Status of the DIRAC Project

A. Casajus, K. Ciba, V. Fernandez, R. Graciani, V. Hamar, V. Mendez, S. Poss, M. Sapunov, F. Stagni, A. Tsaregorodtsev, M. Ubeda

1 Centre de Physique des Particules de Marseille, 163 Avenue de Luminy Case 902 13288 Marseille, France
2 University of Barcelona, Diagonal 647, 08028, Barcelona, Spain
3 CERN CH-1211 Geneva 23, Switzerland
4 Particle Physics Department, Campus Vida Universidade de Santiago de Compostela, 15782, Santiago de Compostela, Spain
5 Port d'Informacio Cientifica (PIC). Campus Universitat Autonoma de Barcelona, 08193, Bellaterra, Spain.

On behalf of the DIRAC Project

E-mail: atsareg@in2p3.fr

Abstract. The DIRAC Project was initiated to provide a data processing system for the LHCb Experiment at CERN. It provides all the necessary functionality and performance to satisfy the current and projected future requirements of the LHCb Computing Model. A considerable restructuring of the DIRAC software was undertaken in order to turn it into a general purpose framework for building distributed computing systems that can be used by various user communities in High Energy Physics and other scientific application domains. The CLIC and ILC-SID detector projects started to use DIRAC for their data production system. The Belle Collaboration at KEK, Japan, has adopted the Computing Model based on the DIRAC system for its second phase starting in 2015. The CTA Collaboration uses DIRAC for the data analysis tasks. A large number of other experiments are starting to use DIRAC or are evaluating this solution for their data processing tasks. DIRAC services are included as part of the production infrastructure of the GISELA Latin America grid. Similar services are provided for the users of the France-Grilles and IBERGrid National Grid Initiatives in France and Spain respectively. The new communities using DIRAC started to provide important contributions to its functionality. Among recent additions can be mentioned the support of the Amazon EC2 computing resources as well as other Cloud management systems; a versatile File Replica Catalog with File Metadata capabilities; support for running MPI jobs in the pilot based Workload Management System. Integration with existing application Web Portals, like WS-PGRADE, is demonstrated. In this paper we will describe the current status of the DIRAC Project, recent developments of its framework and functionality as well as the status of the rapidly evolving community of the DIRAC users.

1. Introduction
The DIRAC Project started in 2003 as a solution for the LHCb Collaboration at LHC, CERN, to manage its distributed computing resources for producing the modelling data for the LHCb experimental setup optimization. The successful application of DIRAC to the Monte-Carlo simulation
tasks made it possible to extend its functionality to also data management and user analysis activity. The main LHCb computing resources were (and still are) provided by the WLCG computing grid. The grid middleware, especially at the early stages of the grid computing development, was not up to the needs of the LHC experiments neither in terms of functionality nor stability. Also other non High Energy Physics communities started progressively to use more and more intensively the grid computing resources.

Grids are not the only way to provide computing resources to the user communities. There are still many sites (universities, research laboratories, etc), which hold computing clusters of considerable size, that are not part of any grid infrastructure. These resources are mostly used by local users and cannot be easily contributed to the pool of common resources of a wider user community even if the site belongs to its scientific domain. Installing the grid middleware to include such computing clusters to the grid infrastructure is prohibitively complicated especially if there are no local experts to do that. There are also emerging new sources of computing power, which are now commonly called computing clouds. Commercial companies provide most of these resources now but open source cloud solutions of production quality are also appearing.

The grid users are organized in communities with common scientific interests. These communities are also sharing common resources provided by the community members. Apparently, the contributed resources may come from any source listed above and a priori they are not uniform in their nature. Including all these resources in a coherent system seen by the users is still a challenging task. In addition, the common resources assume common policies of their usage. Formulating and imposing such policies while acknowledging the possible requirements of the resource providers (sites) is yet another challenging task.

The variety of requirements of different user communities is very large and it is difficult to meet everybody’s needs with just one set of the middleware components. Therefore, many communities, and most notably the LHC experiments, have started to develop their own sets of tools, which are evolving towards complete grid middleware solutions. Examples are numerous, ranging from subsystem solutions (PANDA workload management system [1] or PHEDEX data management system [2]) or close to complete Grid solutions (AliEn system [3]).

The DIRAC project provides now a complete solution for both workload and data management tasks in a distributed computing environment [4]. It provides also a software framework for building distributed computing systems. This allows easy extension of the already available functionality for the needs of a particular user community. The paradigm of the grid workload management with pilot jobs introduced by the DIRAC project brings an elegant and, at the same time, efficient solution to the computing resource heterogeneity problem outlined above. Although developed originally for the LHCb experiment, the DIRAC project is designed to be a generic system with LHCb specific features well isolated as an extension [5].

