DIRAC: A Community Grid Solution

A Tsaregorodtsev [1], M Bargiotti [2], N Brook [3], A Casajus Ramo [4], G Castellani [2], Ph Charpentier [2], C Cioffi [5], J Closier [2], R Graciani Diaz [4], G Kuznetsov [6], Y Y Li [7], R Nandakumar [6], S Paterson [2], R Santinelli [2], A C Smith [2,10], M Seco Miguelez [8], S Gomez Jimenez [9]

[1] Centre de Physique des Particules de Marseille, 163 Av de Luminy Case 902 13288 Marseille, France
[2] CERN CH-1211 Genève 23, Switzerland
[3] H. H. Wills Physics Laboratory, Royal Fort, Tyndal Avenue, Bristol BS8 1TL, UK
[4] University of Barcelona, Diagonal 647, ES-08028 Barcelona, Spain
[5] University of Oxford, 1, Keble Road, Oxford OX1 3NP, UK
[6] Rutherford Appleton Laboratory, Chilton, Didcot Oxon. OX11 0QX, UK
[7] University of Cambridge, Wilberforce Road, Cambridge CB3 OWA, UK
[8] University of Santiago de Compostela, Campus Universitario Sur, ES-15706 Santiago de Compostela, Spain
[9] University Rovira i Virgili, Campus Sescelades, Avinguda dels Països Catalans, 26 Tarragona, Spain

E-mail: atsareg@in2p3.fr

Abstract. The DIRAC system was developed in order to provide a complete solution for using distributed computing resources of the LHCb experiment at CERN for data production and analysis. It allows a concurrent use of over 10K CPUs and 10M file replicas distributed over many tens of sites. The sites can be part of a Computing Grid such as WLCG or standalone computing clusters all integrated in a single management structure. DIRAC is a generic system with the LHCb specific functionality incorporated through a number of plug-in modules. It can be easily adapted to the needs of other communities. A special attention is paid to the resilience of the DIRAC components to allow an efficient use of non-reliable resources. The DIRAC production management components provide a framework for building highly automated data production systems including data distribution and data driven workload scheduling. In this paper we give an overview of the DIRAC system architecture and design choices. We show how different components are put together to compose an integrated data processing system including all the aspects of the LHCb experiment - from the MC production and raw data reconstruction to the final user analysis.

1. Introduction

The LHCb Collaboration is constructing one of the four experiments to run on the future LHC proton-proton collider at CERN, Geneva. The amount of data that will be produced by the experiment annually is so large that it necessitates the development of a specialized system for the data

[10] Marie Curie fellowships program

production, reconstruction and analysis. The DIRAC project of the LHCb Collaboration was started to provide such a system.

Originally the DIRAC project was devoted to the simulation data production. The goals that the project developers fixed for themselves were:

- Seamless use of the various heterogeneous computing resources available to the LHCb Collaboration;
- Light implementation and deployment on the local sites;
- Minimal effort needed from the LHCb site managers to run the system; single production manager should be able to run the whole LHCb production system.

The stated goals have to be supported by a careful choice of the system architecture, the component design and by the implementation technologies. After some experimentation the choice was made in favour of a services oriented architecture supplemented by a network of lightweight agents, which are animating the whole system. This architecture has been proven to be very efficient for simulation data production tasks [1]. However, now the emphasis is shifted towards the data reprocessing and analysis tasks. Therefore, the system was extended to the new class of workload, which is characterized by the constraints due to availability of the input data and by the necessity to minimize the task total execution times. It is important to note that the system provided for the individual user analysis has much higher requirements with respect to the overall efficiency and stability.

The experience of the DIRAC project shows clearly the advantages of the incremental approach where the system gradually evolved from managing relatively simple simulation data production tasks to more complicated data reprocessing tasks including the final user data analysis. This allowed to build a versatile and consistent system out of components based on the same design principles and framework implementation. What is also important is that DIRAC was developed by a single concerted team of developers all sharing the same vision of the project concepts.

The amount of data production workload is very high but the production activity can be planned before the data to be processed become available. This allows creating tools for automatic job preparation and submission and those tools were added recently to the project.

One more important problem arises from the necessity to run different kind of jobs with different priorities and requirements on the same computing resources. Together with potentially complex policies that the Collaboration might want to introduce, this requires an architecture design in which these complex rules can be efficiently applied.

These new requirements can be met by naturally extending the agents based architecture where the agents are deployed right on the worker nodes thus building dynamically an overlay network of readily available resources. Altogether, it allowed DIRAC to evolve to a complete system for all the computing tasks that members of the LHCb Collaboration will have to carry out using the highly distributed computing resources.

