DIRAC - THE DISTRIBUTED MC PRODUCTION AND ANALYSIS FOR LHCB

A.Tsaregorodtsev, CPPM, Marseille, France
Ph. Charpentier, J. Closier, M. Frank, CERN, Geneva, Switzerland
V.Garonne, CPPM, Marseille, France
M.Witek, Henryk Niewodniczanski Institute of Nuclear Physics, Cracow, Poland
V.Romanovski, IHEP, Protvino, Russia
U.Egede, Imperial College, London, United Kingdom
V.Vagnoni, INFN, Bologna, Italy
I.Korolko, ITEP, Moscow, Russia
J. Blouw, MPI, Heidelberg, Germany
G.Kuznetsov, G.Patrick, RAL, Didcot, United Kingdom
M.Gandelman, UFRJ, Rio de Janeiro, Brazil
R.Graciani-Diaz, Universidad de Barcelona, Barcelona, Spain
R. Bernet, Universität Zürich, Zürich, Switzerland
N.Brook, University of Bristol, Bristol, United Kingdom
A.Pickford, University of Glasgow, Glasgow, Scotland
M.Tobin, University of Liverpool, Liverpool, United Kingdom
A.Saroka, I.Stokes-Rees*, University of Oxford, Oxford, United Kingdom
J.Saborido-Silva, M.Sanchez-Garcia, USC, Santiago de Compostela, Spain

Abstract

DIRAC is the LHCb distributed computing grid infrastructure for Monte Carlo (MC) production and analysis. Its architecture is based on a set of distributed collaborating services. The service decomposition broadly follows the CERN/ARDA-RTAG proposal, which should allow for the interchange of the EGEE/gLite and DIRAC components.

In this paper we give an overview of the DIRAC architecture, as well as the main design choices in its implementation.

The light nature and modular design of the DIRAC components allows its functionality to be easily extended to include new computing and storage elements or to handle new types of tasks. The DIRAC system already uses different types of computing resources - from single PC's to a variety of batch systems and to the Grid environment. In particular, the DIRAC interface to the LCG2 grid will be presented.

INTRODUCTION

LHCb is a high-energy physics experiment at the future CERN accelerator LHC that will start data taking in 2007 [1]. The amount of data (3.6 PB per year) and computing power needed to process them (over 29 MSI2K) require innovative solutions to building a distributed computing system capable to serve these needs.

There are a large number of projects that are aimed at providing grid computing systems capable of scaling to the level of overall requirements of the LHCb experiment. However, it would be naïve to think that all of the specific needs of LHCb will be covered by general purpose grid services. Therefore, we have started a project that should combine LHCb specific components with general purpose ones where it proves to be appropriate.

Background

The DIRAC project was started in autumn 2002 in order to provide LHCb with a distributed MC production system, which was sufficiently scalable and easy to operate [2]. At that point we already gained some experience with the EDG grid middleware [3]. This experience suggested that the centralized *push* scheduling paradigm is not sufficiently scalable to meet our needs and is dependent on a reliable, dynamic information system which is very difficult to build. Therefore, we have chosen to explore the *pull* job scheduling approach where the computing resources are "active" and demand workload whenever they are free. A light component, which we call the Agent, was developed to be deployed on the LHCb production sites. The Agents interact with the Central Production Database to get the jobs and update the job status. The system was successfully used in the Data Challenge of March-May 2003 validating the basic scheduling paradigm [4].

However, the DIRAC system was limited to just MC production activity and did not support the reprocessing or user analysis tasks. The latter activity necessitates a very performant and functional Data Management System (DMS), which is far more complex than any Workload Management System (WMS) in the grid environment. There were several projects aiming at providing DMS services [2,5,6]. So, it was decided not to build yet another DMS but to create an open system which will easily accommodate third party products. This was in the

*Marie Curie Training Site Fellow at CPPM, Marseille, France

spirit of the Open Grid Services Architecture (OGSA) elaborated by the Global Grid Forum and further specified as Open Grid Services Infrastructure (OGSI) [7]. At the same time an ARDA/RTAG working group at CERN came out with a proposal for the next generation grid middleware architecture based on a set of loosely coupled services [8]. These two proposals were the main sources of inspiration for the next stage of the DIRAC project.

