LHCbDirac: distributed computing in LHCb

F Stagni, P Charpentier, R Graciani, A Tsaregorodtsev, J Closier, Z Mathe, M Ubeda, A Zhelezov, E Lanciotti, V Romanovskiy, K D Ciba, A Casajus, S Roiser, M Sapunov, D Remenska, V Bernardoff, R Santana, R Nandakumar

1PH Department, CH-1211 Geneva 23 Switzerland
2IT Department, CH-1211 Geneva 23 Switzerland
3C.P.P.M - 163, avenue de Luminy - Case 902 - 13288 Marseille cedex 09
4Universitat de Barcelona, Gran Via de les Corts Catalanes, 585 Barcelona
5STFC Rutherford Appleton Laboratory, Harwell, Oxford, Didcot OX11 0QX
6Ruprecht-Karls-Universitt Heidelberg, Grabengasse 1, D-69117 Heidelberg
7NIKHEF, Science Park 105, 1098 XG Amsterdam
8CBPF, Rua Sigaud 150, Rio De Janeiro - RJ
9IHEP, Protvino

E-mail: federico.stagni@cern.ch

Abstract. We present LHCbDirac, an extension of the DIRAC community Grid solution that handles LHCb specificities. The DIRAC software has been developed for many years within LHCb only. Nowadays it is a generic software, used by many scientific communities worldwide. Each community wanting to take advantage of DIRAC has to develop an extension, containing all the necessary code for handling their specific cases. LHCbDirac is an actively developed extension, implementing the LHCb computing model and workflows handling all the distributed computing activities of LHCb. Such activities include real data processing (reconstruction, stripping and streaming), Monte-Carlo simulation and data replication. Other activities are groups and user analysis, data management, resources management and monitoring, data provenance, accounting for user and production jobs. LHCbDirac also provides extensions of the DIRAC interfaces, including a secure web client, python APIs and CLIs. Before putting in production a new release, a number of certification tests are run in a dedicated setup. This contribution highlights the versatility of the system, also presenting the experience with real data processing, data and resources management, monitoring for activities and resources.

1. Introduction

DIRAC[1][2] is a community Grid solution. Developed in python, it offers powerful job submission functionalities, and a developer-friendly way to create services and agents. Services expose an extended Remote Process Call (RPC) and Data Transfer (DT) implementation; agents are a stateless light-weight component, comparable to a cron-job. Being a community Grid solution, DIRAC can interface with many resource types, and with many providers. Resource types are, for example, a computing element (CE), or a catalog. DIRAC provides a layer for interfacing with many CE types, and different catalog types. It also gives the opportunity to instantiate DIRAC types of sites, CEs, storage elements (SE) or catalogs. It is possible to access resources at certain type of clouds [3]. Submission to resources provided via volunteer computing is also in the pipeline.
DIRAC has been initially developed inside the LHCb\[^4\][^5\] collaboration, as a LHCb-specific project, but many efforts have been made to re-engineering it into a generic framework, capable to serve the distributed computing needs of different Virtual Organizations (VOs). After this complex reorganization, which was fully finalized in 2010, the LHCb-specific code resides in the LHCbDirac extension while DIRAC is VO-agnostic. In this way, other VOs, like Belle II [^6], or ILC/LCD [^7] have developed their custom extensions to DIRAC.

DIRAC is a collection of sub-systems, each constituted of services, agents, and a database backend. Sub-systems are, for example, the Workload Management System (WMS) or the Data Management System (DMS). Each system comprises a generic part, which can be extended in a VO-specific part. DIRAC provides also a highly-integrated web portal. Many DIRAC services and agents can be run on their own, without the need for an extension. Each DIRAC installation can decide to use only those agents and services that are necessary to cover the use case of the communities it serves to, and this is true also for LHCb. There are anyway many concepts that are specific to LHCb, that can not reside in the VO-agnostic DIRAC. LHCbDirac knows, for example, some concepts regarding the organization of the physics data. And has to know how to interact with the LHCb applications that are executed on the worker nodes (WN). LHCbDirac handles all the distributed computing activities of LHCb. Such activities include real data processing (e. g. reconstruction, stripping and streaming), Monte-Carlo simulation and data replication. Other activities are groups and user analysis, data management, resources management and monitoring, data provenance, accounting for user and production jobs, etc.