In this paper we describe in Section 2 the DIRAC design principles and architecture and justify the choice of its components and their implementation. The framework components overview is presented in Section 3. The Workload and Data Management Systems are described in Sections 4 and 5 respectively. The higher level Production Management tools together with the experience gained in the recent production runs are discussed in Section 6. Section 7 is devoted to conclusions and outlook for future work.

2. DIRAC overview

2.1. Scope of the project

Most of the computing resources needed by the LHC HEP experiments as well as for some other communities are provided by Computing Grids. The Grids are providing a uniform access to the computing and storage resources which simplifies a lot their usage. The Grid middleware stack provides also the means to manage the workload and data for the users. However, the variety of requirements of different Grid User Communities is very large and it is difficult to meet everybody's needs with just one set of the middleware components. Therefore, many of the Grid User

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 [2] or PHEDEX data management system [3]) or close to complete Grid solutions (AliEn system [4]). DIRAC project is providing a complete Grid solution for both workload and data management tasks on the Grid.

Although developed for the LHCb experiment, it is designed to be a generic system with LHCb specific features well isolated as pluggable modules. It allows to construct medium sized grids of up to several thousands processors by uniting PC farms with most widely used cluster software systems as well as individual PCs within its integrated Workload Management System. DIRAC is also providing means for managing tasks on Grid resources taking over the workload management functions. The DIRAC Data Management components provide access to standard grid storage systems based on the SRM standard interface or ordinary (S)FTP, HTTP file servers. The File Catalog options include the LCG File Catalog (LFC) as well as a simple DIRAC File Catalog. The modular organization of the DIRAC components allows selecting a subset of the functionality suitable for particular applications or easily adding the missing functionality. All these features allow positioning DIRAC as a complete Grid solution for a medium size community of users.

2.2. Architecture and components

DIRAC follows the Service Oriented Architecture (SOA) paradigm. Its main components are presented in Figure 1 [6]. The DIRAC components can be grouped in the following 4 categories: Resources, Services, Agents and interfaces.

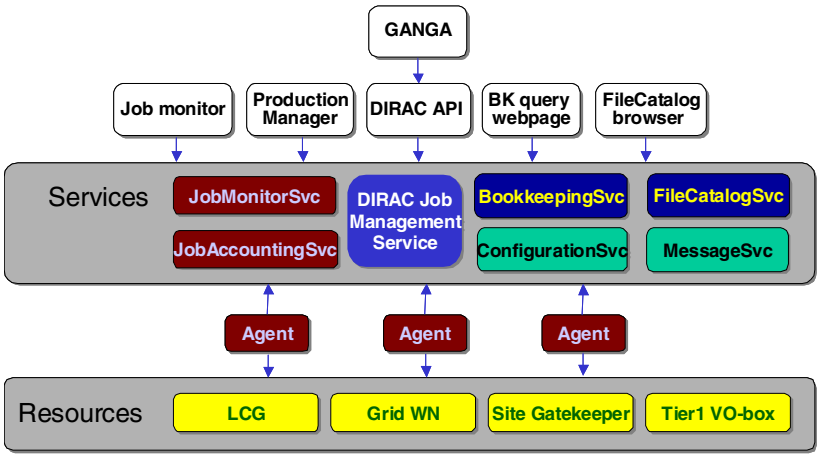

Figure 1 DIRAC architecture overview

2.2.1. Resources.

Resources are components which provide access to the computing and storage facilities available to DIRAC. Computing resources are available in a form of individual PCs, computing farms with various batch systems or computing grids. DIRAC includes clients for most of the popular batch systems: PBS/Torque, LSF, Sun Grid Engine, Condor, and BQS. Recently, the support for the Microsoft Compute Cluster was also added [5]. The clients exist also to access the Computing Elements in the EGEE grid which are based on the GRAM interface. The support for the future EGEE Computing Elements will be provided as soon as they will become available. Apart from the EGEE grid, the client for the NorduGrid was also provided. Together with clients which are representing individual Windows or Linux PCs as DIRAC Computing resources, DIRAC covers all the possible resources available to the LHCb experiment. If necessary, new types of the computing resources can be easily added.

