PROCEEDINGS OF SCIENCE

PoS(ISGC2015)012

Architecture of the LHCb Distributed Computing System

Federico STAGNI

E-mail: federico.stagni@cern.ch

Philippe CHARPENTIER*

E-mail: philippe.charpentier@cern.ch

Christophe HAEN

E-mail: christophe.haen@cern.ch

Zoltan MATHE

E-mail: zoltan.mathe@cern.ch

Cinzia LUZZI

E-mail: cinzia.luzzi@cern.ch

Joel CLOSIER

E-mail: joel.closier@cern.ch

PROCEEDINGS OF SCIENCE

The LHCb Distributed Computing system is based on the DIRAC interware, and includes many LHCb-specific extensions to form the BeautyDIRAC package. BeautyDIRAC is implementing the LHCb computing model. It handles workflows for all the distributed computing activities of LHCb. BeautyDIRAC provides extensions of the DIRAC components and interfaces, including a secure web client, python APIs and CLIs. The LHCb Production Management System (PMS) exposes to physics teams a powerful interface allowing the submission of complex requests for tasks involving a large number of jobs (productions). The PMS manages all types of LHCb production activities: simulation, reconstruction of real and simulated data, physics selection of events (stripping), working group analysis and event indexing. Automatized testing phases are implemented for simulation productions, as well as productions' validation and completion. Managers of simulation productions can therefore test, submit and verify a large number of production requests with a minimal effort. The testing phase is both a functional and a performance test, using a dedicated testing facility. Physics groups requesting productions can follow their progress through a user-friendly web interface. The PMS is built on top of the LHCb extensions of the Dirac Data Management (DMS) and Workload Management systems (WMS), which are highly integrated through the DIRAC components. The productionsâĂŹ tasks are handled by the BeautyDirac extension of the Dirac Transformation System (TS) from their creation to their completion. Tasks requiring input datasets are fully data-driven (using the DMS), as new files becoming part of a dataset can automatically generate the creation of new tasks. It is therefore quite easy to submit chains of productions, some of which consume as input dataset the output datasets of others. For simulation production chains, an agent is also in charge of creating new tasks until the required number of simulated events is available. The tasks are submitted as jobs to the WMS and eventually run on a large palette of resources (Grid sites, Cloud sites, HPC centers, computer clusters, volunteer computing platformsâĂę). User jobs are submitted through the same WMS as production jobs, which allows prioritizing jobs and running seamlessly user and production jobs within the same "pilot jobs", that constitutes a resource overlay in the DIRAC WMS. In this contribution we shall describe the synergy between the various BeautyDIRAC components (DMS, TS, WMS, PMS). We shall present in some details the LHCb Production Management System and show how it is used by a large community of physicists. We shall give examples on how very large productions can be centrally handled very efficiently by a small operations team. Finally, we shall give an overview of the LHCb Computing Operations' successes of the past few years.

International Symposium on Grids and Clouds 2015
15-20 March 2015
Academia Sinica, Taipei, Taiwan

*Speaker.

PoS(ISGC2015)012

1. Introduction

DIRAC[9][1] is a community Grid solution. Developed in python, it offers powerful job submission functionalities, and a developer-friendly way to create services, agents, and executors, together with the integration of external components.

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. Executors are designed around two components: the Mind knows how to retrieve, store and dispatch tasks, and Executors are the working processes that know what to do depending on the task type. Other types of components that are not developed within DIRAC can be included, and managed as well.

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), a catalog, or a storage. DIRAC provides a layer for interfacing with many CE types, and different catalog and storage types. It also gives the opportunity to instantiate DIRAC types of sites, CEs, storage elements (SE) or catalogs.

DIRAC has been initially developed inside the LHCb[7] collaboration, as a LHCb-specific project, but since 2010 the LHCb-specific code resides in the LHCbDirac extension while DIRAC is VO-agnostic. In this way, other VOs, like Belle II [4], or ILC/LCD [2] 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 components 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 lenghty certification process that is applied to each of the release candidates.