LHCbDirac is actively developed by few full time “programmers”, and some contributors. For historical reasons, there is a large overlap between DIRAC and LHCbDirac developers. The LHCb extensions of DIRAC also includes extensions to some of the web portal pages, and new LHCb specific pages.

While DIRAC and its extensions follow independent release cycles, LHCbDirac is built on top of an existing DIRAC release. This means that the making of a LHCbDirac release has to be carefully programmed, considering also the release cycle of DIRAC. In order to lower the risks of introducing potentially dangerous bugs in the production setup, the team has introduced a lengthy certification process that is applied to each of the release candidates.

This paper explains how LHCbDirac is developed and used, giving an architectural and an operational perspective. It will also show experience from 2011 and 2012 Grid processing.

This paper is organized as follows: section 2 gives a quick overview of the LHCb distributed activities. Section 3 constitutes the core of this report, giving an overview of how LHCb extended DIRAC. Section 4 explains the adopted development process. Section 5 shows some results from past year’s run, and final remarks are given in section 6.

2. LHCb distributed computing activities

The LHCb distributed computing activities are based on the experiment Computing Model [^8]. Even if the computing model has been recently subject to some relaxation, in its fundamentals it is still in use. We make a split between data processing, handled in the following section, and data management, explained in section 2.2. LHCbDirac has been developed for handling all these activities.

2.1. Data processing

We split data processing into production and non-production activities. Production activities are handled centrally, and include:

- **Prompt Data Reconstruction, and Data Reprocessing**: data coming directly from the LHC machine is distributed, as RAW files, to the six Tier 1 centers participating to the LHCb Grid, and reconstructed. The distribution of the RAW files and their
subsequent reconstruction is driven by shares, agreed between the experiment and the resource providers. A second reconstruction of already reconstructed RAW files is called data reprocessing.

- **Data (re-)**Stripping: stripping is the name LHCb gives to the selection of interesting events. Data Stripping comes just after Data Reconstruction or Data Reprocessing.
- **Data Swimming**: a data driven acceptance correction algorithm, used for physics analysis.
- **Calibration**: a lightweight activity that follows the same workflow as Data Stripping, thus starts from already reconstructed data.
- **Monte Carlo (MC) simulations**: the simulation of real collisions started long before real data recording, and it is still a dominant activity for what regards purely the number of jobs handled by the system. Even if simulated events are continuously produced, large MC campaigns are commonly scheduled during longer shut-down periods, when LHCb is not recording collisions.
- **Working groups analysis**: very specific, usually short-lived productions recently introduced in the system, with the goal of providing working groups with shared data and information.

Some of these production activities also include a merging phase: for example, streamed data that is produced by stripping jobs is usually stored into small files. Thus, it is convenient to merge them into larger files that are then stored in the Grid Storage Elements. Non-production activities include user data analysis jobs, which represent a good fraction of the total amount of jobs handled by the system. User jobs content can vary, so maximum flexibility is required.

### 2.2. Data management

Data Management activities can be a direct consequence of the production activities of Data Processing, or their trigger. Data management activities include the data-driven replication on the available disks, the archival on a tape system, their removal for freeing disk space for newer data or because their popularity decreased, or checks, that include consistency, for example between the file catalogs in use, or with the real content of the storage elements.

### 2.3. Testing

Continuous testing is done using test jobs. Some of these test jobs are sent by the central Grid middleware group, and are VO-agnostic. LHCb complements the tests with VO-specific ones. Other kind of tests include, for example, tests on the available disk space, or on the status of the distributed resources of LHCb.

### 3. LHCbDirac: a DIRAC extension

LHCbDirac extends the functionalities of many DIRAC components, while introducing few LHCb-only ones. Throughout this section, we try to summarize all of them.