DIRAC does not provide a complex Storage Element service capable of managing multiple disk pools or tertiary storage systems. It includes however a Storage Element service which is providing

access to a disk storage managed by a POSIX compliant file system. The data access is done through a proprietary DISET protocol which is part of the DIRAC services framework (see below Section 3). To access other storage systems DIRAC provides various clients which are representing many different services in a uniform way. Clients exist to access Storage Elements with the SRM standard interface as well as clients for the most popular data access protocols: gridftp, (s)ftp, http, and some others.

Sometimes the same physical storage is available through several different protocols. This can be expressed in the storage configuration description and the DIRAC data access tools will be able to use either of the possible protocols in an optimal way. This also adds redundancy ensuring higher storage availability in case of intermittent failures.

2.2.2. Services.

The DIRAC system is built around a set of loosely coupled services which keep the system state and help to carry out workload and data management tasks. The services are passive components which are only reacting to the requests of their clients possibly soliciting other services in order to accomplish the requests.

All services and their clients are built in the DISET framework which provides secure access and flexible authorization rules. Each service has typically a MySQL database backend to store the state information. The services as permanent processes are deployed centrally at CERN and on a number of hosts (VO-boxes) at several sites. The number of sites where services are installed is limited to those with well-controlled environment where an adequate support can be guaranteed. The services are deployed using system start-up scripts and watchdog processes which ensure automatic service restart at boot time and in case of service interruptions or crashes. Standard host certificates typically issued by national Grid Certification Authorities are used for the service/client authentication.

The services are accepting incoming connections from various clients. These can be user interfaces, agents or running jobs. But since services are passive components, they have to be complemented by special applications to animate the system.

2.2.3. Agents.

Agents are light and easy to deploy software components which are running as independent processes to fulfil one or several system functions. All the agents are built in the same framework which is organizing the main execution loop and provides a uniform way for deployment, configuration, control and logging of the agent activity.

Agents are running in different environments. Those making part of DIRAC subsystems, for example Workload Management or Data Distribution, are usually deployed close to the corresponding services. They watch changes in the service states and react accordingly by initiating actions like job submission or result retrieval. Agents can run on a gatekeeper node of a site controlled by the DIRAC Workload Management System. In this case they are making part of the DIRAC WMS ensuring the pull job scheduling paradigm.

Agents can also run as part of a job executed on a Worker Node as so called “Pilot Agents”.

2.2.4. Interfaces.

The DIRAC functionality is exposed to the system developers and to the users in a variety of ways. The DIRAC main programming language is Python and programming interfaces (APIs) are provided in this language. Each service is complemented by the corresponding client class which is in the simplest case is very thin and just translates the service interface calls.

For the developers making use of the DIRAC system, the functionality is provided as Python API which is encapsulating in a small number of classes all the methods necessary to build higher level applications. This API is used, for example, by the GANGA grid user interface project [7] to provide access to the DIRAC back-end.

For the users of the DIRAC system the functionality is available through a command line interface. Some DIRAC subsystems have specialized shells to work with. The shells are having extra advantage of having online help assistance. They also keep the user session state which can be very useful, for example for meta catalog browsing and other tasks.

DIRAC is also providing Web interfaces for users and system managers to monitor the system behaviour and to control the ongoing tasks. The Web interfaces are based on the DIRAC Web Portal framework which is providing secure access to the system service using X509 certificates loaded into the user browsers.

2.3. Some Design Principles

Here are presented several design patterns that are widely used in the DIRAC system in order to increase its efficiency and resilience to failures.

2.3.1. Redundancy

The distributed computing environment is intrinsically unstable. Of course, it is necessary to spend much effort in order to increase the availability of all the services, especially in the grid environment which has more means for their monitoring and control. However, it is impossible to eliminate all the reasons for failures because of the software and hardware faults but also because of human mistakes. Therefore, the distributed computing systems should be built in such a way that the damage of temporary services unavailability is minimized. It is unacceptable to loose results obtained after consumption of a large amount of the computing resources or have the whole system blocked because one of its components is down. The distributed computing system in this case should continue to function although with a reduced capacity. This can be achieved by adding redundancy to each operation where each failure can be retried with either another instance of a service or at a later time when the faulty service is returning back into operation.

In DIRAC the redundancy is achieved in several ways. First, the information which is vital to the successful system operation is duplicated at several services to ensure that at least one copy will be available to client requests. This is done for the DIRAC Configuration Service and for the File Catalog each of which has several mirrors kept synchronized with the master instance. Together with the necessary redundancy it allows to introduce a certain load balancing reducing the services load.

