DIRAC Pilot Framework and the DIRAC Workload Management System

Adrian Casajus
Universitat de Barcelona
E-mail: adria@ecm.ub.es

Ricardo Graciani
Universitat de Barcelona
E-mail: graciani@ecm.ub.es

Stuart Paterson
CERN
E-mail: stuart.paterson@cern.ch

Andrei Tsaregorodtsev
CPPM Marseille
E-mail: atsareg@in2p3.fr

on behalf of the LHCb DIRAC Team

Abstract.
DIRAC, the LHCb community Grid solution, has pioneered the use of pilot jobs in the Grid. Pilot Jobs provide a homogeneous interface to an heterogeneous set of computing resources. At the same time, Pilot Jobs allow to delay the scheduling decision to the last moment, thus taking into account the precise running conditions at the resource and last moment requests to the system.

The DIRAC Workload Management System provides one single scheduling mechanism for jobs with very different profiles. To achieve an overall optimisation, it organizes pending jobs in task queues, both for individual users and production activities. Task queues are created with jobs having similar requirements. Following the VO policy a priority is assigned to each task queue. Pilot submission and subsequent job matching are based on these priorities following a statistical approach.

1. Introduction

DIRAC [1] is the LHCb solution to interact with distributed computing resources. All detector data processing, Monte Carlo simulation and later end user access to these data for physics analysis is integrated in a single DIRAC system. For Workload Management tasks DIRAC

pioneered the pull scheduling mechanism within the LCG project. Besides it was the first LHC experiment framework to introduce the, now widely extended, concept of Pilot Jobs in the Grid. Pilot Jobs, implemented as a special kind of DIRAC Compute Element, provide DIRAC Workload Management System, WMS, with an overlay layer to any computing resource such that they are all presented in an uniform manner to a central WMS server for payload matching.

The use of Pilot Jobs allows DIRAC to apply late binding of resources to payload in a very natural manner. Finding at any time the pending task in the system that better matches the available computing resource grabbed by the pilot as well as centrally applying any VO policy concerning shares and priorities is very much simplified with the use of Pilot Jobs. For optimization of this late matching and better scalability, DIRAC WMS organizes pending Tasks into TaskQueues, formed by payloads with identical set of requirements. In section 2 DIRAC usage of Pilot Jobs is discussed in some detail, and the key elements for the DIRAC WMS scheduling design are described. Section 3 explains how this elements are put together. Some considerations about the usage of private and generic pilots are also discussed. In the final section 4 a summary is presented.

2. DIRAC Pilot Framework approach

DIRAC [1] implements a community-wide WMS able to manage simultaneously computing tasks for a given user community [1], where not all tasks are of the same nature. They can range from very high priority short payloads to check the current situation at the different computing resources available to the system, to chaotic end-user payloads. Of course, they also include community-wide low- and high-priority activities like Monte Carlo simulations and real time detector data processing. A more in-depth view of LHCb computing needs can be found at [3].

All these different payloads have different levels of priorities for LHCb as well as a variety of requirements that must be met for their successful completion. Eventually LHCb might also need to apply other policies like limiting the total amount of resources used by certain users or group of users, establishing some quota mechanism, or defining fair-share policies.

In order to optimize the access to the computing resources as well as to enforce the above community-wide policies, DIRAC, very early in its development phase, choose a solution with a single central server holding the complete queue of pending payloads, late binding of resources to payloads and the usage of the Pilot Job paradigm. For the implementation of this design choice three key components are necessary: the Pilot Jobs, the TaskQueues, and the TaskQueue Directors. They are described in some detail in the following.

2.1. Pilot Jobs

Pilot Jobs are nothing more than empty resource reservation containers that are sent to the available computing resources with the final aim of executing the most appropriate pending payload in the central WMS queue. There are several advantages in this approach with respect to a more classical payload pushing scheduling mechanisms:

- Pilots allow to perform basic sanity checks of the running environment prior to any binding with a given payload. In case problems are found thy can be reported to the DIRAC WMS framework for further actions. The user payload has not been assigned yet, so there is no impact from these problems onto the payloads.
- Pilots allow to create an overlay network that masks the central DIRAC WMS components from the heterogeneity of the underlying resources. Once the DIRAC Pilots are deployed on the Worker Node, they all present the same interface to DIRAC no matter if they are running in a large Computing Centre forming part of a given Grid infrastructure, like LHCb

[1] In this paper the community is the LHCb collaboration running the LHCb experiment at CERN [2]

Tier1's on LCG, or they are running on a set of Desktops that have been instrumented with DIRAC to execute certain payloads as background activity.

- Pilots allow an effective implementation of the pull scheduling paradigm. Once they are safely running in the final computing resource, they contact central WMS servers for a late binding of the resource to the payload. This overcomes many intrinsic problems of the push mechanism when running on a world-wide distributed environment with all the unforeseeable delays in the delivery of the pushed payload to the final executing resource, as well as with those issues derived from an exact knowledge of the current and near future local situation at each and every target resource, an impossible pre-requisite for a proper functionality of a such type of scheduling when attempted in this kind of environments. When presenting their grabbed resources Pilots produce the most accurate picture of those currently available.