This paper is organized as follows: section 2 explains the architecture of some of the systems that form the LHCb production system, introducing the concepts used, together with a short description of their technical implementation. The section concludes explaining the production system at work. Within section 3, we explain a capability of our simulation jobs: their elasticity. Section 4 explains the management of productions, with a look at operational issues. Section 5 gives a brief explanation of pilots 2.0, a new generation of pilots recently introduced. Final remarks are given in section 6.

2. DIRAC and LHCbDIRAC systems and concepts

Within this section, we will explain briefly some of the concepts and systems of DIRAC and LHCbDIRAC. It is not in our interests to cover all the aspects of DIRAC: rather, we will concentrate on those tools, systems, and concepts that forms the backbone of what is used to run Production jobs for LHCb.

2.1 DIRAC job workflows

A workflow is, by definition, a sequence of connected steps. DIRAC provides an implementation of the workflow concepts in one of its *Core* packages, with the declared scope of running "complex" jobs, i.e. jobs who run one application after another, whose input/outputs are usually directly connected to each others. The implementation comes in the interchangeable formats of an XML file, an extended python dictionary, or Job Description Language (JDL), a know format for describing jobs running on distributed systems.

All production jobs, as well as user jobs and test jobs, are described using DIRAC workflows. As can be seen in figure 1, each worfklow 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. Parameters can be specified at any level: workflow, step, and even modules.

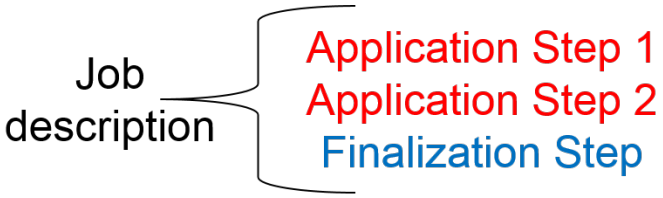

Figure 1: Concepts of a job Workflow: each workflow is composed by a number of steps.

2.2 Transformation System

The DIRAC 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 the "plugin" specified. A "plugin" 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 DIRAC workflow specified when the transformation is created. Figure 2 shows the basic concepts behind the implementation.

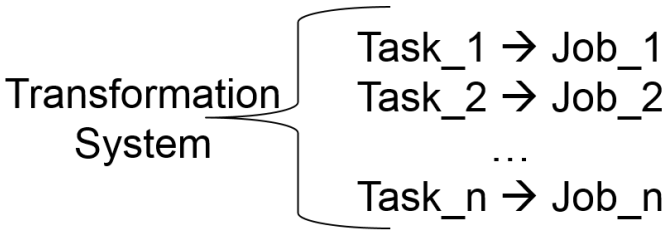

Figure 2: Concepts of the Transformation System: each Transformation has a number of similar tasks, which may become a Job when submitted to the WMS.

A task, when inside the transformation system, is not yet a Grid job nor a data management operation. For this last step, agents are defined, that upon inspections of the Transformation System tables, will submit the tasks either to the Workload Management System, or the Data Management System. The Transformation System, unless when being used just for simulation activities, can not live as a stand-alone system: whenever there are input data to be handled, external Metadata and Replica catalogs are needed. For LHCb, the Metadata and Replica Catalogs are the LHCb Bookkeeping [5], presented in the next section, and the DIRAC File catalog (DFC: [8]). Figure 3 shows some components of the Transformation System implementation.