DIRAC design principles

DIRAC is conceived as a light grid system. It should support a rapid development cycle to accommodate ever-evolving grid opportunities. It should be easy to deploy on various platforms. Updates, to fix bugs and/or introduce new functionalities, should be transparent or even automatic.

DIRAC is designed to be highly adaptable to the use of heterogeneous computing resources available to the LHCb Collaboration.

One of the main design goals is the simplicity of installation, configuring and operation of various services. This makes the threshold low for new sites to be incorporated into the system. Once installed and configured, the system should automate most of the tasks, which allows all the DIRAC resources to be easily managed by a single Production Manager.

The system should be robust and scale well to the computing needs of the LHCb Collaboration. This scale we roughly define for the moment as ~10^4 concurrent jobs, ~10^5 jobs in the queue, processing ~10^7 datasets.

ARCHITECTURE

DIRAC follows the paradigm of a Services Oriented Architecture (SOA).

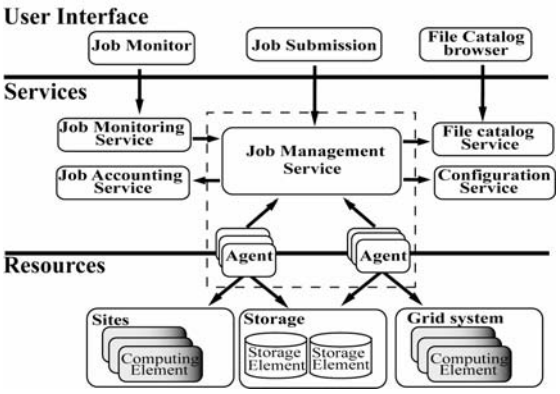

Figure 1 General view of the DIRAC architecture

Services

All the DIRAC services are written in Python and implemented as XML-RPC servers. The standard Python library provides a complete implementation of the XML-RPC protocol for both server and client part.

The more elaborated successor to XML-RPC, the SOAP protocol that is commonly used in building Web Services based applications is considered to be most suitable for building services based grid middleware. It was tried out and abandoned, at least for the time being, for several reasons. It is heavier and presents a non-significant overhead of parsing complex XML encoded messages. Different implementations of SOAP servers and clients suffer from interoperability problems.

A significant effort was taken to provide fault tolerant services [9]. The crucial services are duplicated to increase their availability. Many requests are repeated in case of failures to overcome network outages or service saturation. All the services are run using *runit* watchdog tool [10], which insures restarting in case of failure or on the machine reboot. It provides also many other useful features for service manipulation and debugging.

Computing Element

The Computing Element (CE) in DIRAC is an API abstracting common operations of job manipulation by computing batch systems. It also provides access to the state information of the computing resource, such as its capabilities, environment or occupancy. The API is implemented for various back-end batch systems: PBS, LSF, NQS, BQS, Sun Grid Engine, Condor or standalone PC. One particular case is access to the LCG grid, which is realized as a standard DIRAC CE.

Workload Management System

The Workload Management System (WMS) consists of the three main components: central Job Management Service (JMS), distributed Agents running close to DIRAC Computing Elements and Job Wrappers which are encapsulating the user job applications.

The JMS itself is a set of services that provide job reception from users, sorting jobs in task queues, serving jobs to the Agent requests, accumulating and serving job status information.

Agents continuously check the availability of the respective CE, pull jobs from the JMS and steer job execution on the local computing resource.
Job Wrappers prepare job execution on the Worker Node, get the job's input sandbox, send job status information to the JMS, and upload the job output sandbox.

More detailed description of the DIRAC WMS can be found in [11].

One interesting feature of the WMS is that services or users can communicate with Agents and Job Wrappers by means of an Instant Messaging (IM) protocol. In particular, the Jabber/XMPP protocol is used in DIRAC. It provides a reliable asynchronous bidirectional communication channel that can be used to monitor Agents and Jobs or even maintain interactive sessions with running jobs. The many interesting opportunities and advantages of this approach are discussed in [12].

Data Management System

The Data Management System (DMS) includes File Catalog Services, which keep track of available data sets and their replicas, as well as tools for data access and replication.