Second, all the important operations which success is mandatory for the functioning of the system without losses are executed in a failover recovery framework which allows retrying them in case of failures. All the information necessary for the operation execution is encapsulated in an XML object called request which is stored in one of the geographically distributed request databases. For LHCb in the WLCG environment these databases are deployed on the VO-boxes. Special agents running close to the request databases are attempting to accomplish the stored operations as many times as necessary until the final success. For the data management operations, for example for initial data file uploading to some grid storage, in case of failure the files are stored temporarily in some spare storage element with a failover request to move the data to the final destination when it will become available.

2.3.2. System state information

The information about the system state is one of the most vital ingredients of the successful operation. It is very important to distinguish the slowly changing static information such as the system configuration data and fast changing dynamic information, for example the available capacity of the computing resources. It is important not to mix the static and dynamic data within the same information system. In DIRAC the static configuration data is made available to all the clients via the Configuration Service (CS) which has multiple reservations as described above. Moreover, this information can be cached on the client side for relatively short periods without risks of the client misbehaviour. Keeping the static and dynamic information separately reduces the risk of compromising the static information due to system overloading. The dynamic information is in most cases looked for at its source. For example, the current state of a computing resource is obtained directly from the batch system or from the computing element to avoid obsolete data. This is why, for example, the DIRAC Workload Management System is following the “pull” paradigm where the computing resources availability is examined by a network of agents running in close connection to the sites.

2.3.3. Requirements to sites

The main responsibility of the sites is to provide resources for the common use in a grid. The resources are controlled by the site managers and made available through middleware services (Computing and Storage Elements). It is not unusual that the middleware expertise level is sometimes not very high at the sites especially when the site is a newcomer to the grid community. It is important to keep requirements on the site operations as simple as possible to lower the thresholds for joining the grid and thus making more resources available. In particular, specific requirements of different VO's to the site operations should be minimized. DIRAC is putting very low requirements on the sites asking for no special support for the LHCb VO. For example, unlike other VO's there are no requirements to run special LHCb services on sites like local File Catalogs or data access infrastructures. The data production activity is requiring no special support from the site managers apart from ensuring availability of the standard services. There is also no special requirement on VO job optimization and accounting. All this allows exploiting numerous sites providing resources to the LHCb VO by a small central team of production managers.

3. DIRAC Framework

The DIRAC framework for building secure SOA based systems is providing generic components not specific to LHCb which can be applied in the contexts of other VOs as well [8]. The framework is written in Python language and includes the following components :

- DISET (DIRAC Secure Transport) secure communication protocol
- Web Portal framework
- Configuration System
- Logging System
- Monitoring System

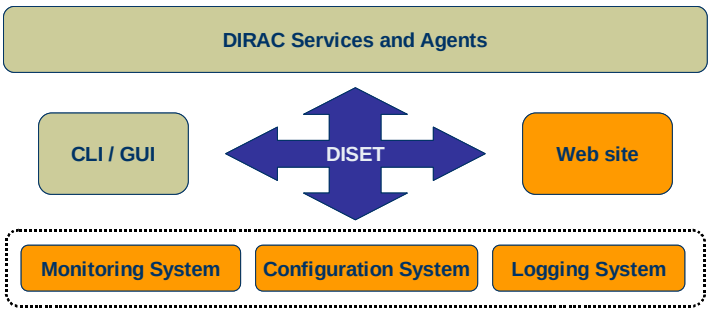

Figure 2 DIRAC Framework components

3.1. DISET protocol

Originally DIRAC used the XML-RPC protocol for client/service communications. This was a simple and efficient choice for the early stages of the project. The original XML-RPC was enhanced with GSI compliant authentication mechanism [9]. The more intensive use of the system made it necessary to increase the efficiency of the client/service interactions which resulted in the introduction of a proprietary DISET protocol.

The latest DISET framework is providing distributed applications with a standard Grid GSI based authentication mechanism. It allows definition of configurable authorization rules which can be specified for each service method using user identities and groups. Finer grained rules can be coded for each method as well.

DISET is providing also a complete framework for creating distributed services. It has a built-in support for multiple threads and automatic logging of the history of the requests. Along with the RPC functionality it allows also transfers of files or groups of files in the same interface with a special binary protocol. With DISET developers can also build portals which serve as a single access point for a secure access to a set of services with a single authentication step. All these mechanisms are employed in construction of the DIRAC distributed system

3.2. Web portal framework

The Web portal framework allows building Web interfaces to DIRAC services. It provides authentication based on user grid credentials and user groups which can be selected during the interactive session. The framework is using the DISET portal functionality to redirect client requests to corresponding services and to collect responses. It provides means to organize the contents of the DIRAC Web sites using the Pylons contents management system [10]. All the monitoring and control

tools of a DIRAC system are exported through the Web portal which makes them uniform for users working in different environment and on different platforms.