In this paper we describe the current status of the DIRAC project making emphasis on the recent developments and novelties in the usage of its software. We present the status of the DIRAC software framework in Section 2, followed by its application to management of the user workload in Sections 3. Managing various computing resources is discussed in Section 4. The DIRAC Data Management system is outlined in Section 5. The interfaces available to the DIRAC users are presented in Section 6. Sections 7 and 8 are devoted to examples of the DIRAC usage by various user communities and grid infrastructures. The prospects of the project evolution are presented in Section 9 followed by Conclusions.

2. DIRAC middleware framework

The DIRAC middleware components are all built in the same framework that provides solution for the common tasks and facilitates the work on various subsystems allowing developers to concentrate on specific features.

The DIRAC system is based on a Service Oriented Architecture (SOA). All the subsystems are built around the following components:
• **Services** – passive components listening to incoming queries by clients;
• **Databases** where **Services** are keeping their state;
• **Agents** – active components that are orchestrating the collaborative work of all the system;
• **Clients** – the software components used to access their corresponding **Services**.

All the DIRAC components are talking to each other with a custom secure DISET protocol [6]. This protocol is optimized for the efficiency of queries and is complying with the standard grid GSI security infrastructure. However, unlike common GSI solutions, the DISET implementation is relying directly on the OpenSSL libraries without the use of the Globus toolkit that makes it much lighter and less platform dependent.

The DIRAC Workload Management System (WMS) is using **Agents** very heavily. In particular, the user jobs getting into the WMS are being processed with a number of **Agents** called **Optimizers** in order to check the job definition validity and prepare them for the further execution. This feature revealed certain weaknesses during very periods of very intensive production activity of the LHCb experiment. The chains of **Optimizers** were not reactive enough to handle the high inflow of user jobs. The problem is due to the fact that the job status is kept in the database and each **Optimizer** in the chain has to look up for the job to handle independently. This solution did not scale especially for jobs with the input data where heavy checks of their availability and consistency are done. In order to cope with this problem a new type of DIRAC components was introduced – **Executors** [7]. Unlike **Agents** or **Optimizers**, **Executors** are capable of receiving messages sent to them by a **Dispatcher** service. The **Dispatcher** is organizing the common work of various **Executors** by sending them messages that contain the description of the jobs to be handled. These messages are pushing the **Executors** to carry out their function without the need for extra look-ups in the database. In addition, multiple **Executors** of the same kind can be started as necessary to cope with the level of the incoming job flow.

With the introduction of the **Executors**, handling of incoming jobs becomes event driven and, thus, the reactivity of the DIRAC WMS considerably increases readying it for even more intensive future data production runs. These new components will be used also for other highly intensive tasks like data replication.

### 3. Workload Management

The DIRAC WMS is based on the concept of pilot jobs that it introduced back in 2004 [8]. In the pilot scheduling paradigm (figure 1), the user tasks are submitted to the central Task Queue service. At the same time the pilot jobs are submitted to the computing resources by specialized components called Directors. Directors use the job scheduling mechanism suitable for their respective computing infrastructure: grid brokers, batch system schedulers, virtual machine dispatchers for clouds, etc. The pilot jobs start execution on the worker nodes, check the execution environment, collect the worker node characteristics and present them to the Matcher service. The Matcher service chooses the most appropriate user job waiting in the Task Queue and hands it over to the pilot for execution. Once the user task is executed and its outputs are delivered to the DIRAC central services, the pilot job can take another user task if the remaining time of the worker node reservation is sufficient.
The advantages of the pilot job concept are now well established. The pilots are not only increasing the visible efficiency of the user jobs but also help to manage the heterogeneous computing resources presenting them to the central services in a uniform coherent way. Large user communities can benefit also from the ability of applying the community policies that are not easy, if at all possible, with the standard grid middleware [8]. Furthermore executing several user tasks in the same pilot largely reduces the stress on the batch systems no matter if they are accessed directly or via grid mechanisms, especially if users subdivide their payload in many short tasks trying to reduce the response time.

3.1. Application of the site policies
During the user task scheduling, it is necessary to apply not only the community policies but also the policies imposed by the site usage. Some sites want to participate in just one type of the activity, for example, just Monte-Carlo simulation, or they want to limit the user analysis in favour of data production procedures. Another important point is that sites can have limitations in their hardware configuration that require limiting the number of jobs of certain types to run concurrently. For example, limitations in the disk I/O throughput should be taken into account when scheduling jobs accessing input data. In order to cope with these requirements, the DIRAC job scheduling was complemented by the possibility to define these site rules. This is done in a form of a number of limits that can be defined on per site basis. Limits can be set on the numbers of concurrently running jobs with different properties defined in the job descriptions: the user name and the group; job type, etc. These limits are taken into account at the moment when the pilots are querying the Matcher service (figure 1). If the number of already running jobs at the site is exceeding the defined limit, the pilot is be given a job of another type with possibly lower priority. If there are no jobs of allowed type at the moment of matching, the pilot fails to get any job and frees the computing slot.