File Catalogs. The LHCb Bookkeeping Database (BD), which keeps track of the executed jobs and metadata of the available datasets (what is usually called the Metadata Catalog and Job Provenance Database) [13] also keeps information about the physical replicas of the files. A service was built as a front-end to this part of the BD, which allows usual File Catalog operations (registering files and their replicas, queries for file replicas for a given location, etc). However, this File Catalog implementation has rather limited functionality, and we looked for other solutions that can be imported as a service into the DIRAC.

We have chosen the File Catalog which is part of the AliEn project [5] because of its rich functionality and proven robust implementation. This catalog provides almost all the necessary features that we would expect: hierarchical structure following the file system paradigm, ACL-like control of access, possibility to store metadata associated with files. A front-end service was developed to provide access to the AliEn File Catalog functionality. This service maintains connection to the catalog and translates incoming queries into the AliEn UI commands.

Interfaces of both File Catalog services are identical, so that data management tools can use either of them (or both simultaneously) by just setting the appropriate configuration parameters.

Storage Element. The DIRAC Storage Element (SE) is a combination of a standard server, like gridftp, and information stored in the Configuration Service on how to access it. The Storage Element API provides a possibility to dynamically plug-in modules for transport protocols by which the SE is accessible as described in its configuration. Modules for most of the protocols are available: gsiftp, bbftp, sftp, ftp, http, rfio, direct file access. A special XML-RPC protocol allows transfer of relatively small files encapsulated into an XML-RPC message.

Reliable File Transfer Service. File transfer is a fragile operation because of potential network and storage hardware failures or errors in the associated software services. It is not unusual to loose output of a long job because of the failed data transfer that was never retried. Therefore, a Reliable File Transfer Service (RFTS), which allows retries of the failed operations until complete success is a vital part of the DMS.

In DIRAC the RFTS is constructed using the same building blocks as the WMS (Figure 2). Each site maintains a Request Database (RDB) of data operation requests. The requested operations can be data transfer, replication or registration in a File Catalog. One request can contain any number of operations. A special module called the Transfer Agent is continuously checking the contents of the RDB for outstanding requests and attempts to execute them. In case of failures, the request stays in the RDB for a later retry. Partially accomplished requests are modified to retry only undone operations.

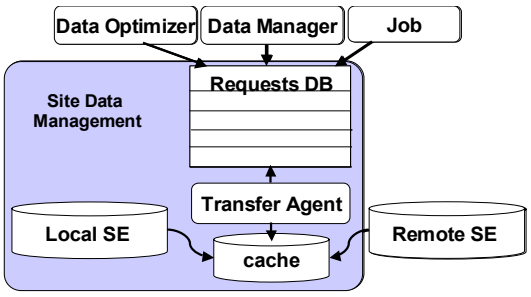

Figure 2 On-site data management tools

The RDB can be populated either by a regular production job executed on the site or by a special job the only purpose of which is to set a data transfer request. In both cases, the progress of the request execution can be monitored by the standard job monitoring tools provided by the WMS.

Application Software installation

One use of the DMS in DIRAC is the provision of application software for installation on production sites. The software is first packaged in binary distribution tar files. These files are treated as any other file in the DMS with possibly multiple available replicas. Standard tools are then used to locate and download the distribution files for local installation.

Information Services

The Configuration Service (CS) provides necessary configuration parameters to other services, Agents and Jobs to insure their collaborative work. The CS contains endpoints of all the DIRAC services and resources, their properties as well as policies for the current production run.

The Job Monitoring Service receives status information about the running jobs and provides it to the requests of users, for example through a dedicated Web Portal, or to other services. The Accounting Service accumulates statistics of the usage of the computing resources and generates reports that can be used to follow the production progress or to apply policies and quotas while job scheduling [14].

INTERFACING DIRAC TO LCG

In its present state, LCG already provides a large number of computing resources. There are different possible ways to exploit these resources.

The seemingly most straightforward way is to use the standard LCG provided middleware for job scheduling. However, these means are not yet reliable enough for large scale production, so other possibilities have to be explored. Another approach would be sending jobs directly to the LCG CE. This approach was tried out

successfully in our DC 2003 [4] to gain access to resources provided by the EDG testbed. However, in the recent Data Challenge 2004 yet another approach was realized.