3.3. Configuration Service

The Configuration Service is built in the DISET framework to provide static configuration parameters to all the distributed DIRAC components. This is the backbone of the whole system and necessitates excellent reliability. Therefore, it is organized as a single master service where all the parameter updates are done and multiple read-only slave services which are distributed geographically, on VO-boxes at Tier-1 LCG sites in the case of LHCb. All the servers are queried by clients in a load balancing way. This arrangement ensures configuration data consistency together with very good scalability properties.

3.4. Logging and Monitoring Services

All the DIRAC components are using the same logging facility which can be configured with one or many back-ends including standard output, log files or external service. The amount of the logging information is determined by a configurable level specification. One important use of the logger is the possibility to report to the Logging Service where all the distributed components are reporting encountered cases of system failures. This service is accumulating information for the analysis of behaviour of the whole distributed system including third party services provided by the sites and central grid services. The quick error report analysis allows spotting and even fixing the problems before they are hitting the user.

The Monitoring Service is collecting activity reports from all the DIRAC services and some agents. It presents the monitoring data in a variety of ways, e.g. historical plots, summary reports, etc. Together with the Logging Service, it provides a complete view of the health of the system for the managers.

4. Workload Management System

Originally DIRAC was developed to support the production of the Monte-Carlo simulation data for the LHCb experiment. The Workload Management System (WMS) was the central component of this activity. The necessity of integration of various heterogeneous resources within the same system led to the design of the WMS with a central Task Queue and a network of light agents [11]. The new design allowed extending the DIRAC WMS to also data processing and analysis tasks [12]. In the following the main advantages of this approach are discussed.

4.1. Overlay network

The use of light pilot agents as part of the WMS with a "pull" scheduling paradigm is now widely accepted as a way to hide the fragility of the underlying distributed resources and thus increase the efficiency of their usage as seen by the end users. This is achieved by checking the sanity of the exact operational environment in which the user jobs will be executed before actually pulling the real workload from the central Task Queue. But this approach has also other advantages as well.

The agents, which are deployed close to the computing resources – either on the computing cluster gatekeepers or right on the grid worker nodes – are presenting various kinds of resources in a uniform way. Therefore, they are forming an overlay network which is the only one seen by the DIRAC WMS. This makes it easy to integrate the whole variety of computing systems from single PCs to various grids into a single community system. To include a new type of a computing resource it is sufficient to develop the corresponding specialized agent. Therefore, it was trivial to combine with the DIRAC WMS such back-ends as WLCG and NorduGrid Grids, standalone Linux (e.g. PBS or Sun Grid Engine) and Windows Compute Clusters, and standalone PCs, for example from the Online LHCb computing farm with no batch system installed.

4.2. Managing VO policies

The WMS architecture with the central Task Queue offers also an elegant solution to the problem of the VO workload optimization and application of the VO policies by managing priorities of the tasks

coming from different user groups. In the systems without central queues, the job priorities can be only applied at the level of the site local batch systems. If even it is possible in principle, in practice this requires a lot of maintenance work because each change in the VO policies should be applied at all the sites by local system managers. This takes a lot of time in large distributed systems like Grids and it is extremely difficult to maintain consistent policies across all the sites. It is important also that the application of VO policies can not be precise because of uncertainties due to badly defined waiting times in the local batch queues.

The Central Task queue together with the pilot agent approach allow for application of the VO policies in a single place which simplifies a lot the definition and maintenance of the policy rules. The policies are applied consistently across the whole distributed system even in the case of heterogeneous resources, for example across several grids (**Error! Reference source not found.**). The application of the policies is precise because of the "late" job scheduling where it is guaranteed that the job starts immediately on the Worker Node when pulled by the pilot agent from the Task Queue. In fact, the whole distributed system can be regarded as a single large batch system where the standard scheduling tools can be applied. In DIRAC the policies can be applied by either a simple job priority calculator based on configurable group and user static priorities or more complex calculators. In particular, the use of the MOAB commercial task scheduler in conjunction with the DIRAC central Task Queue was demonstrated [13].