The same mechanism can be also applied to define a small delay in the matching of a job of a given site to forbid matching of a similar job during a certain time after the previous job was scheduled. This allows to derandomize the burst of jobs of a certain type starting simultaneously on the site that can cause congestions due, for example, to intensive I/O operations during the job initialization phase when downloading input data or accessing databases or performing some software installation.

Setting the site limits allows increasing considerably the efficiency of job execution. However, it needs careful tuning according to the site properties that are usually not exposed in the standard site description, like, for example, the site local network capacity or the throughput of the database system.

3.2. Direct submission to the Grid Computing Elements
Originally, the WLCG grid was used by the LHCb/DIRAC job scheduling by submitting pilot jobs with native gLite middleware tools (first the Resource Broker and later the WMS component). This was the simplest way to build the corresponding pilot Directors, however, this method quickly showed some difficulties. By design the gLite resource brokers are supposed to be central components. They take the site status information from the central information system (BDII) and then decide which site is the most appropriate for the given job. Once the decision is taken, the job is submitted to the chosen site and enters the local task queue there. The capacity of a single resource broker is not sufficient enough to accommodate the load of a large community like the LHCb experiment. Therefore, LHCb is obliged to use multiple independent
resource brokers. In this case the job submission is obviously not optimal. Indeed, all the brokers use
the same site state information and therefore choose the same site as the most attractive one (as shown
in the right branch of figure 2). They start to submit jobs to this site without knowing about each other.
The gLite information system is not reactive enough in order to propagate the changed site status
information to the brokers. As a result the site often gets too many submitted jobs, which wait in the
local task queue, whereas other sites remain underloaded.

With the recent development and wide deployment of the CREAM Computing Element service, it
became easy to submit pilot jobs directly to the sites (left branch in figure 2). The CREAM interface
allows obtaining the more up to date Computing Element status information directly from the service
and not from the BDII system. This information is used by the CREAM Director to decide whether the
load of the site permits to submit pilot jobs provided that there are suitable user jobs in the DIRAC
Task Queue. In this case, there are individual independent Directors for each site and the number of
pilot jobs submitted is chosen in order to avoid unnecessary site overloading. A typical strategy
consists in maintaining a given number of pilot jobs in the site local queue. With the direct pilot job
submission, the sites are competing with each other for the user jobs making their turnaround faster.
This is the ultimate goal of any workload management system. In 2011 more than a half of the LHCb
jobs were executed using direct submission to the CREAM Computing Elements. There are plans to
progressively migrate the pilot job submission to the direct mode for the sites running CREAM
services.

3.3. MPI pilot jobs
In many domains other than the High Energy Physics a large fraction of applications require parallel
execution using the MPI technology. Although this mode of operation can be supported formally by
the sites, in practice MPI jobs are rather fragile due to more complicated requirements, need of various
flavours of the MPI software, etc. On the other hand, using WMS systems with pilot jobs open new
possibilities to manage the MPI jobs. Therefore, the support for the MPI jobs in the DIRAC WMS was
recently developed as an extension in the GISELA Grid Project [9].

The support of MPI jobs by the DIRAC WMS do not require any special effort at the sites, for
example, preinstalled MPI software or configuration of the local batch systems to support this kind of
jobs. Cooperating pilot jobs controlled by a newly developed DIRAC MPI service create the dynamic
virtual clusters – the so called MPI rings. The necessary MPI software is also installed by the pilot
jobs as required by the given MPI user task. In addition, a user space virtual file system can be created
for each ring providing shared disk space for the MPI processes which is often required by the parallel
applications in addition to the message passing mechanism.

As a result, the new DIRAC MPI service allows running parallel jobs even on the sites that do not
declare the corresponding support. The pilot job mechanism allows reusing the already constructed
MPI rings to execute several jobs to increase the job turnaround.

4. Resource Management
One of the main goals of the DIRAC project is providing access to any computing and storage
resource that is accessible for the users. It was first of all providing access to the grid infrastructures:
gLite middleware based grids like EGI, GISELA, WLCG, OSG. Access to the ARC middleware
based grids like NorduGrid was also demonstrated. However, there are many sites that do not
participate in any grid infrastructure and are willing to contribute their resources to their user
communities. Also new types of resources based on the Cloud technology are emerging. The DIRAC
Project was focusing efforts on providing this support.

