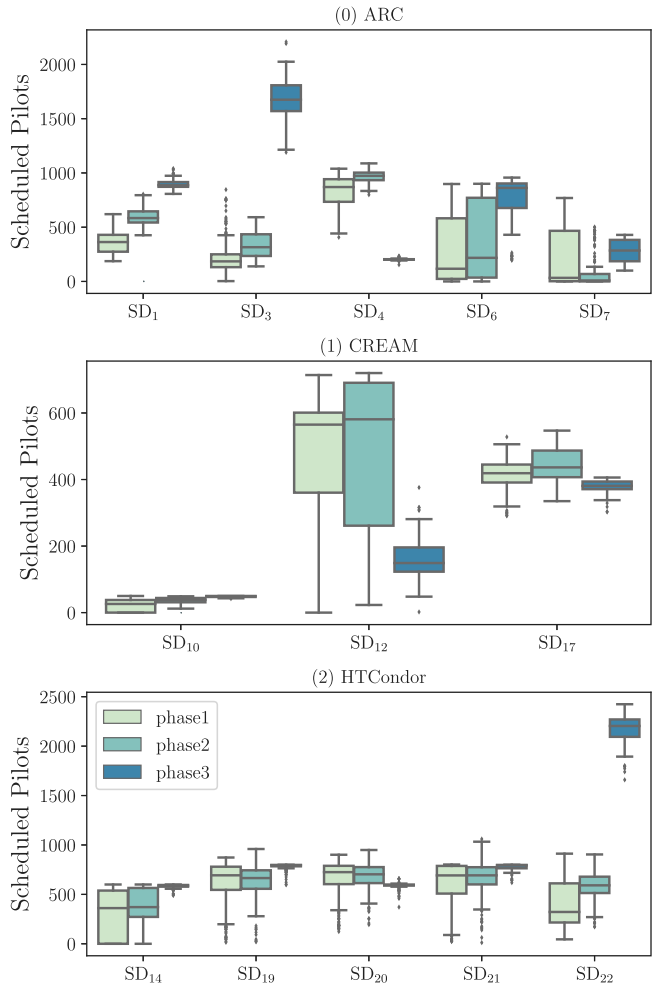

Fig. 16. Distribution of waiting pilots per phase, classified by Site Director. (0) gathers Site Directors managing ARC CEs; (1) gathers Site Directors dealing with CREAM CEs; (2) gathers Site Directors interacting with HTCondor CEs.

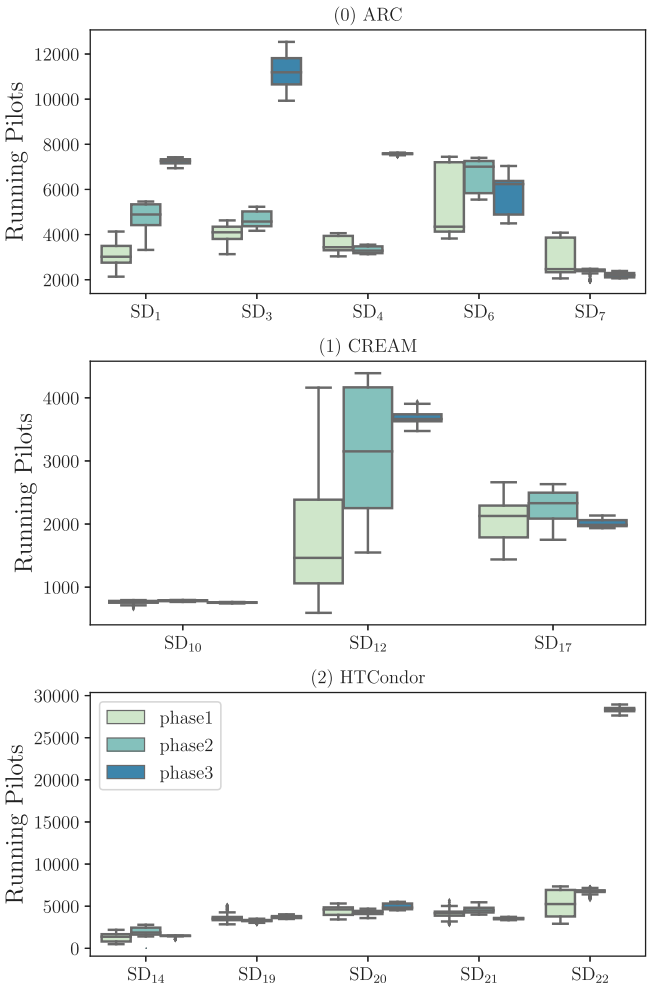

Fig. 17. Distribution of running pilots per phase, classified by Site Director. (0) gathers Site Directors managing ARC CEs; (1) gathers Site Directors dealing with CREAM CEs; (2) gathers Site Directors interacting with HTCondor CEs.

many more scheduled pilots since the beginning of the Phase3 (+584% compared to Phase1). At the same time, the number of scheduled pilots coming from SD_{20} declined in Phase3. On the contrary, in (1), CREAM Site Directors interact with a more stable number of waiting pilots, but we can notice a global decrease in Phase3 compared to Phase2. Results from SD_{10} remained low and almost identical through the different phases. To complete these data, we also analyzed the distribution of running pilots per Site Director over time, classified per phase in Fig. 17.