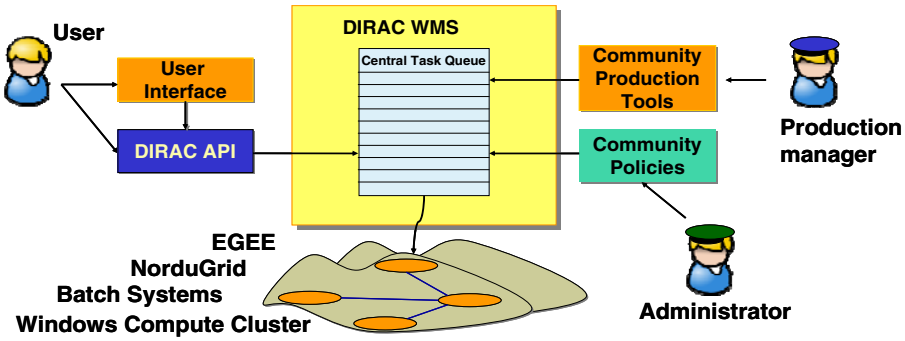

Figure 3 DIRAC WMS with the central Task Queue

An important prerequisite of the VO policies application in the systems with the central Task Queues is the necessity of usage of generic pilot agents where the identity of the owner of the pilot job is not necessarily the same as the one of the owner of the actual workload. This allows the pilot agent to pick up the highest priority job across the whole user community at each scheduling operation. The changing of the ownership of the workload executed on a Worker Node have seen doubts of the site managers because of potential security concerns. However, introduction of a ***glexec*** authorization tool which can be used on the Worker Nodes to apply site policies to the user workload obtained by the generic pilot agent resolves this problem [14].

5. Data Management System

The DIRAC project is providing a range of tools to handle data management tasks. The tools can be classified in several levels (Figure 4).

The low level tools include clients of various kinds of storages and file catalogs. DIRAC project provides only a simple Storage Element service accessible through the DISET protocol with a disk file system back-end. Otherwise, clients of all the mostly used storage systems are provided exposing to the users a uniform API.

As was mentioned above, the main File Catalog in use by LHCb is LCG File Catalog (LFC) with a single master write accessible instance and multiple read-only mirrors. DIRAC is also providing its own simple File Catalog solution which is used in LHCb in the context of the Production Management System although provides all the basic functionalities needed for general applications.

The Replica Manager class which is encapsulating all the basic file management operations: uploading, replication, registration. The Replica Manager is masking the diversities of different storage systems and can handle several file catalogs at the same time. For the Grid Storage Elements the Replica Manager functionality is based on the GFAL/LCG utilities.

The higher level Data Management components include an automatic Data Distribution System and a Data Integrity Checking System. The Data Distribution System allows defining destination storages for all the types of data used by a VO before the data actually become available. Once the first replicas of the data to be distributed are registered in the File Catalog, the replication requests are formed and sent for execution automatically. The data transfers are effectuated either by means of the FTS service of the WLCG project or by specialized DIRAC agents [15].

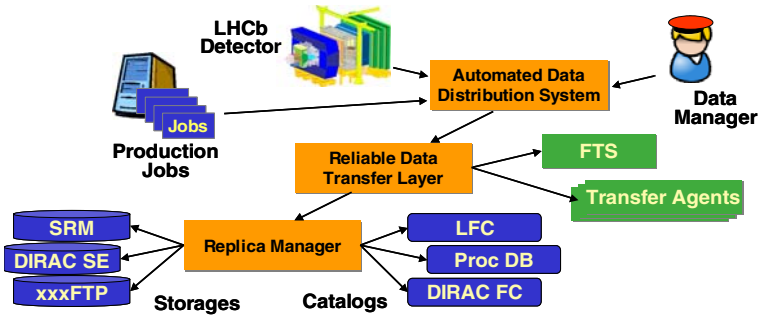

Figure 4 Data Management System components

The experience of managing large volumes of data in the distributed computing environment made it clear the necessity of thorough checking of the integrity of the data stored in various storage systems and the contents of the replica catalogs. Since the storage systems and catalogs are completely decoupled, these checks are not trivial. DIRAC provides several ways to carry out the checks [16]. The corresponding tools are centred around the Integrity Database which is accumulating reports of inconsistencies of the contents of the storage and catalogs as well as reports on the data access problems. The reports are sent by specialized agents which are systematically examining the storage and catalog name spaces but also by any other components which can fail to access some data. The contents of the Integrity Database are used in turn by agents attempting to recover the corrupted data or to restore the consistence of the catalogs. If the recovery operation can not be performed automatically, the control is passed to the human Data Manager.