The third approach is workload management with reservation of computing resources. We took advantage of having a light easily deployable "mobile" agent, which is part of the DIRAC native WMS. The jobs that are sent to the LCG Resource Broker (RB) are just executing a simple script, which downloads and installs a standard DIRAC agent. Since the only environment necessary for the agent to run is the Python interpreter, this is perfectly possible on all the LCG sites. The agent is configured to use the hosting Worker Node (WN) as a DIRAC CE. Once this is done, the WN is *reserved* for the DIRAC WMS and is effectively turned into a virtual DIRAC production site for the time of reservation. The reservation jobs are sent whenever there are waiting jobs in the DIRAC Task queue eligible to run on LCG.

There are many advantages to this approach. The agents running on the WN are ensuring the minimally sufficient environment (otherwise the agent itself would not run) before scheduling the real jobs. If the agent fails to start for whatever reason (failure of the RB, site misconfiguration, etc), the DIRAC Task Queue is not affected. This approach allowed the use of both LCG and non-LCG resources in a consistent way. In fact, the LCG middleware was used to dynamically deploy the DIRAC infrastructure upon the LCG resources providing a completely homogeneous system. The jobs running on LCG resources were still steered, monitored and accounted for in the same way and by the same services as other DIRAC jobs. This way allowed for substantial use of the LCG resources during the DC 2004 (over 5000 concurrent jobs at peak) with low effective failure rate [15].

The workload management with "resource reservation" opens interesting opportunities for optimization of job scheduling. While the resource is reserved, it can be used flexibly, for example, running multiple short jobs without repeated scheduling or participating in a coordinated parallel work together with other reserved resources. The latter mode of operation is suitable for running interactive data analysis sessions on the grid.

CONCLUSION

The DIRAC grid system of the LHCb experiment has evolved into a complete system covering the whole spectrum of the computing tasks to be performed in distributed computing environment. The system demonstrated good scalability properties for MC data production. We are preparing now for the massive data reprocessing and analysis phases.

The concept of the workload management which combines a central task repository with a network of dynamically distributed light agents proved to be very promising. Its efficiency was well demonstrated during the LHCb Data Challenge 2004. This concept opens new interesting opportunities to explore like peer-to-peer networks of agents redistributing the workload to adjust it to the immediate dynamic state of the computing grid.

The paradigm of workload management with advanced resource reservation allows reducing the impact of failed scheduling on the task repository, which simplifies significantly the task management.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the friendly help of administrators at all the sites where the DIRAC services were deployed insuring their stable operation. We are also very thankful to our contacts in the LCG deployment project, Flavia Donno and Roberto Santinelli, whose help in integrating DIRAC with LCG was essential.

REFERENCES

[1] LHCb, S. Amato et al, LHCb Technical Design Report, CERN-LHCC-2003-030, September 2003.
[2] DIRAC, http://dirac.cern.ch
[3] European DataGrid Project, http://eu-datagrid.web.cern.ch/eu-datagrid.
[4] A.Tsaregorodtsev et al, "DIRAC – Distributed Infrastructure with Remote Agent Control", In *Proceedings of CHEP2003 Conference*, La Jolla, April 2003.
[5] AliEn, http://alien.cern.ch
[6] Storage Resource Broker, http://www.npaci.edu/DICE/SRB
[7] GGF, Open Grid Services Architecture, http://www.globus.org/ogsa/
[8] "Architectural Roadmap towards Distributed Analysis ", CERN-LCG-2003-033, November 2003.
[9] I.Stokes-Rees et al, Developing LHCb Grid Software: Experiences and Advances, In *UK e-Science All Hands Meeting*, September 2004.
[10] G.Pape, Runit Service Supervision Toolkit, http://smarden.org/runit.
[11] V.Garonne, I.Stokes-Rees, A.Tsaregorodtsev, DIRAC Workload Management System, *these proceedings* [365].
[12] V.Garonne, I.Stokes-Rees, A.Tsaregorodtsev, Grid Information and Monitoring System Using XML-RPC and Instant Messaging for DIRAC, *these proceedings* [368].
[13] C.Cioffi et al, File-Metadata Management System for the LHCb Experiment, *these proceedings* [392].
[14] M.Sanchez Garcia et al, A Lightweight Monitoring and Accounting System for LHCb DC04 Production, *these proceedings* [388].
[15] J.Closier et al, Results of the LHCb Experiment Data Challenge 2004, *these \proceedings* [404].