We also summed the median value of each Site Director per phase. LRMS queues contain 21% more running pilots between Phase1 and Phase2, and 68% more between Phase2 and Phase3. In (0), there were many more running pilots in resources managed by SD_1, SD_3 and SD_4 in Phase3 than in Phase1 and Phase2 (between +43% and +172%). Yet we noticed in Fig. 16 that SD_4 monitored fewer waiting pilots in Phase3. Similarly, SD_7 monitored less running pilots in Phase3 than in Phase1, while it had more waiting pilots in Phase3. Most of the results in (2) rose between 4% and 8% between Phase1 and Phase3, which remains significant according to the large number of pilots these Site Directors manage. As in Fig. 16, SD_{22} monitored a larger number of running pilots in Phase3 than in Phase1: more than 25,000 running pilots at the same time, which represents the highest number of pilots monitored at the same time by a Site Director. SD_{20} have also less running pilots. To finish, in (1), SD_{10} results did not change through time, while results from SD_{12} grew (+150%) and the ones from SD_{17} went down (−6.53%).

While general results provide meaningful data about the evolution of the LHCbDIRAC production environment through time, they do not furnish any information about monitoring duration and submission pace. After some details dealing with failed submissions, involvements of the paper contributions are analyzed in depth in Section 5.5

5.4. Evaluation of the LHCbDIRAC production environment: details about failed submissions

The changes made could have potentially generated new errors: in this part, we provide additional information about errors at the Site Director level to strengthen the answer to our initial research question in Section 6.1.

Table 3 provides details about the number of errors that occurred at submission within each Site Director. Errors mainly concern 8 out of the 13 Site Directors of the study. We grouped Site Directors that got a small number of errors in the *Others* category. Most of the Site Directors encountered more issues in Phase2 than in Phase1, but the number fell in Phase3. Among us, SD_1 got even fewer errors in Phase3 (399) than in Phase1 (662). Only three Site Directors got more errors in Phase3 than in Phase2: SD_{19}, SD_{20}, and SD_{21} that interact with a common Grid

Table 3
Number of failed submission per Site Director, classified per phase.

	Phase1	Phase2	Phase3
SD_1	662.920238	5 535.317857	399.515476
SD_3	86.851190	5 528.092857	170.407143
SD_4	51.630952	1 861.404762	666.035714
SD_{17}	653.317857	1 581.589286	4 219.523810
SD_{19}	957.285714	2 165.017857	16 975.357143
SD_{20}	871.130952	1 742.678571	5 566.863095
SD_{21}	595.896429	5 972.588095	2 636.358333
SD_{22}	504.809524	1 231.464286	612.035714
Others	217.190476	324.409524	71.126190
Total	4601.033333	25 942.563095	31 317.222619

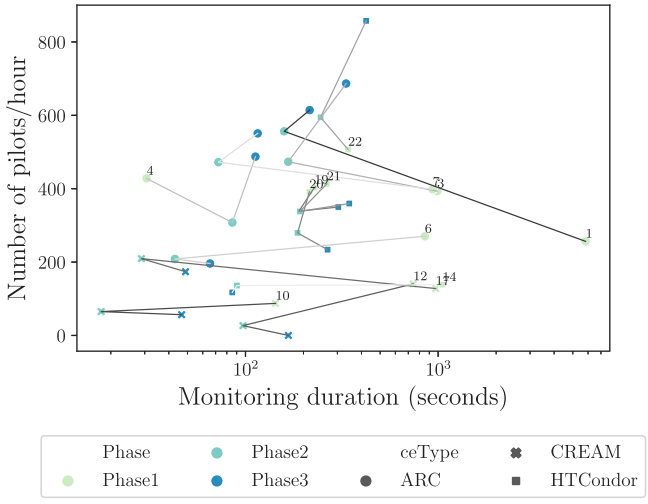

Fig. 18. Evolution of the number of pilots successfully submitted per hour (mean), function of the monitoring duration of the Site Directors through the different phases. Each point represents a Site Director; its shape, a type of CE managed; and its color, a certain phase. Phases of a same Site Director are associated via a line.