4.1. DIRAC sites
It is quite common to encounter sites owning considerable computing power in their local clusters but
not willing to be integrated into a grid infrastructure because of the lacking expertise or other
constraints. The DIRAC solution to this problem is similar to the case with the direct pilot job
submission to the CREAM Computing Element service. It consists in providing a dedicated Director component. Two variations of DIRAC site Directors are available.

In the first case, the Director is placed on the gateway node of the site, which is accessible from the outside of the site, and, at the same time, has access to the local computing cluster (figure 3). The gateway host must have a certificate that is used to contact the DIRAC services. The Director interrogates the DIRAC Task Queue to find out if there are waiting user jobs suitable for running on the site. If there are user jobs and the site load is sufficiently low, the Director gets the pilot credentials (proxy) from the DIRAC ProxyManager service, prepares a self-extracting archive containing a pilot job bundled with the pilot proxy and submits it as a job to the local batch system. Once the pilot job starts running on the worker node, it behaves in exactly the same way as any other pilot job, for example, the one submitted through the gLite resource broker as described above.

In the second case, the Director runs as part of the central DIRAC service. A special `dirac` user account is created on the site gateway, which is capable of submitting jobs to the local computing cluster. This account is used by the Director to interact with the site batch system through an ssh tunnel using the `dirac` user account credentials (either ssh keys or password). Otherwise, the behaviour is similar to the previous case. The self-extracting archive containing the pilot job and proxy is transmitted to the site gateway through the ssh tunnel and submitted to the batch system. After the pilot job is executed, its pilot output is retrieved via the same ssh tunnel mechanism.

The first method is used in case the site managers want to have a full control of the pilot submission procedure, for example, providing their own algorithms for evaluation of the site availability. The second method requires minimal intervention on the sites, only creation of the dedicated user account and, possibly, setting up a dedicated queue available for this account. This makes incorporation of new sites very easy. The second method is the most widely used by the LHCb and other DIRAC user communities. The batch systems currently supported include PBS/Torque, LSF, Sun Grid Engine, Condor, BQS. Access to other batch systems can be easily provided by writing the corresponding plugins.

4.2. Cloud support
The support of the cloud resources becomes more and more important with this technology getting more mature and more sites starting to offer their capacity in this form. The support of the commercial
Amazon EC2 cloud was introduced in DIRAC to meet requirements of the Belle II Collaboration at KEK [10]. This development continued to provide access to other open source cloud infrastructures as described in [11].

It is important to note that the resources provided to user communities by cloud infrastructures are very flexible to use due to the virtualization technology, however, they require a special effort to allocate the necessary amount and type of virtual machines to follow the exact needs of the user community at any given moment. Furthermore, the virtual machines should be stopped to free the resources that are not used to avoid wasting. These tasks are not usually covered by the cloud infrastructures and should be addressed by the user community job scheduling systems. The DIRAC Project is filling up this gap with the introduction of the Virtual Machine Scheduler system [11]. Interfacing of the DIRAC WMS to the OCCI standard compliant OpenStack and OpenNebula clouds, CloudStack with an non-OCCI interface together with the earlier work on the Amazon EC2 cloud positions DIRAC as a mean for integrating various cloud infrastructures into a single coherent system from the user perspective. It allows speaking about the DIRAC interware as a middleware to interconnect various computing resources and management infrastructures.

Another important development is the DIRAC Image Engine, adopting a generalized solution unlike “ad-hoc” images. As described in [11], the Image Engine scheme based on the CERNVM image provides a contextualization mechanism for dynamic image configuration as required by the user payloads.

4.3. Resource Status System

The more computing resources are available to the user communities the more effort it takes to monitor their status and to react quickly in case of misbehaviour. In the grid infrastructures there are dedicated systems based, for example, on the Nagios technology [12] to perform continuous testing of the controlled resources. The tests can be both of general nature, like checking the service availability, but also tests specific to a given user community. Since DIRAC supports not only the grid resources and also various user communities, it was necessary to develop a system for various kinds of tests that require a minimal effort of a human operator.

The newly developed Resource Status System (RSS) is fulfilling the above requirements [13]. It provides a framework to describe any kind of resources (computing and storage elements, third party services, etc) and to define specific tests (probes) for each of them. The RSS framework automatically schedules launching of the probes, collects the testing results and sets the status flags in the resources description. The status information is then served to all the DIRAC components needing access to those resources.