#### 3.1. Interfacing

DIRAC provides interfaces to expose, in a simplified way, most of its functionalities, in the form of APIs, scripts and via a web portal. Through the APIs, users can, among other things, create, submit and control jobs, together with managing the interaction with grid resources. LHCbDirac extends these interfaces for the creation and management of LHCb workflows. It also provides interfaces for creating and submitting productions. These APIs can be used directly by the users, while external tools like Ganga[9], which is the tool of choice for the submission of user jobs, masquerade their usage providing an easy to use tool for configuring the LHCb specific analysis workflows. Each client installation also provides the operation team and the users with many additional scripts.
3.2. Interfacing with LHCb applications, handling results

One of the key goals of LHCbDirac is dealing with LHCb applications. More in details, it provides tools for the distribution of the LHCb software, its installation, and use. Using the LHCb software not only means preparing the environment and to run its applications on the worker nodes, but also dealing with its options, and handling its results. The LHCb core software team has provided an easy-to-use interface for setting run options for the LHCb applications, and LHCbDirac now actively use this interface; when a new production request (see section 3.4.4 for details) is created, the requestor should know which application options will be executed.

Handling the results of the LHCb applications means, for example, reporting their exit code, the number of events processed and/or registered in their output files, but also parsing of log files and application summary reports. LHCbDirac handles the registration and upload of the output data, and implements the naming conventions for output files as decided by the computing team.

3.3. Data management system

LHCbDirac extends the Data Management System of DIRAC for providing some specific features, like an agent that is in charge of merging histograms used to determine the quality of the LHCb data. It also provides agents and services to assess the storage used, for production and user files. Those users exceeding their storage quotas are notified, while special extensions can be granted. The LHCbDirac data management is also experimenting a new “popularity” service, whose aim is to assess which datasets are more popular between the analysts. The aim is to develop a system that, based on the popularity information that are collected, would automatically increase or reduce the number of disk replicas of the datasets.
3.4. The production system

The Production System constitutes a good part of LHCbDirac. It transforms physics requests into physics events. To do so, it implements the LHCb computing model. It is based on the DIRAC Transformation System, a key component for all the distributed computing activities of LHCb. The Transformation System is reliable, performant, and easy to extend. Together with the Bookkeeping System and the Requests Management system, it provides what is called the Production System. We introduce the concepts used in the system, and their implementations: workflows, the transformation system, the LHCb bookkeeping, and the production requests system.

3.4.1. Workflows

A workflow is, by definition, a sequence of connected steps. DIRAC provides an implementation of the workflow concepts as a core package, with the scope of running “complex” jobs. All production jobs, but also the user jobs and the test jobs, are described using DIRAC workflows. As can be seen in figure 2, each workflow is composed of steps, and each step includes a number of modules. These workflow modules are connected to python modules, that are executed in the specified order by the jobs. It is duty of a DIRAC extension to provide the code for the modules that are effectively run.

![Figure 2. Concepts of a workflow: each workflow is composed by steps, that include modules specifications.](image)

3.4.2. Transformation system

The Transformation System is used for handling “repetitive” work. It has two main uses: the first is for creating job productions, and the second for data management operations. When a new “Production Jobs” transformation is requested, a number of transformation tasks are dynamically created, based on the input files (if present), and using a “plugin”, that specifies the way the tasks are created, and can decide where the jobs will run, or how many input files can go into the same task. Each of the tasks pertaining to the same transformation will run the workflow specified when the transformation is created. The tasks are then submitted either to the Workload Management System, or to the Data Management System. The Transformation System can use external Metadata and Replica catalogs. For LHCb, the Metadata and Replica Catalogs are the LHCb Bookkeeping, presented in the next section, and the LFC catalog [10]. LHCb is now evaluating the DIRAC catalog as integrated metadata and replica file catalog solution. Figure 3 shows some components of the system implementation.