Site. During Phase2, 65% of the errors were bound to three Site Directors: SD_1, SD_3 and SD_{21}; while they were only bound to 10% of the total number of errors in Phase3. Indeed, in Phase3, SD_{19} was bound to 54% of the errors.

5.5. Evaluation of the LHCbDIRAC production environment: involvement of the contributions

Decreasing the monitoring period is the primary way that we choose, in this paper, to go towards an efficient Pilot-Job provisioning on grid resources. We investigated the evolution of the monitoring through the different phases and especially between Phase1 and Phase2. We extracted the monitoring duration from the logs of the Site Directors – each log file contains around 150 values – and we computed the mean for each Site Director and each phase. We coupled monitoring values with successfully submitted pilots from Fig. 15 and, thus, we obtained Fig. 18.

Monitoring duration dropped down from Phase2 (−49% on average). Indeed, in Phase2, SD_{22} spent the longest time monitoring its pilots: 245 s on average; while 6 Site Directors spent more than 850 s monitoring their pilots on average during Phase1. In Phase3, monitoring duration slightly went up (+64% compared to Phase2 on average, but −22% compared to Phase1): SD_{22} spent 423 s, on average, which remained the longest monitoring duration. SD_4 was the only Site Director to increase its monitoring duration through time (from 30 s in Phase1 to 85 s in Phase2 and 112 s in Phase3). In Phase3, 7 out of 13 Site Directors spend more than 120 s, the minimum cycle duration by default, against 12 during Phase1.

Significant results on the monitoring duration do not always involve an increase in the number of pilots successfully submitted per hour, according to Fig. 18. Indeed, 7 Site Directors submitted fewer pilots per hour from Phase2 despite spending less time on the monitoring operations. It involves 2 of the 3 CREAM Site Directors, one ARC Site Director and 4 of the 5 HTCondor Site Directors. Changes have a more significant impact on ARC Site Directors than on CREAM and HTCondor Site Directors. HTCondor Site Directors provide diverse results: (i) SD_{14} submitted almost the same number of pilots over time; (ii) SD_{17} submitted more pilots per hour in Phase2, but the value went down from Phase3; (iii) SD_{22} submitted many more pilots in Phase2 and especially in Phase3; (iv) SD_{19}, SD_{20} and SD_{21}, that work on the same Site, submitted fewer pilots over time but handled many errors according to 3.

Increasing the number of pilots submitted per cycle is the second way – and inherent to the monitoring duration – that we choose to go towards an efficient Pilot-Job provisioning on grid resources. After configuring the pilot submission pace for each Site Director, we investigated the number of pilots submitted per cycle per Site Director and per phase (Fig. 19). We extracted the number of pilots submitted per cycle from the logs, which contain around 1500 values per log file.

In Phase1, Site Director submitted a median value of 0 pilot per cycle as submission occurred every 10 cycles, which correspond to outliers on the plot. ARC Site Directors, after Phase3, submitted every cycle: they were all able to submit between 10 and 20 pilots per cycle (median value) and outliers were, in general, smaller than in Phase1. Results from CREAM and HTCondor Site Directors are less meaningful, which was expected as they have been configured to submit pilots every 5 or 6 cycles. Only SD_{10}, SD_{14} and SD_{22} submitted pilots more often.

6. Discussions

6.1. Does the improvement of the Pilot-Job provisioning tool speed up the Pilot-Job submission frequency and, by extension, the throughput of the jobs on grid resources?

The capacity of LHCb to leverage Grid Site resources considerably rose over a year, and is probably the result of a combination of numerous factors.

According to Fig. 18, changes applied on the monitoring step have, overall, considerably decreased the monitoring duration of the Site Directors, and especially the ones managing ARC CEs and the ones having a large number of queues. Indeed, contributions seem to have a greater impact on Site Directors sharing these characteristics according to Figs. 12 and 10. Effects on the monitoring seem to last: Site Directors managing additional queues and hundreds or even thousands more pilots have spent less or almost the same time monitoring the pilots (see SD_1, SD_3 and SD_{22} on Fig. 18). Yet, the changes have had almost no visible impact on the number of pilots submitted per hour. Indeed, changes have decreased the duration of the cycles to a value close to 120 s, the minimum duration of a cycle, but did not change the submission pace. Thus, changes have only affected Site Directors that spent a long time monitoring pilots, blocking the submission process, such as SD_1.