5. Data Management

DIRAC provides a rich set of tools to support the Data Management operations. It introduces an abstraction layer for accessing Storage Elements together with plugins for various specific storage types. In particular, it provides access to the SRM standard based Storage Elements, but also standard file servers (FTP, SFTP, HTTP, etc) can be accessed with the same mechanism. DIRAC is also providing its own Storage Element solution within the DISET framework with custom data transfer protocol. In a similar way, DIRAC can give access to different types of File Replica Catalogs. The LFC File Catalog and the newly developed DIRAC File Catalog (DFC) can be used. Several different catalogs can be used together to achieve a higher level of redundancy and/or for complementarity.

5.1. Data Replication System

The Data Replication System is built on top of the Request Management System (RMS). The RMS is providing a repository for requests that can be formulated by users to perform any operation in the DIRAC Framework. In particular, data replication, migration and registration requests can be put into the Request Queue (figure 5). Dedicated central agents pick up pending requests and perform the requested operations on behalf of their owners. The request owner proxy is delegated to the agent in
order to access the Storage Elements and other services like FTS or File Catalogs with proper credentials. The data replication can be done in two different ways: a direct movement between Storage Elements using public network or by using VO-dedicated FTS infrastructure. The Transfer Agent can use both transfer modes in parallel. In case of the FTS transfers it schedules file replications using various strategies to gain the best performance and throughput. These requests are grouped by source-destination channels and then handed over to the FTS service by the FTS Submit Agent. The status of the FTS requests are monitored by the FTS Monitor Agent that takes care of new replica registration in the File Catalog after the FTS request is successfully executed.

![Data Replication System Diagram](image)

Figure 5. Data Replication System

The data replication requests can be put into the Request Queue by the users, but also by production managers performing bulk data transfers. In the latter case, the requests are usually created automatically using dedicated plugins in the DIRAC Transformation System [14]. The Transformation System allows automatic creation of the replication/removal requests triggered, for example, by the initial data registration. The replication requests can also be created as part of the failover strategy. In this case, jobs failing to store their output, due to the unavailability of the destination Storage Element, can temporarily store their data in another Storage Element, and send the data migration request to the Data Replication System. This request will be then accomplished asynchronously as soon as the target Storage Element is back into operation.

5.2. DIRAC File Catalog

The File Catalog (DFC) is a recent addition to the list of DIRAC services. The DFC development started from the needs of the user communities that for various reasons cannot use the grid de facto standard LFC service. In addition some limitations in the performance and functionality of the LFC forced to propose a solution overcoming these flaws.
The DFC is built in the same DISET framework as the rest of the DIRAC components and is very well integrated with the DIRAC infrastructure being part of the standard installation setup. It makes use of the DIRAC Configuration Service to get information about the users and groups as well as the Storage Elements where the registered file replicas are residing [15]. The DFC has a similar functionality to that of the LFC as a File Replica Catalog with a special emphasis on the efficiency of the bulk operations like replica look-up for a large set of files. It supports also efficient storage usage reports necessary to build storage monitoring systems as well as to impose user storage quota rules. The key feature that distinguishes DFC from other similar solutions is the ability to add user defined metadata and provenance information to its entries. All these functionalities are integrated in a single service providing a number of advantages with respect to other alternatives, like for instance a combined use of LFC and AMGA grid Metadata Catalog [16]. It makes complex queries involving both metadata and replica information more efficient and reduces the service maintenance effort by avoiding independent services for replica and metadata catalogs.

The DFC was used heavily in the data production system of the ILC/CLIC Collaboration with a very useful feedback for the developers that helped to improve the catalog stability and performance [17]. The DFC was also thoroughly evaluated by the BES Collaboration in IHEP, Beijing, especially in the Metadata Catalog part comparing its performance with the one of AMGA [18]. As a result of this evaluation, the BES Collaboration has chosen the DFC for the phase III of the experiment.

6. Interfaces

The DIRAC Project is first of all providing middleware for building distributed computing systems. But much attention is paid also to the convenience of its usage. Historically first, DIRAC was providing a rich command line interface made deliberately in a similar style as the edg/gLite commands. This interface counts now more than two hundreds commands to access the functionality of all the DIRAC subsystems. More commands are being added along with the development of new components. Particular user communities can create their own commands within the same framework and style. As the DIRAC software is written almost entirely with the Python programming language, the project is providing a comprehensive Python API for all the aspects of the system. This library is used to build high-level applications on top of DIRAC. For example, the Ganga Project [28] is using this API to provide the DIRAC backend to its user frontend interface. Other user interfaces are also available and are getting more and more attention of both developers and users. Those are detailed in the next sections.