3.4.3. LHCb bookkeeping

The LHCb Bookkeeping System is the data recording and data provenance tool. The Bookkeeping allows to perform queries for datasets that are relevant for data processing and analysis. A query returns a list of files belonging to the dataset. It is the only tool that is 100% LHCb specific, i.e. there’s no reference to it in the core DIRAC. The DIRAC framework has been used to integrate the LHCb specific Oracle database designed for this purpose, which is hosted by the CERN Oracle service. The Bookkeeping is widely used inside the Production System, by a large fraction of the members of the Operations team, and by all the physicists doing data analysis within the collaboration. Figure 4 shows the main components within this system.
3.4.4. Production request system

The Production Request System [11] is a way to expose the users to the production system. It is an LHCbDirac development, and represents first of all a formalization of the processing activities. A production is an instantiation of a workflow. A production request can be created by any user, provided that there are formal steps definition in the steps database. Figure 5 shows this simple concept. Creating a step, a production request, and subsequently launching a production is done using a web interface, integrated in the LHCbDirac web portal, in a user-friendly way. The production request web page also offers a simple monitoring of the status of each request, like the percentage of processed events, which is publicly available. The production request system also represents an important part of the LHCb extension of the DIRAC web portal.

3.4.5. Managing the productions

The production system provides ways to partially automatize the productions management. For example, simulation productions extensions and stoppage are automatically triggered, based on the number of events already simulated with respect to those requested. For all productions types, some jobs are also automatically re-submitted, based on the nature of their failure. In figure 6 all the concepts discussed within this section are put together.

3.5. Resource Status System

The DIRAC Resource Status System (RSS) [12] is a monitoring and generic policy system that enforces managerial and operational actions automatically. LHCbDirac provides LHCb specific policies to be evaluated, and also uses the system for running agents that collects monitoring information that are useful for LHCb.
Figure 4. Schematic view of the LHCbDirac Bookkeeping System. The complexity of the system deserves an explanation per se. A number of layers are present, as shown in the legend in the bottom-right corner of the figure. Within this paper, we just observe that the data recorded represent the first source of information for production activities within LHCb.

Figure 5. Concepts of a request: each request is composed by a number of steps.

4. Development process
DIRAC and LHCbDirac follow independent release cycles. Their code is hosted on different repositories, and it is also managed using different version control systems. Every LHCbDirac version is developed on top of an existing DIRAC release, with which it is compatible, and does not go in production before a certification process is finalized. For example, there might be 2 or more LHCbDirac releases which depend on the same DIRAC release. Even if it is not always strictly necessary, every LHCbDirac developer should know which release of DIRAC her development is for. The major version of both DIRAC and LHCbDirac changes rarely, let’s say
Figure 6. Putting all the concepts together: from a Request to a Production job.

every 2 years. The minor version has, up to now, changed more frequently in LHCbDirac with respect to DIRAC, and sometimes LHCbDirac follows a strict release schedule.

Software testing is an important part of the LHCbDirac development process. A complete suite of tests comprises functional and non-functional tests. Non-functional tests like scalability, maintainability, usability, performance, and security are not easy to write and automatize. Unfortunately, performance, or scalability problems, are usually spot when a LHCbDirac version is already in production. Dynamic, functional tests are instead part of a certification process, which consists of a series of steps. The first entry point for a certification process comes from the recent adoption of a continuous integration tool for helping the code testing, its development, and to assess the code quality. This tool has already proven to be very useful, and we plan to expand its usage. Through it, we run unit tests, and small integration tests. Still, the bulk of the so-called certification process, that is done for each release candidate, becomes a series of system and acceptance tests. Running system and acceptance tests is still a manual, non-trivial process, and can also be a lengthy one.