Tweaking the submission pace after decreasing the monitoring duration has been substantial. Site Directors submitting pilots more often, such as the ARC ones, have better shared the workload between their cycles (Fig. 19). They have filled LRMS queues with waiting pilots more rapidly, to reach and maintain

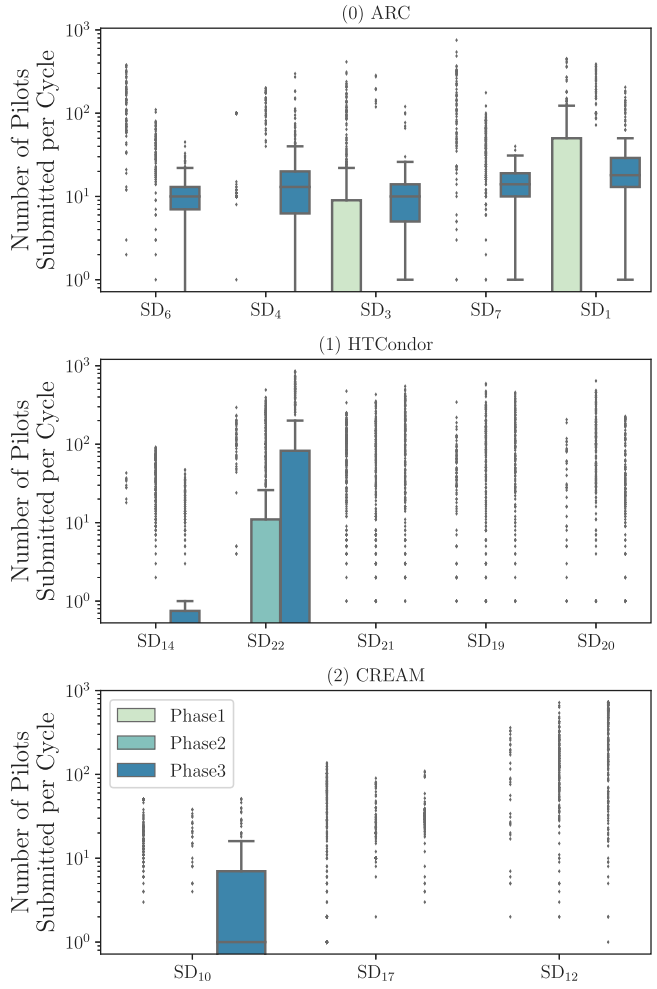

Fig. 19. Evolution of the number of pilot submitted per cycle (median), classified by Site Directors and phases.

max waiting pilots as we can observe in Fig. 16. Thus, Grid Sites have had more pilots at their disposal for available resources, which probably explains the rise of running pilots (Fig. 17).

Some external factors have complemented the results of the solutions. Many Site Directors have managed a various number of LRMS queues through time according to Table 2. Overall, there was an increase of LRMS queues: 42 added against 3 removed. Site administrators have also tweaked *max pilots* and *max waiting pilots* over time according to the log files, but we did not keep track of the fluctuations. Therefore, we cannot know with accuracy whether Site Directors have maintained *max waiting pilots* in the different LRMS queues. Such variations have probably significantly modified the monitoring duration of the Site Directors along with the number of pilots that they have supervised (see SD_3 and SD_{22} in Figs. 16–18). Our changes have helped Site Directors to support a growing number of LRMS queues that better handle the generation of new pilots and maximize the use of new computing resources.

Errors have diluted the effect of the changes. Indeed, some Site Directors have failed to submit a large number of pilots. Many failures have been likely independent from our contributions: SD_{19}, SD_{20} and SD_{21} have failed to submit plenty of pilots, but errors occurred for a limited period and concerned the same Grid Site (Table 3). Overall, we noted a larger number of submission failures after the changes. Further investigations in the logs suggest errors within the CEs and queues but remain unclear. These errors were already existing in Phase1: changes have probably eased the submission process, even within these queues, which highlights them.

The role of the following components remains unclear, and their effects are difficult to measure. First, the number of jobs per pilot has changed over time (Fig. 4). In general, the more jobs a pilot handles, the longer it remains on a WN. In practice, it also depends on the execution duration of the jobs, which leads to the second point. Jobs and pilots have had a varying execution duration. A short job triggers: (i) the generation of a pilot that can take several minutes before running; (ii) the allocation of a WN for a limited time - a few seconds for instance -; which is not efficient. In theory, the repetition of a large number of long jobs would reduce the risk of having unused resources: Site Directors would have more time to generate pilots and they would be replaced less often in the WNs. We did not keep track of such information over time as it would represent a massive amount of data.

Factors have had a direct impact on the number of pilots submitted per hour. Yet, some Site Directors – that have been positively affected by our contribution and not significantly impacted by external factors that we can measure – have not produced and submitted plenty of pilots: SD_4, SD_{10} and SD_{12} for instance. Some Site Directors were probably already submitting *max waiting pilots* in the LRMS queues, such as SD_{10}, which had the same number of scheduled and running pilots over time (Figs. 16 and 17) Other Site Directors were likely supervising a considerable number of running pilots, which modified *max waiting pilots*. Indeed, the value of *max waiting pilots* depends from *max pilots* such as:

$$max\ waiting\ pilots = min(max\ waiting\ pilots, max\ pilots - running\ pilots) \quad (2)$$

This is probably the case for SD_4 and SD_{12} that have a low and steady number of scheduled pilots in Phase3 (Fig. 16) while having a large and steady number of running pilots (Fig. 17).