6. Production Management System

The Workload and Data Management Systems are providing interfaces to the computing and storage resources and serve as a solid foundation for higher level applications. In the data production activity, managers have to deal with hundreds of thousands of jobs and data file replicas. To be efficient, they need a set of convenient tools to manage these large workloads. The DIRAC system provides various tools to cope with these tasks.

6.1. Workflow definitions

Each data production stage usually consists of several interdependent steps where the outputs of initial steps serve as input to subsequent steps. DIRAC provides means to describe workflows of any complexity as sequences of any number of simple operations. The operations are coded as Modules which can be assembled in Steps which typically correspond to invoking single applications. The Modules can be as simple as calling a user defined scripts or custom to execute a VO specific application. The Steps can be combined in complex Workflows executing several applications.

The Production Managers are using a library of predefined Modules, Steps and Workflows to either create new workflows or update the old ones. A special graphical Workflow Editor is provided to help this work.

6.2. Automatic Data Processing

Once the workflow is prepared, it is necessary to specify the corresponding jobs should be created, i.e. define the corresponding production. All the production definitions including the necessary statistics and input data to be processed are registered in the Processing Database. These definitions are used by a special agent to generate and submit jobs to the Workload Management System. It is important to

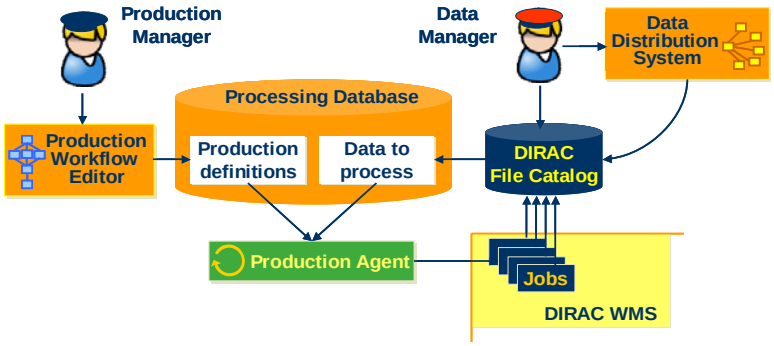

Figure 5 Production Management System components

note that the data to be processed are registered in the Processing Database at the same time and in the same operation as uploading the files. It means that the data is automatically available to define and submit the corresponding processing jobs as soon as it is stored in the Grid environment (Figure 5).

The Production Management System is capable to perform definition and submission of all the foreseen tasks in a fully automatic, data driven way. In the case of LHCb, the automatic processing chain will include the following tasks:

- Saving data from the on-line disk buffers to the persistent storage in the CERN Castor system;
- Making the second replica of the RAW data in the Tier-1 centres;
- Reconstruction of the RAW data at Tier-1 centres and distribution of the resulting DST data to other Tier-1 centres;
- Stripping (preselection) of the reconstructed data and distribution of the resulting analysis data to all the other Tier-1 centres for final analysis.

As a result, the user analysis of the experimental data should be possible with a minimal delay with respect to the moment of the data acquisition in all the Tier-1 centres according to the LHCb Computing Model [17].

6.3. Production experience and performance

The performance of the LHCb Production System based on the DIRAC project was evaluated during the last long Data Challenge 2006 (DC'06) which took place in 2006-2007. This was, in fact, a period of normal production with just few special test runs for new introduced Grid functionality. During this period there were about 1.5 million jobs executed. The number of concurrently executed jobs reached the peak value of 9750 together with several tens of thousands jobs waiting in the DIRAC central Task Queue. More than 120 distinct sites were involved mostly in the WLCG Computing Grid but also some standalone clusters. In the whole the Production System showed a very stable performance with an effective job success rate well above 90%. More details on the LHCb production experience can be found in [18].

The DIRAC DC'06 performance was achieved with all the central WMS services, including the job database, running on just one host situated at CERN. Since all the services are completely independent of each other, they can be deployed on different machines with even several instances of each services. Standard database performance enhancement solutions can be also applied to the central DIRAC database. Altogether, this shows a clear scalability potential of the system. Even in its present setup it is capable to serve a complete mid-range VO having access to heterogeneous distributed resources and more capacity can be easily added if necessary.