6.1. Web Portal

The command line interface and API’s are very powerful tools for the expert users to get access to the full power of DIRAC. However, with more user communities starting to use DIRAC, it starts to be very important to provide simpler interfaces that resemble more familiar desktop applications. This became very evident after a series of tutorials given by the DIRAC experts to the users coming from the application domains other than the High Energy Physics. The DIRAC Web Portal provides such friendly platform independent graphical user interface [19] which is an essential system performance monitoring tool for both users and administrators. It is still being actively developed with the goal of providing access to all the DIRAC functionalities. As an example, the DIRAC File Catalog browser is provided allowing various operations on files and on their metadata including data downloading from the Web (figure 6). Another example is Job Launch Pad that allows a simple way to submit tasks to DIRAC with similar options as the python API.
6.2. DIRAC RESTful interface

There is a large number of already existing application portals which are carefully crafted to capture the user input, launch jobs transparently behind the scene and visualize the output results [20]. The portals are becoming more and more popular so that more users are starting to use them with more resources required to run the user jobs. The portal frameworks can use either a single backend computing infrastructure or several different types of resources. The user jobs are usually pushed to one of the available resources with the help of dedicated Job Submitters or using standard computing resource interfaces. For the latter, the JSAGA, the Java implementation of the OGF SAGA standard, is one of the most popular interfaces [29]. However, with this mode of operation, all the resources are independent which makes it difficult to use them in a balanced and optimized way. This is why application portals as user interfaces and the Community WMS systems like DIRAC as managers of the backend computing resources can nicely complement each other.

A demonstration of integration of gUSE/WS-PGRADE and InSilicoLab portals with DIRAC was performed recently. This increased enormously the amount of computing resources available to the portal users [21].

Having just a rich Python API to build application specific user interfaces turned out to be quite limiting with a lot of already existing application Web Portals that are written in languages other than Python. Therefore, a language neutral DIRAC interface became necessary. Such an interface is now provided in a form of a RESTful service [22] making available the core DIRAC functionality to application portals written in languages like Java or PHP. A special attention is paid to the security aspects of the interface to avoid regressions compared to a strict security model of the DIRAC Web Portal and the DISET protocol itself. OAuth with JSON over HTTP technologies turned out to be the most suitable solution to achieve this goal.

7. Community Users

Several user communities use the DIRAC middleware now for their everyday work. More experiments and user groups are evaluating the functionality and performance of DIRAC comparing them with other solutions that exist on the market today.

7.1. LHCb

The LHCb Collaboration is historically the first and still is the largest user of the DIRAC system [21]. It is used for all the LHCb distributed computing tasks: workload and data management but also high-level data production workflows and user analysis. The LHCb DIRAC installation is managing up to 35,000 simultaneous jobs (figure 7) and over 100,000 jobs executed per day. The limits are determined
by the amount of available resources and not by the capacity of the DIRAC WMS. About 10PB of data is stored and registered in the LHCb catalogs. The LHCb DIRAC installation is currently the most powerful one. Its hardware configuration includes a cluster of about 12 mid-range servers located at CERN and hosting the central DIRAC services including 6 MySQL database servers. Some of the MySQL servers are slave mirrors of the others for redundancy and the database backup. Some of the servers are running complete DIRAC installations for testing purposes, connected to the same pool of computing resources. Six additional servers at each of the LHCb Tier1 computing centres in Europe complement this cluster. These servers are running redundant instances of the Configuration and Request Management services to ensure their 100% availability at any moment.

Along with using almost all the DIRAC core components, LHCb is developing and maintaining a large number of extensions that are built in the same software framework. The extensions provide the LHCb services and plugins to the core DIRAC components to manage specific data and workflows. LHCb is using LFC as its main Replica Catalog, however, the DIRAC File Catalog is being evaluated now. In particular, the ability to generate storage usage reports in real time is a very demanded feature. A custom Metadata Catalog called Bookkeeping database, and the Production Management System for creating and controlling the data production workflows are among the LHCb DIRAC extensions.

DIRAC based LHCb distributed computing system was in place from the day one of the LHC start and contributed significantly to the success of the first years of the experiment.

Figure 7. LHCb simultaneously running jobs

7.2. High Energy Physics and Astrophysics Experiments

The Belle experiment at KEK performed a careful evaluation of DIRAC in 2009-2010. In particular, the DIRAC was used to demonstrate the possibility to integrate in a single system different computing resources like the KEK Computing Centre, grid sites and also the Amazon EC2 commercial cloud [24]. After the prototype successfully demonstrated the feasibility of such integration, the DIRAC middleware was chosen as a base element for the Computing Model of the phase II of the Belle experiment due to start taking data in 2015.