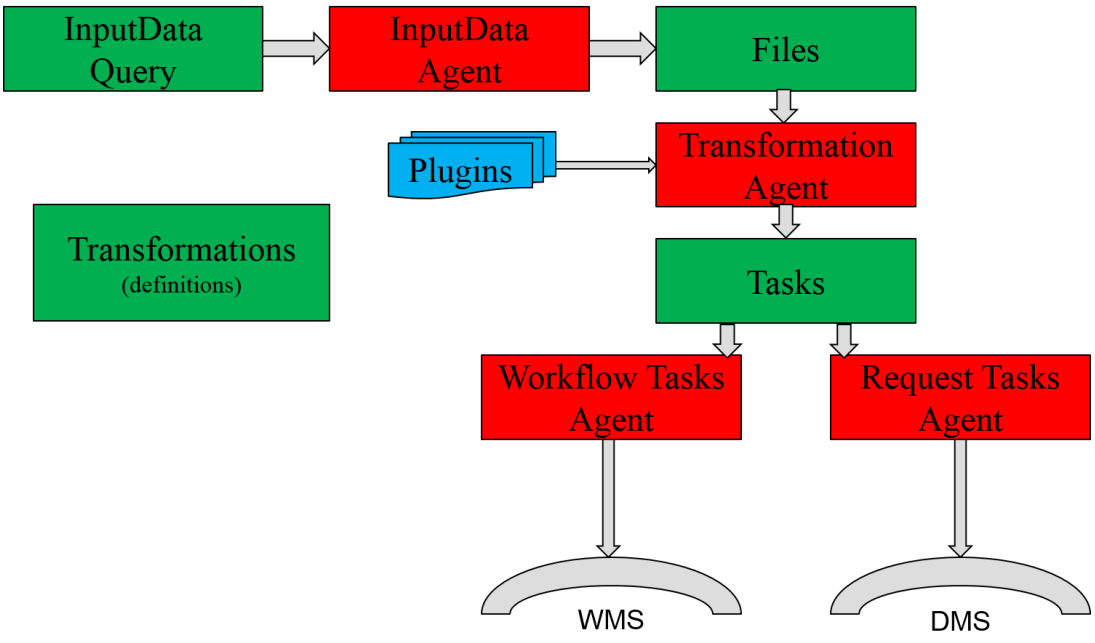

Figure 3: Schematized view of the design of the Transformation System.

2.3 The LHCb Bookkeeping System

The LHCb Bookkeping System is integrated within LHCbDIRAC, and it is the LHCb data

recording and data provenance tool. The Bookkeeping System 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. Being recognized as a key system by the collaboration, its design and implementation were subject to important modifications throughout the years, but has been stable enough for the last few years.

The bookkeeping is not necessarily a tool for doing distributed computing. Among its functions, users and machines are retrieving datasets for analysis and productions, together with metadata, like data taking and simulation conditions, or event types and file types. The bookkeeping is a read-only catalog for general users. Unlike other DIRAC systems, where MySQL is the prime choice as RDBMS, the CERN Oracle backend service has been chosen instead. Figure 4 shows the main components within this system.

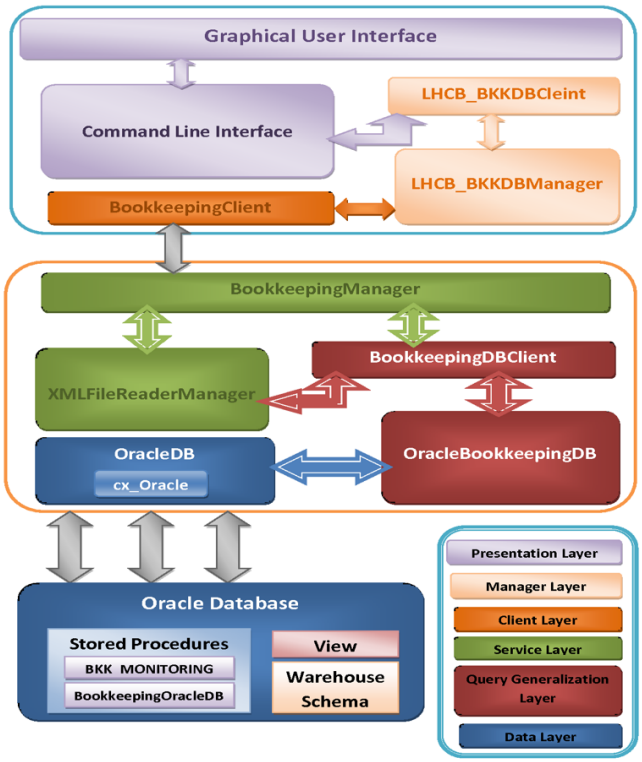

Figure 4: The LHCb Bookkeeping: a data recording and data provenance tool.