The combination of all these component have likely increased and stabilized the number of waiting pilots in the LRMS queues (Fig. 16) and have even allowed LHCb to exploit a larger number of allocations on Grid Sites (Figs. 17 and 14). Besides, the number of running pilots approximately corresponds to the number of CPU seconds processed per second over time (Figs. 17 and 14).

This study has mainly focused on grid resources, especially in the context of the LHCb experiment, but remains appropriate for any pool of distributed and shared computing resources aggregated into high-performance clusters and clouds. VOs relying on supercomputers providing external connectivity, or cloud resources orchestrated by LRMS, could reuse parts of the presented content.

6.2. Future directions and challenges

This section provides further recommendations on topics not covered by this study, to help VOs depending on the distributed architecture to better exploit shared computing resources.

CEs may have different numbers of pilots to handle, as well as heterogeneous performances to execute a similar operation, as we can observe in Figs. 11–13. To better leverage the multi-threads integration, DIRAC administrators can bind Site Directors to multiple CEs, and especially CEs having similar performances. Indeed, the time spent in each thread should be equally distributed to avoid having one thread spending more time than all the other together, which would result in a duration close to the sequential one. One could propose an automated way to balance

the number of Site Directors and the number of CEs per Site Director to minimize the duration of the cycles while maximizing the submission frequency.

Again for DIRAC administrators, in Section 3.3, we have demonstrated that jobs were rarely processed by pilots generated for this purpose, which may call into question the need to check the presence of jobs, before instantiating the pilots. Indeed, a Site Director only generates a limited number of pilots according to the jobs available that could run in a given resource, as well as the number of free slots. When pilots are finally running, this limited number may not reflect the number of jobs previously available as other pilots from different Site Directors may have already processed the jobs. This case happens when the number of waiting jobs is inferior to the number of free slots in the resources, which is rarely the case in production but can occur in specific Sites. Thus, one could imagine a Site Director strategy consisting in continuously sending pilots in the queues, which would slow down production rates in the case that pilots do not fetch any job. The challenge in this approach lies in the various scheduling policies of the Sites: while some Sites can prioritize VOs submitting the most pilots, others can favor pilots that effectively run for a long time.

We have explained in Section 3.3 that getting an accurate value of the time left allocated to a pilot is a complex operation due to the various batch systems composing the grids: different types, versions and configurations. In combination with the time left value, pilots need to get the "power" of the CPU, namely how efficient is a CPU to run an application of interest. Indeed, two different processors will likely not spend the same time running the same application. Solutions such as the DIRAC Benchmark [38,39] have been developed to provide an estimation of the CPU power for Monte-Carlo simulations in the LHC context. Valassi et al. have also started to work on a new benchmarking solution to deal with various computing resources and to provide accurate values [40]. Designing efficient ways of getting accurate CPU power and time left values would help better exploiting the allocations by fetching the most adapted jobs.

Finally, we encourage VOs that would like to conduct similar research with other systems to record and collect data about pilots, jobs and also about configuration changes in the Sites in order to get a clear overview of the impact of the external factors. The task remains challenging because (i) information from BDII is not always reliable and (ii) the grid architecture involves many operations that are hard to follow: there are external and internal issues, maintenances, upgrades involving various actors every day.

7. Conclusion

This work primarily supports research efforts conducted by the LHC experiments – and especially LHCb –, which mainly run Monte Carlo simulation workloads to replicate experimental conditions and performance of the detectors. In the context of the constant improvement of the LHC and the arrival of the High-Luminosity LHC, such an approach becomes critical in order to increase the quality of the analysis of the acquired data. More generally, this paper should assist any community working with distributed and shared computing resources – even aggregated into High-Performance Computers or clouds – in processing a growing amount of data.

Through this paper, we have demonstrated the importance of continuously improving Pilot-Job provisioning mechanisms to better exploit shared and distributed heterogeneous computing resources. After exposing the advantages and limitations of the Pilot-Job paradigm (Section 2), we explored one of the main Pilot-Job provisioning tools: the DIRAC Site Director in the context of the LHCb experiment (Section 3). For 12 months, we analyzed 13 Site Directors managing 65 grid sites on WLCG, dealing with 57,000 LHCb jobs simultaneously. By introducing multi-threading within the Site Directors and including CE-specific performance improvements, we speeded up the monitoring mechanism: the duration of the activity dropped down (−22%). Additionally, we better shared the workloads between the cycles of the Site Directors to generate a fewer number of pilots more frequently (Section 4).