The DC'06 revealed a clear necessity of thorough monitoring of the sanity of the Grid resources provided by the sites. It turned out that often the site managers are not aware that the services under their responsibility are not functioning properly for a given VO. The SAM framework for monitoring the status of the Grid services had to be complemented by specific VO tests to help site managers to maintain their installations. In the case of LHCb these test jobs turned out to be crucial for successful functioning of the Production System. The peculiarity of the LHCb approach is that the DIRAC WMS is used to submit and monitor the jobs performing tests in the SAM framework with the results reported to the SAM Database [19]. This resulted in higher testing efficiency and in shorter service downtimes.

7. Conclusions and outlook

The DIRAC project once started as a simulation data production system now has evolved into a general purpose grid middleware which is covering all the major tasks performed in a distributed computing environment. The incremental development process ensured stable performance along with steadily increasing functionality. It allows now integrating different types of computing resources in a single system of up to 10'000 CPUs and there is a clear path of increasing its capacity due to a modular, service oriented architecture.

The system supports not only execution of individual jobs but also handling large sets of logically grouped tasks. Much attention is paid to the automation of well defined production activities in order to reduce the number of human operators. The general purpose functionality can be easily adapted to the needs of specific VOs by providing pluggable components developed in the common framework. The problems of stability of the resources in the ever changing grid environment are solved to a large extent with various failover mechanisms adding redundancy to the whole system. All these features allow positioning the DIRAC project as a general purpose Community Grid Solution for a mid-range to large VO having access to various heterogeneous computing resources.

Further DIRAC developments will be aimed at perfection of the current functionality which will be fully exploited during the upcoming series of the LHCb Dress Rehearsals to be held in 2008. This should scale will to the requirements of the real data taking period. Multiple improvements in the DIRAC interfaces are also foreseen to make it a viable alternative for other user communities as well.

8. Acknowledgements

The authors would like to acknowledge the European Marie Curie Fellowships program which allowed several young and dynamic developers to contribute to the DIRAC project.

References

[1] A.Tsaregorodtsev et al, DIRAC – The Distributed Data Production and Analysis for LHCb, Proceedings of the CHEP 2004 Conference.
A.Tsaregorodtsev et al, DIRAC, The LHCb Data Production and Distributed Analysis System, Proceedings of the CHEP 2006 Conference.
[2] T. Maeno, PanDA: Distributed production and distributed analysis system for ATLAS, Proceedings of the CHEP 2007 Conference
[3] L. Tuura et al, Scaling CMS data transfer system for LHC start-up, Proceedings of the CHEP 2007 Conference
[4] P. Saiz et al, AliEn2: the ALICE grid Environment, Proceedings of the CHEP 2007 Conference
[5] Y.Y.Li et al, Extension of the DIRAC workload-management system to allow use of distributed Windows resources, Proceedings of the CHEP 2007 Conference,
[6] R.Graciani Diaz, A.Casajus Ramo, DIRAC Agents and Services, Proceedings of the CHEP 2007 Conference
[7] A. Maier et al, Ganga - a job management and optimising tool, Proceedings of the CHEP 2007 Conference
[8] R.Graciani Diaz, A.Casajus Ramo, DIRAC Framework for Distributed Computing. Proceedings of the CHEP 2007 Conference
[9] R.Graciani Diaz, A.Casajus Ramo, DIRAC Security Infrastructure, Proceedings of the CHEP 2006 Conference.
[10] Pylons project, http://pylonshd.com .
[11] S.K.Paterson, A.Tsaregorodtsev, DIRAC Optimized Workload Management, Proceedings of the CHEP 2007 Conference
[12] A.Maier, S.K.Paterson, Distributed Data Analysis in LHCb, Proceedings of the CHEP 2007 Conference
[13] G.Castellani et al, DIRAC Job Prioritization and Fair Share in LHCb, Proceedings of the CHEP 2007 Conference
[14] D. Groep et al, glExec - gluing grid computing jobs to the Unix world, Proceedings of the CHEP 2007 Conference

I. Sfiligoi et al, Addressing the Pilot Security Problem With gLExec, Proceedings of the CHEP 2007 Conference
[15] A.C.Smith et al, DIRAC Reliable Data Management for LHCb, Proceedings of the CHEP 2007 Conference
[16] M. Bargiotti, A.C.Smith, DIRAC Data Management: consistency, integrity and coherence of data, Proceedings of the CHEP 2007 Conference
[17] A.C.Smith et al, DIRAC Data Production Management, Proceedings of the CHEP 2007 Conference
[18] R.Nandakumar et al, The LHCb Computing Data Challenge DC06, Proceedings of the CHEP 2007 Conference
[19] J.Closier et al, Ensuring GRID resource availability with the SAM framework in LHCb, Proceedings of the CHEP 2007 Conference