2.4 The Production Request System

The Production Request System is a way to expose users to the production system. It is an LHCbDIRAC developement, and represents first of all a formalization of the processing activities. A Production Request (PR) is a combination of steps, that can be created by any users, provided that there are formal steps definition in the steps database. In order to run, each PR has to be accepted by a member of the Physics Planning Group, and by one the production managers. If a PR is accepted is ready to be submitted to the LHCb grid, as one or more "Job Productions".

Steps inside a PR are then grouped and run by production jobs. Production managers can decide to group production steps the way they prefer. Figure 5 shows this simple concept. As per reminder, section 2.2 explained the Transformation System as a way to submit and manage Productions, and this is exactly what happens here: Production Requests are split in as many Transformations as the number of needed productions.

Figure 5: Concepts of a Production Request: each request is composed by a number of steps. Steps are grouped and run by productions.

Creating a step, a production request, and subsequently launching such production request are all operations 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: for example, the percentage of processed events, as requested, is reported and publicly available. Such information is also used, for simulation productions, to trigger their automatic completion. Figure 6 shows a screenshot of the Production Request and Production steps web pages.

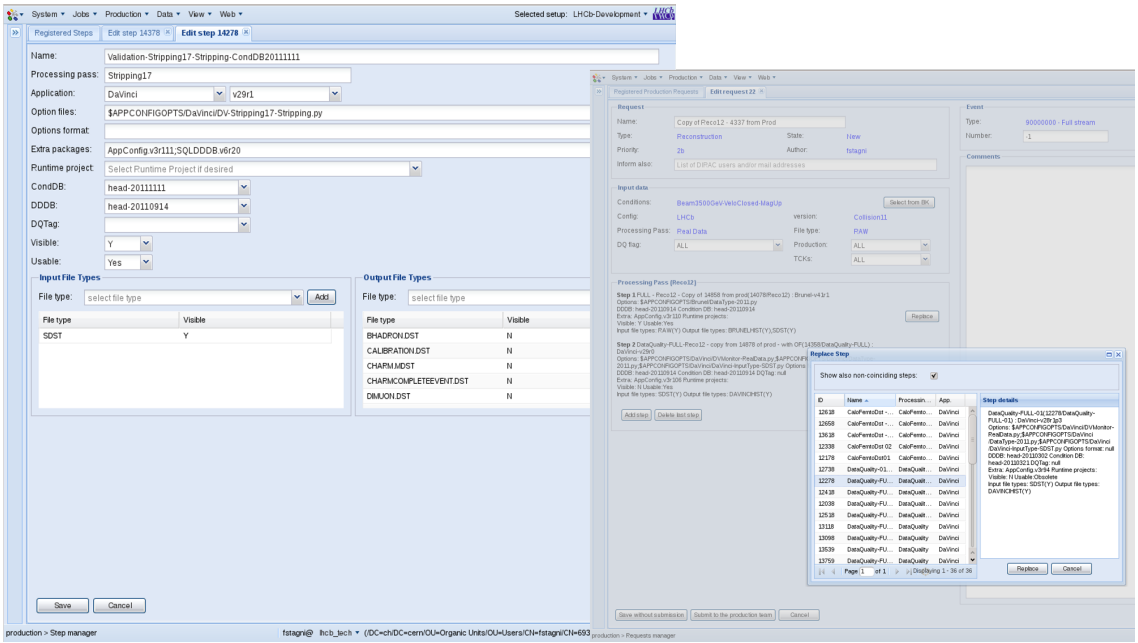

Figure 6: Web pages for creating, modifying, launching and monitoring a Production Request.