We conducted performance studies, repeated multiple times over one year, to prove the efficiency of every change made [34]. We measured an overall gain of 18.41% of the number of pilots successfully submitted per hour, which represents 728 additional pilots per hour. We also recorded an increase of 40.86% of the number of jobs processed simultaneously per second, which means that WLCG is simultaneously in charge of 80,300 LHCb jobs (Section 5). Thus, this study enables the generation of more pilots to meet the increasing demand for computing power. In this context, computing power is essential to refine the analysis and increase the statistics and the confidence that we can place in the discoveries made thanks to the LHC, which will affect our understanding of the universe.

Future studies should focus on further increasing the Pilot-Job submission frequency (Section 6). Automatically fine-tuning the parameters of the Site Directors – the number and the nature of the queues that they supervise – depending on the load on the Sites would be a solution. A complementary solution would consists in adapting the submission rate according to the scheduling policies of the Sites to optimize the priority of the Pilot-Jobs within the queues. Working on a CPU benchmarking solution providing accurate CPU "power" estimations for various processors would constitute another approach to maximize the use of the allocated resources.

CRediT authorship contribution statement

Alexandre F. Boyer: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft. **Christophe Haen:** Conceptualization, Supervision, Validation. **Federico Stagni:** Project administration, Resources. **David R.C. Hill:** Supervision, Writing – review & editing, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We would like to thank Vladimir Romanovskiy for his precious advice about the Site Director mechanisms.

References

[1] CERN, LHC season 2 facts & figures, 2018, URL: https://home.cern/sites/home.web.cern.ch/files/2018-07/factsandfigures-en_0.pdf, (accessed 29 June 2021).

[2] M. Clemencic, G. Corti, S. Easo, C.R. Jones, S. Miglioranzi, M. Pappagallo, P.R. and, The LHCb simulation application, Gauss: Design, evolution and experience, J. Phys. Conf. Ser. 331 (3) (2011) 032023, http://dx.doi.org/10.1088/1742-6596/331/3/032023.

[3] CERN, Worldwide LHC computing grid, 2021, URL: https://wlcg.web.cern.ch/, (accessed 11 February 2021).

[4] M. Turilli, M. Santcroos, S. Jha, A comprehensive perspective on pilot-job systems, ACM Comput. Surv. 51 (2) (2018) 43:1–43:32, http://dx.doi.org/10.1145/3177851.