5. The system at work in 2011 and 2012

Figure 7 shows four plots related the activities in the last year. These plots are shown within this paper as a proof of concept of some LHCb activities, and are not meant to represent a detailed report. For further information on the computing resource usage, the reader can found many more details in [13]. For what regards the plots shown here, what can be extrapolated is that:

- DIRAC installations can sustain an execution rate of 5 thousands jobs per hour, and 70 thousands a day, without showing sign of degradations.
- The CPU used for simulation jobs still represents the larger portion of the used resources.
- It is easy to compare the CPU efficiency of the LHCb applications running on the Grid, using a step accounting system.
- Real Data processings produce the majority of data stored.

6. Conclusions

DIRAC has proven to be a scalable, and extensible software. LHCb is, up to now, its main user, and its extension, LHCbDirac, is the most advanced one. An LHCbDirac installation,
intended as the combination of DIRAC and LHCbDirac code, is a full, all-in-one system, which started to be used for only simulation campaigns, and it is now used for all the distributed computing activities of LHCb. The most consistent extension is represented by the production system, which is vital for a large part of LHCb distributed activities. It is also used for datasets manipulations like data replication or data removal, and some testing activities. During the last years, we have seen how the number of jobs handled by the Production system steadily grew, and represents now more than half of the total number of jobs created. LHCbDirac provides extensions also for what regards its interfacing, with the system itself, and for the handling of some specific data management activities. The development and deployment processes have been adapted, over the years, and have now achieved a higher level of resilience. DIRAC, together with LHCbDirac, fully satisfies the LHCb needs of a tool for handling all its distributed computing activities.

References
[1] Tsaregorodtsev A, Bargiotti M, Brook N, Ramo A C, Castellani G, Charpentier P, Cioffi C, Closier J, Diaz R G, Kuznetsov G, Li Y Y, Nandakumar R, Paterson S, Santinelli R, Smith A C, Miguelez M S and Jimenez S G 2008 Journal of Physics: Conference Series 119 062048
[2] Tsaregorodtsev A, Brook N, Ramo A C, Charpentier P, Closier J, Cowan G, Diaz R G, Lanciotti E, Mathe Z, Nandakumar R, Paterson S, Romanovsky V, Santinelli R, Sapunov M, Smith A C, Miguelez M S and Zhelezov A 2010 Journal of Physics: Conference Series 219 062029

[3] Fifield T, Carmona A, Casajs A, Graciani R and Sevior M 2011 Journal of Physics: Conference Series 331 062009

[4] Alves A e a 2008 J. Instrum. 3 S08005 also published by CERN Geneva in 2010

[5] Antunes-Nobrega R e a 2005 LHCb computing: Technical Design Report Technical Design Report LHCb (Geneva: CERN) submitted on 11 May 2005

[6] Diaz R G, Ramo A C, Agero A C, Fifield T and Sevior M 2011 J. Grid Comput. 9 65–79

[7] Poss S 2012 Dirac filecatalog structure and content in preparation of the clic cdr LCD-OPEN 2012-009 CERN URL http://edms.cern.ch/document/1217804

[8] Brook N 2004 Lhcb computing model Tech. Rep. LHCb-2004-119. CERN-LHCb-2004-119 CERN Geneva

[9] Maier A, Brochu F, Cowan G, Egede U, Elmsheuser J, Gaidioz B, Harrison K, Lee H C, Liko D, Moscicki J, Muraru A, Pajchel K, Reece W, Samset B, Slater M, Soroko A, van der Ster D, Williams M and Tan C L 2010 J. Phys.: Conf. Ser. 219 072008

[10] Bonifazi F, Carbone A, Perez E D, D’Apice A, dell’Agnello L, Dillmann D, Girone M, Re G L, Martelli B, Peco G, Ricci P P, Sapunenko V, Vagnoni V and Vitracil D 2008 J. Phys.: Conf. Ser. 119 042005

[11] Tsaregorodtsev A and Zhelezov A 2009 CHEP 2009

[12] Stagni F, Santinelli R and Collaboration L 2011 Journal of Physics: Conference Series 331 072067

[13] Graciani Diaz R 2012 Lhcb computing resource usage in 2011 Tech. Rep. LHCb-PUB-2012-003. CERN-LHCb-PUB-2012-003 CERN Geneva