2.5 The LHCb Production System

Figure 7 shows all the concepts explained in the previous sections put together. One thing that should be noticed is how application steps, defined by users via the Production steps web page, are executed as Job workflow steps by each job produced by the Transformation System. The finalization steps assures, among performing several different operations, that output data is uploaded and registered.

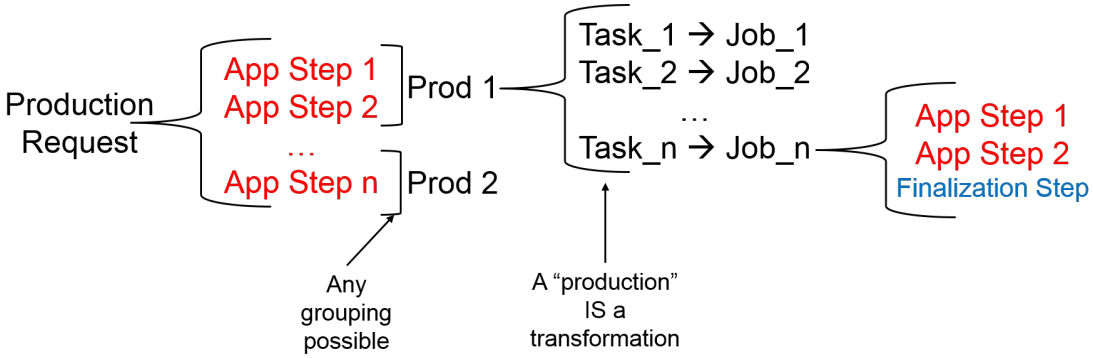

Figure 7: All concepts together.

3. Elastic Simulation jobs

When it comes to ease of management, simulation productions jobs have a decidive advantage over data processing jobs: being strictly CPU-bound, they can run on almost every type of resource. Figure 8 explains a simple concept of LHCb simulation jobs: its elasticity.

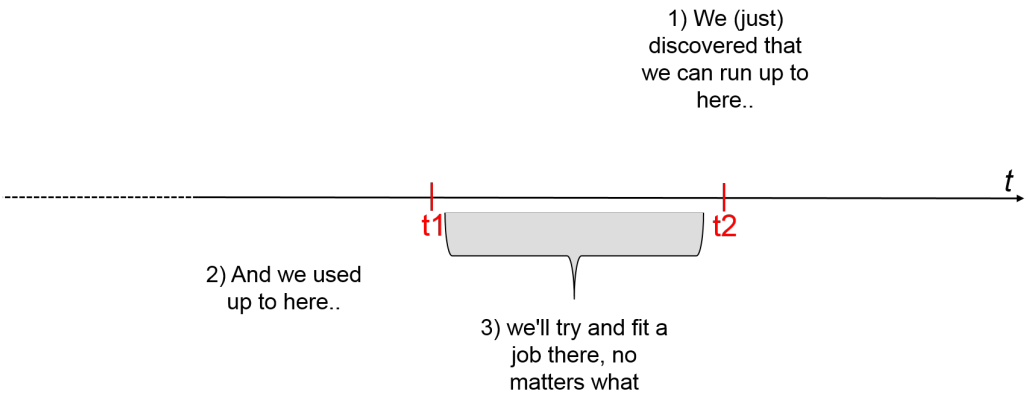

Figure 8: Elastic simulation jobs

Considering the fact that most of today's computing resources are limited in time (think about batch systems, for example) LHCb decided that it would have been of great help if short jobs would have been always present, for filling short queue slots with jobs. LHCb simulation jobs have added elasticity because the amount of events produced is determined only at run time, once the job