[5] F. Stagni, A. McNab, C. Luzzi, W. Krzemien, D. Consortium, DIRAC universal pilots, J. Phys. Conf. Ser. 898 (9) (2017) 092024, http://dx.doi.org/10.1088/1742-6596/898/9/092024.
[6] R. Grzymkowski, T.H. and, Belle II public and private cloud management in VMDIRAC system, J. Phys. Conf. Ser. 664 (2) (2015) 022021, http://dx.doi.org/10.1088/1742-6596/664/2/022021.
[7] F. Stagni, A. Valassi, V. Romanovskiy, Integrating LHCb workflows on HPC resources: status and strategies, EPJ Web Conf. 245 (2020) 09002, http://dx.doi.org/10.1051/epjconf/202024509002.
[8] W. Wu, T. Hara, H. Miyake, I. Ueda, W. Kan, P. Urquijo, BelleII@home: Integrate volunteer computing resources into DIRAC in a secure way, J. Phys. Conf. Ser. 898 (2017) 102003, http://dx.doi.org/10.1088/1742-6596/898/10/102003.
[9] A. McNab, F. Stagni, M.U. Garcia, Running jobs in the vacuum, J. Phys. Conf. Ser. 513 (3) (2014) 032065, http://dx.doi.org/10.1088/1742-6596/513/3/032065.
[10] H. Miyake, R. Grzymkowski, R. Ludacka, M. Schram, Belle II production system, J. Phys. Conf. Ser. 664 (2015) 052028, http://dx.doi.org/10.1088/1742-6596/664/5/052028.
[11] L. Arrabito, C. Barbier, R.G. Diaz, B. Khélifi, N. Komin, G. Lamanna, C. Lavalley, T.L. Flour, J. Lenain, A. Lorca, M. Renaud, M. Sterzel, T. Szepieniec, G. Vasileiadis, C. Vuerli, Application of the DIRAC framework to CTA: first evaluation, J. Phys. Conf. Ser. 396 (3) (2012) 032007, http://dx.doi.org/10.1088/1742-6596/396/3/032007.
[12] EGI, Workload manager, 2019, URL: https://wiki.egi.eu/wiki/Workload_Manager, (accessed 11 February 2021).
[13] FranceGrilles, FG-DIRAC, 2021, URL: http://www.france-grilles.fr/catalogue-de-services/fg-dirac/, (accessed 29 June 2021).
[14] M. Ellert, M. Grønager, A. Konstantinov, B. Kónya, J. Lindemann, I. Livenson, J.L. Nielsen, M. Niinimäki, O. Smirnova, A. Wäänänen, Advanced resource connector middleware for lightweight computational grids, Future Gener. Comput. Syst. 23 (2) (2007) 219–240, http://dx.doi.org/10.1016/j.future.2006.05.008.
[15] P. Andreetto, S. Bertocco, F. Capannini, M. Cecchi, A. Dorigo, E. Frizziero, A. Gianelle, F. Giacomini, M. Mezzadri, S. Monforte, et al., Status and developments of the CREAM computing element service, J. Phys. Conf. Ser. 331 (6) (2011) 062024, http://dx.doi.org/10.1088/1742-6596/331/6/062024.
[16] D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: the Condor experience, Concurr. Comput.: Pract. Exper. 17 (2–4) (2005) 323–356, http://dx.doi.org/10.1002/cpe.938, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.938.
[17] A.B. Yoo, M.A. Jette, M. Grondona, SLURM: Simple linux utility for resource management, in: D. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 44–60, http://dx.doi.org/10.1007/10968987_3.
[18] IBM Knowledge Center, IBM platform LSF, 2016, URL: https://www.ibm.com/products/hpc-workload-management, (accessed 11 February 2021).
[19] F. Stagni, A. Tsaregorodtsev, A. McNab, C. Luzzi, Pilots 2.0: DIRAC pilots for all the skies, J. Phys. Conf. Ser. 664 (6) (2015) 062061, http://dx.doi.org/10.1088/1742-6596/664/6/062061.
[20] A. Casajus, R. Graciani, S. Paterson, A. Tsaregorodtsev, t.L.D. Team, DIRAC pilot framework and the DIRAC Workload Management System, J. Phys. Conf. Ser. 219 (6) (2010) 062049, http://dx.doi.org/10.1088/1742-6596/219/6/062049.
[21] A. Bricker, M. Litzkow, M. Livny, Condor Technical Summary, Technical Report, University of Wisconsin-Madison Department of Computer Sciences, 1992.
[22] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor-G: A computation management agent for multi-institutional grids, Cluster Comput. 5 (3) (2002) 237–246, http://dx.doi.org/10.1023/A:1015617019423.
[23] I. Sfiligoi, gideinWMS—a generic pilot-based workload management system, J. Phys. Conf. Ser. 119 (6) (2008) 062044, http://dx.doi.org/10.1088/1742-6596/119/6/062044.
[24] P. Nilsson, The PanDA system in the ATLAS experiment, in: Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research — PoS(ACAT08), Vol. 70, SISSA Medialab, 2009, p. 027, http://dx.doi.org/10.22323/1.070.0027, URL: https://pos.sissa.it/070/027/.
[25] S. Bagnasco, L. Betev, P. Buncic, F. Carminati, F. Furano, A. Grigoras, C. Grigoras, P.M. Lorenzo, A.J. Peters, P. Saiz, The ALICE Workload Management System: Status before the real data taking, J. Phys. Conf. Ser. 219 (6) (2010) 062004, http://dx.doi.org/10.1088/1742-6596/219/6/062004.
[26] M. Hategan, J. Wozniak, K. Maheshwari, Coasters: Uniform resource provisioning and access for clouds and grids, in: 2011 Fourth IEEE International Conference on Utility and Cloud Computing, 2011, pp. 114–121, http://dx.doi.org/10.1109/UCC.2011.25.
[27] A.J. Rubio-Montero, E. Huedo, F. Castejón, R. Mayo-García, GWpilot: Enabling multi-level scheduling in distributed infrastructures with GridWay and pilot jobs, Future Gener. Comput. Syst. 45 (2015) 25–52, http://dx.doi.org/10.1016/j.future.2014.10.003.
[28] D. Oleynik, S. Panitkin, M. Turilli, A. Angius, S. Oral, K. De, A. Klimentov, J.C. Wells, S. Jha, High-throughput computing on high-performance platforms: A case study, in: 2017 IEEE 13th International Conference on E-Science (E-Science), 2017, pp. 295–304, http://dx.doi.org/10.1109/eScience.2017.43.
[29] S. Bagnasco, L. Betev, P. Buncic, F. Carminati, C. Cirstoiu, C. Grigoras, A. Hayrapetyan, A. Harutyunyan, A.J. Peters, P. Saiz, AliEn: ALICE environment on the GRID, J. Phys. Conf. Ser. 119 (6) (2008) 062012, http://dx.doi.org/10.1088/1742-6596/119/6/062012.
[30] A.F. Boyer, D.R. Hill, C. Haen, F. Stagni, Pilot-job provisioning through cream computing elements on the worldwide LHC computing grid, in: 34th European Simulation and Modelling Conference (ESM), Vol. 34, Eurosis, Toulouse, France, 2020, pp. 33–38.
[31] A. Tsaregorodtsev, DIRAC distributed computing services, J. Phys. Conf. Ser. 513 (3) (2014) 032096, http://dx.doi.org/10.1088/1742-6596/513/3/032096.
[32] DIRACGrid, Source code of the DIRAC project, 2021, URL: https://github.com/DIRACGrid/DIRAC, (accessed 11 February 2021).
[33] A. Osman, A. Anjum, N. Batool, R. McClatchey, A fault tolerant, dynamic and low latency BDII architecture for grids, 2012, arXiv:1202.5512 [cs]. URL: http://arxiv.org/abs/1202.5512, arXiv:1202.5512.
[34] A.F. Boyer, DIRAC Site Director: Analysis and Performance Evaluation, Mendeley Data, V1, 2021, http://dx.doi.org/10.17632/6r388827fz.1.
[35] Python Software Foundation, GlobalInterpreterLock, 2020, URL: https://wiki.python.org/moin/GlobalInterpreterLock, (accessed 11 February 2021).
[36] B. David, Inside the python GIL, 2009, URL: http://www.dabeaz.com/python/GIL.pdf, (accessed 11 February 2021).
[37] Nordugrid, ARC job supervisor, 2014, URL: http://www.nordugrid.org/documents/code/sdk/classArc_1_1JobSupervisor.html, (accessed 11 February 2021).
[38] DiracGrid, Source code of the Dirac Benchmark 12, 2017, URL: https://github.com/DIRACGrid/DB12, accessed 5 July 2021.
[39] P. Charpentier, Benchmarking worker nodes using LHCb productions and comparing with HEPSpec06, J. Phys. Conf. Ser. 898 (2017) 082011, http://dx.doi.org/10.1088/1742-6596/898/8/082011.
[40] A. Valassi, M. Alef, J.-M. Barbet, O. Datskova, R. De Maria, M. Fontes Medeiros, D. Giordano, C. Grigoras, C. Hollowell, M. Javurkova, V. Khristenko, D. Lange, M. Michelotto, L. Rinaldi, A. Sciabà, C. Van Der Laan, Using HEP experiment workflows for the benchmarking and accounting of WLCG computing resources, EPJ Web Conf. 245 (2020) 07035, http://dx.doi.org/10.1051/epjconf/202024507035.