The ILC Collaboration adopted DIRAC for their data production and user analysis system in early 2010. This turned out to be an essential tool for modelling various aspects of the physics detectors at the future linear collider [17].
The ILC DIRAC installation is located at CERN and consists of 2 dedicated servers one of which is used mostly for running the DFC service. The ILC developers contributed significantly to the development and testing of the DFC that was updated following the ILC requirements. The ILC instance of DFC contains about 3 million file records. Both Replica and Metadata Catalog features are heavily used.

The BES Collaboration at IHEP, Beijing has also chosen DIRAC to build their distributed computing system for the phase III of the experiment. A special attention is paid to the possibility to include non-grid computing clusters at various sites into the common pool of resources together with grid sites. A careful evaluation of the DFC was done. In particular, the performance of its Metadata part was tested and compared with the AMGA Metadata service. As a result, DFC was adopted for both Replica and Metadata Catalog tasks [18].

Among other user communities starting to use DIRAC we can mention the CTA and Fermi-LAT Astrophysics experiments [25]. The SuperB Collaboration is carrying out an extensive assessment of DIRAC features and performance as part of their Computing Model preparation process [26].

8. DIRAC as a Service

Large user communities usually have enough number of computing experts in order to build dedicated distributed computing systems. However, such communities are representing a small fraction of the number of users needing access to computing resources. Most of the users are working in small groups with a small or no expertise in computing systems. For such users installation and maintenance of software systems like DIRAC is a prohibitively complicated task. On the other hand these users are in particular need for a simple interface for running their applications on the grids. Therefore, several National Grid Initiative (NGI) projects have chosen to provide DIRAC services to their users as part of their grid infrastructures. For the moment, such installations exist for GISELA/EELA Latin America grid, France-Grille and IBERGrid (Spain-Portugal) NGIs.

It is essential that these DIRAC service installations are complemented with user tutorials as well as assistance in porting the user applications to the grid. The user support is equally important and should be integrated as part of the corresponding grid user support systems.

DIRAC was originally developed to support a single user community or Virtual Organization (VO). In this kind of installations many different communities can access the resources controlled by DIRAC. In this case, there is a problem that the computing resources should be described including details of access rights for different VOs. The VO access rights must be taken into account at the moment of the user job scheduling. In the WMS systems the pilot jobs are usually running with credentials that are the same for all the jobs of the given VO. These are the so-called “generic” pilot jobs. However, for security and accountability purposes it is not possible to mix the generic pilot jobs from different VOs. To overcome this difficulty, the central DIRAC Task Queue is effectively split into several parts each dedicated to the tasks of users of a given VO (figure 8). The VO specific pilot jobs are configured to match only the tasks of the users from their corresponding VO.

Figure 8. Split WMS Task Queue in the multi-VO DIRAC installation
Similarly, the storage resources are also described in the DIRAC Configuration Service together with the information about which VO can access each of them. This information can be taken into account while performing storage access operations using DIRAC tools.

9. DIRAC Community

The DIRAC Project has gone a long way from the LHCb Monte-Carlo production system to a general-purpose middleware provider. The community of the DIRAC users and developers is growing and needs its own organization. The recently created DIRAC Web site at http://diracgrid.org contains general information about the project, news, the project calendar as well as links to other project resources. Users and developers have access to various guides and tutorials. They can also put their questions in the DIRAC forum web page created by means of the Google Groups service. Separate forums for end users, developers and DIRAC system administrators are available.

The DIRAC developers are contributing their codes using the Git code management system. The Project code repository is hosted by the GitHub service [27]. This service offers a complete software project management environment including a WiKi engine, issue tracker, various visualizations of the project structure and statistics.

The Project organizes regular meetings with users and developers. The meetings are used to exchange the development news, perspectives and usage experience. The mandatory part of the meetings is a joint examination of the use cases to find the most appropriate strategy for porting the applications to the grid infrastructures. This turned out to be the most efficient way for introduction of the new users into the grid community.

10. Conclusions and outlook

Multiple user communities from High Energy Physics, but also others, now use the DIRAC middleware. The new applications stimulating the development of new features of DIRAC become quickly an important driving force for the project, determining the main directions of its evolution. Here are some of the development lines being actively pursued by the project.

The computing resources become more and more diversified with the necessity of incorporating non-grid HPC clusters, various open source and commercial clouds, GPU based farms, etc. DIRAC proposes a solution for aggregation of any combination of resources in a coherent system seen as a single entity easy to use and monitor. The work on making it a universal tool for connecting new novel computing means will certainly continue.