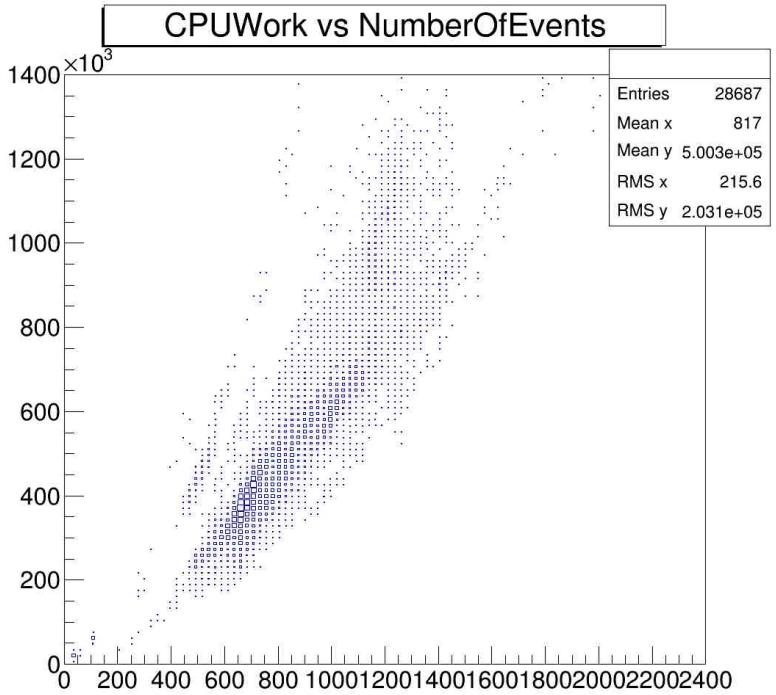

Figure 9: CPU work vs number of events produced by each simulation job, for a single production

already knows about the time left, the power of the machine, and the estimated amount of CPUtime necessary for simulating a single event.

Figure 9 shows a quick analysis of the jobs of one simulation production, consisting of about thirty thousands jobs. As can be seen, jobs were producing different number of events depending from the power of the machine and the CPU time left.

4. Productions Management

Doing productions management means managing a production from its start, to its end. It includes tesing, monitoring, consistency checks, and archival.

4.1 Automations specific of Simulation requests

Simulation Production Requests represent the largest fraction of Job Productions requests, with a ratio of about 10:1 with Data processing production requests. Simulation Production Requests are also, for several reasons, easier to handle with respect to data processing production requests, having no input data, a quicker lifecycle, and also requiring less strict consistency checks when the same productions end, due to the inner nature of simulated data. LHCb developed several automations for handling Simulation production requests.

For example, when a Simulation PR is started, its simulation production go through a testing phase before being fully submitted. Within this phase, a limited amount of jobs are created and submitted to a site chosen for testing, and each of the submitted job during this phase will produce

a fixed amount of events. Productions undergoing a testing phase are monitored by a dedicated agent, and when all the jobs are finished, an evaluation takes place: if all jobs failed, the PR is rejected. Instead, if jobs are successful, results are evaluated, and among few other checks, the CPUTime per event (CPUe) is calculated. It's at this moment that the Job description is modified: CPUe is added, destination changed, and so on.

Another type of automation happes while PRs are actively producing data: since each PR of type simulation needs to produce at least a requested amount of events, in case not enough events are produced, simulation productions are automatically extended.

4.2 Closing productions

There's a number of consistency checks that are common to all types of PRs. These checks happen when the production is at its the end, which means, for simulation productions, that enough events have been produced, and for other types of productions that all its input files have been processed. Figure 10 shows which types of consistency checks take place. Once successfully completed, productions can be be archived, so their jobs can be removed, as well as their transformation tasks, and logs can be archived.