Alexandre Boyer:

Alexandre Boyer is currently a Ph.D. student at the Clermont Auvergne University, France. As part of his Ph.D, he contributes to the offline activities of the LHCb experiment, hosted at CERN, Switzerland. His research focuses on approaches to efficiently integrate High-Throughput Computing workflows on heterogeneous – grid and supercomputer – computing resources. Previously, he worked at NIST, USA as a Guest Researcher. He was mainly involved in the design, development and maintenance of machine learning evaluation infrastructures. His research interests lie in the distributed computing area including cloud, grid and high-performance computing. They imply performance evaluation, development and data analysis.

Christophe Haen (Ph. D.):

Christophe Haen works at CERN for the LHC Beauty experiment. After graduating as a computer scientist from ISIMA (2010), he joined the LHCb Online team as a Ph.D. student with the Blaise Pascal University (Clermont Ferrand, France). His work focused on reinforcement learning applied to system administration. Upon completion of his Ph.D. in 2013, Christophe joined the LHCb Offline team, mainly as the responsible for the large scale data management and developer of LHCbDIRAC and DIRAC software. He is also in charge of the DevOps aspects of the LHCb distributed computing.

Federico Stagni (Ph.D.):

Born in 1980 in Ferrara, Italy, Federico attended the University of Ferrara, where he graduated cum laude in 2005. In 2009 he completed a Ph.D. in computing at the same university. In 2009 he started working at CERN, first as a fellow and then as a CERN staff, covering a few roles in the computing for the LHCb experiment. Since 2012 he is LHCb and DIRAC developers coordinator.

David R.C. HILL (Ph.D.):

Professor David Hill is Deputy Director ISIMA Computer Science & Modeling Institute (French "Grande Ecole d'Ingénieur"). He was Vice President of Blaise Pascal University (2008–2012) and also past director of a French Regional Computing Center (CRRI) (2008–2010). Professor Hill is doing his research at the French CNRS. With Vincent Breton, he is the scientific head of AuverGrid the Regional Computing Mesocenter. Professor Hill has authored or co-authored more than 250 papers (including more than 60 papers in indexed referred journals) and he has also published several text books. For his research interests and teaching see his Web page: www.isima.fr/~hill.