Managing large volumes of data in the distributed computing environment stays a difficult task. DIRAC has most of the components to cope with it. Storage and Catalog services access tools, powerful Replica and Metadata Catalog, high level services for massive data driven automated operations, all these already exiting components should be now complemented by a user friendly functional Web interface to make these tasks accessible to multiple user communities.

The DIRAC middleware interconnecting various distributed services, or interware, becomes more and more complicated for the end users to install, configure and maintain. Therefore, the DIRAC functionality provided as a ready to use service becomes a more attractive choice for the users, especially those that are non-experts in the computing matters. Provisioning DIRAC as a Service by the grid infrastructures will continue on a larger scale. Development of the tools necessary for efficient maintenance of such installations will continue. The goal is to provide users with easy to use application portals connected behind the scene to a powerful workload and data management systems.

References

[1] T Maeno, PanDA: distributed production and distributed analysis system for ATLAS, 2008 J. Phys.: Conf. Ser. 119 062036
[2] R Egeland et al, PhEDEx Data Service, 2010 J. Phys.: Conf. Ser. 219 062010
[3] S Bagnasco et al, AliEn: ALICE environment on the GRID, 2008 J. Phys.: Conf. Ser. 119 062012
[4] http://diracgrid.org
[5] A Tsaregorodtsev et al, DIRAC3: The New Generation of the LHCb Grid Software, 2010 *J. Phys.: Conf. Ser.* **219** 062029

[6] A.Casajus, R.Graciani, DIRAC Distributed Secure Framework, 2010 *J. Phys.: Conf. Ser.* **219** 042033

[7] A.Casajus, R.Graciani, Executor framework for DIRAC, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[8] A.Casajus, R.Graciani, A.Tsaregorodtsev, DIRAC pilot framework and the DIRAC Workload Management System, 2010 *J. Phys.: Conf. Ser.* **219** 062049

[9] V.Hamar, A. Tsaregorodtsev, MPI support in the DIRAC Pilot Job Workload Management System, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[10] T.Fifield, A.Carmona, A.Casajús, R.Graciani, M.Sevior, Integration of cloud, grid and local cluster resources with DIRAC, 2011 *J. Phys.: Conf. Ser.* **331** 062009

[11] V.Mendez, V.Fernandez, The Integration of CloudStack and OpenNebula with DIRAC, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[12] http://www.nagios.org

[13] V.Bernardoff, F.Stagni, M.Ubeda, Grid administration: towards an autonomic approach, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[14] A.Smith, A.Tsaregorodtsev, DIRAC: data production management, 2008 *J. Phys.: Conf. Ser.* **119** 062046

[15] S.Poss, A.Tsaregorodtsev, DIRAC File Replica and Metadata Catalog, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[16] N.Santos, B.Koblitz, Distributed Metadata with the AMGA Metadata Catalog, in Workshop on Next-Generation Distributed Data Management, HPDC-15, Paris, France, June 2006

[17] http://www.linearcolider.org

[18] C.Nicholson, Z.Deng, L.Lin, W.Li, File and Metadata Management for BESIII Distributed Computing, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[19] A.Casajus, M.Sapunov, DIRAC: Secure web user interface, 2010 *J. Phys.: Conf. Ser.* **219** 082004

[20] P.Kacsuk: P-GRADE portal family for Grid infrastructures, Concurrency and Computation: Practice and Experience journal, Volume: 23, Issue: 3, 2012, pp. 235-245; http://www.gridsite.org

[21] A.Puig, D.Viana, A.Casajus, R.Graciani, Integration of WS-PGRADE/gUSE portal and DIRAC, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[22] A.Casajus, R.Graciani, A.Tsaregorodtsev, DIRAC RESTful API, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[23] F.Stagni et al, LHCbDIRAC: distributed computing in LHCb, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[24] T.Kuhr, T.Hara, Computing at Belle II, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[25] L.Arrabito et al, Application of the DIRAC framework in CTA: first evaluation, Proceedings of the CHEP’2012 International Conference, New-York, May 2012; S.Zimmer, The Fermi-LAT Dataprocessing Pipeline, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[26] A.Fella, G.Donvito, A.Gianoli, E.Luppi, B.Santeramo, DIRAC evaluation for the SuperB experiment, Proceedings of the CHEP’2012 International Conference, New-York, May 2012

[27] http://github.com

[28] D.C.Vanderster et al, Ganga: User-friendly Grid job submission and management tool for LHC and beyond, 2010 *J. Phys.: Conf. Ser.* **219** 072022

[29] S.Reynaud, Uniform Access to Heterogeneous Grid Infrastructures with JSAGA, in Production Grids in Asia, Ed. S.Lin, ISBN 978-1-4419-0045-6, Springer, 2010, pp. 185-198.