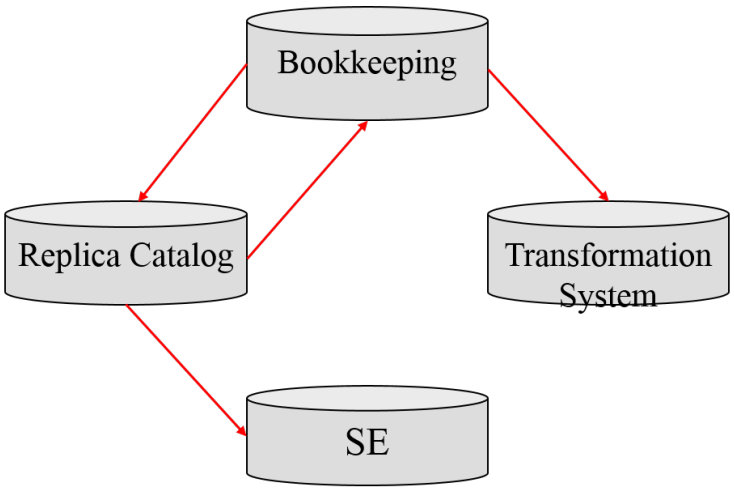

Figure 10: Consistency Checks

5. Pilots to fly in all the skyes

DIRAC has recently introduced a new generation of pilots, dubbed pilots 2.0, that is a complete rewriting from the previous implementation. Pilots 2.0 can be sent from agents, targeting Computing Elements and thus becoming "pilot jobs", or can be started by Worker Nodes, often in the form of Virtual Machines, that are exposed by IAAS (Infrastructure As A Service), or IAAC (Infrastructure As A Client).

Today's distributed computing has seen the emergence of IAAS in the form of cloud computing, and IAAC like VAC [6], or even mixed solutions, like those provided by volunteer computing projects like BOINC [3]. Experiments' distributed computing resources became more eterogeneous: in other words, the grid is not any more (only) The Grid. Community solutions like

(LHCb)DIRAC wants to hide such eterogeneity from the final users. There are two possible approaches: developing resource directors for each and every resource type, or creating a generic, configurable script, that can be sent, or fecthed, and run, by every resource type. We followed the second solution, leading to a new generation of pilots, the "pilots to fly in all the skyes". A pilot 2.0 can run on every computing resource, e.g.: on CREAM Computing elements, on DIRAC Computing elements, on Virtual Machines in the form of contextualization script, or IAAC (Infrastructure as a Client) provided that these machines are properly configured.

Pilots 2.0 are easy to configure and extend. A pilot has, at a minimum, to:

- install DIRAC (or its extension)
- configure DIRAC (or its extension)
- run the JobAgent

LHCb extended Pilots 2.0 changing the installation procedure of LHCbDIRAC: LHCbDIRAC is not fecthed from a remote web server, rather it's setup from CervVM-FS. Figure 11 shows simply how LHCbDIRAC generic pilots are running on each of its computing resource.

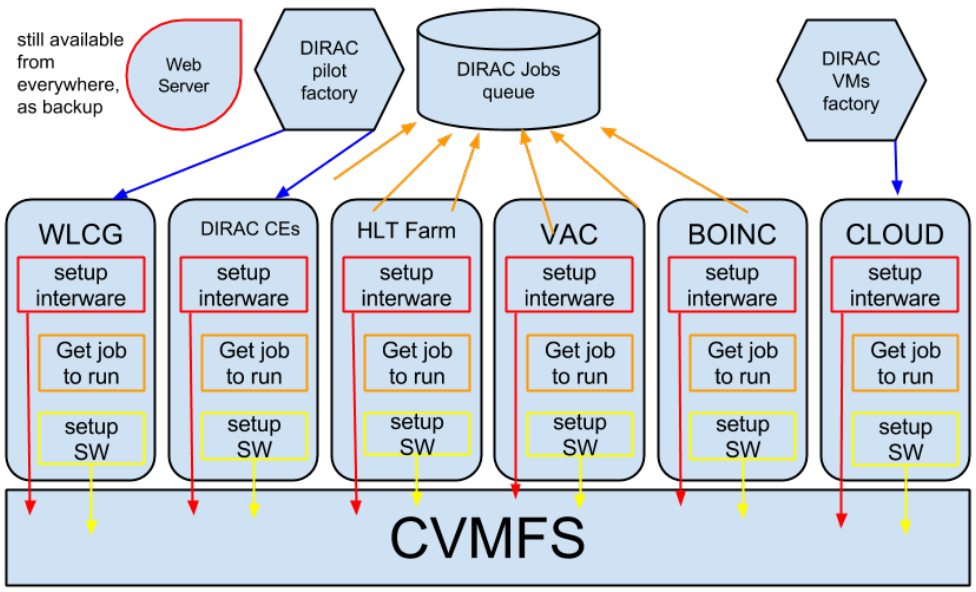

Figure 11: Pilots setup LHCbDIRAC from CVMFS. If they fail, they fall back downloading a tarball from a web server

6. Summary and conclusions

LHCbDIRAC is a DIRAC extension used for all the distributed computing activities of LHCb. DIRAC has proven to be a scalable, and extensible software. LHCb is, up to now, its main user and main contributor, and LHCb created the most advanced extension among DIRAC communities. 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 deployement processes have been adapted, over the years, and have now achieved a higher level of resiliance. The development cycle is such that LHCbDIRAC has about one major release every three or four year, while we create two or three minor releases per year; patch releases are as frequently as required (weekly, on average).

The current LHCbDIRAC installation is spread over thirty servers, with around fifty services and a hundred agents running. About twenty MySQL databases serves the production instances, with the obvious exception of the Bookkeeping database. LHCb is currently looking into introducing NoSQL databases and Queueing systems into the current production system.

DIRAC, together with LHCbDirac, fully satisfies the LHCb needs of a tool for handling all its distributed computing activities.

References

[1] A Casajus, K Ciba, V Fernandez, R Graciani, V Hamar, V Mendez, S Poss, M Sapunov, F Stagni, A Tsaregorodtsev, and M Ubeda. Status of the dirac project. *Journal of Physics: Conference Series*, 396(3):032107, 2012.

[2] C Grefe, S Poss, A Sailer, A Tsaregorodtsev, the Clic detector, and physics study. Ilcdirac, a dirac extension for the linear collider community. *Journal of Physics: Conference Series*, 513(3):032077, 2014.

[3] N HÃÿimyr, J Blomer, P Buncic, M Giovannozzi, A Gonzalez, A Harutyunyan, P L Jones, A Karneyeu, M A Marquina, E Mcintosh, B Segal, P Skands, F Grey, D LombraÃśa GonzÃąlez, and I Zacharov. Boinc service for volunteer cloud computing. *Journal of Physics: Conference Series*, 396(3):032057, 2012.

[4] T Kuhr and the Belle Ii Distributed Computing Group. First production with the belle ii distributed computing system. *Journal of Physics: Conference Series*, 513(3):032050, 2014.

[5] Zoltan Mathe and Philippe Charpentier. Optimising query execution time in lhcb bookkeeping system using partition pruning and partition-wise joins. *Journal of Physics: Conference Series*, 513(4):042032, 2014.

[6] A McNab, F Stagni, and M Ubeda Garcia. Running jobs in the vacuum. *Journal of Physics: Conference Series*, 513(3):032065, 2014.

[7] 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, and R Nandakumar. Lhcbdirac: distributed computing in lhcb. *Journal of Physics: Conference Series*, 396(3):032104, 2012.

[8] A Tsaregorodtsev and S Poss. Dirac file replica and metadata catalog. *Journal of Physics: Conference Series*, 396(3):032108, 2012.

[9] A Tsaregorodtsev and the Dirac Project. Dirac distributed computing services. *Journal of Physics: Conference Series*, 513(3):032096, 2014.