Every DIRAC Pilot Job performs an *in situ* DIRAC installation including a full download of the most current version of the configuration. In this fresh installation and after checking the working conditions (exact location where execution is taking place, available disk space, memory, cpu, available Grid environment, running platform, ...) a DIRAC Job Agent is executed. This Job Agent is responsible for placing the payload request to the central DIRAC WMS server and for the later execution of the received payload.

To provide a common execution environment for all payloads, the Job Agent instantiates a Job Wrapper object, last responsible of the payload execution. At the same time it also instantiates a Watchdog to monitor the proper behaviour of the Wrapper. The watchdog checks periodically the situation of the wrapper, takes actions in case the disk or available cpu is about to be exhausted or the payload stalls, and reports to the central WMS. It can also execute management commands received from the central WMS, like killing the payload. The wrapper retrieves the input sandbox, checks availability of required input data and software, executes the payload, reports success or failure of the execution, and finally uploads output sandbox and output data if required.

This mechanism allows the users to concentrate their efforts in the definition of the real payload since DIRAC takes care of the burden of all these extra steps that make the difference between a local and a remote execution. The DIRAC wrapper takes care in an uniform way of all these details providing the user with a well defined and common execution environment across different computing resources.

In order to efficiently determine the most appropriate payload to be handled for execution to a Pilot Job at any given time, pending tasks are organized into TaskQueues.

2.2. TaskQueues

As soon as payloads are introduced into the DIRAC WMS system and after a consistency check and a proper resolution of possible target computing resources in case there are Input Data requirements new tasks are organized into TaskQueues. TaskQueues are nothing more than sorted groups of payloads waiting for execution with identical requirements to the possible Pilot Job requesting a match. Among these requirements are the user identity submitting the task, the amount of computing power requested, the computing platform required (if any), the eligible or banned sites for the payload (if necessary),... Each task in a TaskQueue may get a priority assigned by the user at submission time, a number from 0 to 10 reflecting in linear scale how many times more likely to be match does the user wants this particular task to be with respect to his/her other pending tasks in the same TaskQueue. 0 and 10 are handled specially, and they are internally given 10^{-5} and 10^{5} weights respectively.

For each user group[2] LHCb assigns a certain overall group priority that is equally divided

[2] A user may belong to more than one DIRAC group but at any given time he/she can only act in one particular

among all users currently having TaskQueues defined in the system. If more than one TaskQueue is present for a certain user/group combination, due to a difference in any other of the relevant requirements among the pending tasks, the priority assigned to each of these TaskQueues is proportional to the sum of the user defined priorities for all the tasks in the TaskQueue.

A new TaskQueue is created when a new payload that does not fit in any of the existing ones enters the system. Analogously, when all pending task have been removed by Pilot Jobs from a certain TaskQueue, the TaskQueue gets deleted. Priorities for all TaskQueues get re-evaluated every time a TaskQueue is created or deleted as well as periodically every two minutes to take into account the effect form newly introduced and matched payloads.

The matching of the Pilot provided resources to an specific pending payload is done in two steps. First an appropriate TaskQueue is found and then a pending payload is selected and removed from the TaskQueue. In both cases the priorities, of the TaskQueue or of the payloads, are used as weights to give each matching candidate an statistical probability. Like the rest of the DIRAC WMS components, the code is from the design thought to run in a multi-threaded environment to achieve the best scalability. To minimize the likelihood of a collision between two or more matching requests executing simultaneously, a randomization mechanism is used when applying the weights. The oldest payloads for the selected TaskQueue and priority are preselected and removal is attempted in an atomic operation. In the unlikely case of a collision, ie two request executing simultaneously and eventually preselecting the same payload, the second removal fails and a different payload is preselected. In case the preselected payload was the last one pending in the TaskQueue and a collision occurs other matching TaskQueues are considered following the same logic.

The TaskQueue mechanism allows DIRAC WMS to produce a best match in less than 0.01 second even in the case of one million pending payloads distributed over more than ten thousand different TaskQueue. This figure is significantly bellow the 0.1-0.2 second overhead due to networking, including the full ssl handshake and encryption mechanism.

The final actor in the picture is the TaskQueue Director, responsible for populating the computing resources with appropriate Pilots Jobs able to match the pending payloads.

2.3. DIRAC TaskQueue Directors

TaskQueue Director Agents is the way chosen by DIRAC to populate the available computing resources, the worker nodes, with Pilot Jobs. As any other DIRAC Agent, see [5], Directors work in cycles repeating the same logic on each iteration. This logic is the following:

(i) Query the central WMS server to get a list of TaskQueues. If the Director is populating a complete general resource like the LCG, no requirements are set on this query. On the other hand if the Director is in front of a dedicated facility, for instance a local cluster intended just for Monte Carlo simulation, this requirement is passed in the query. In this way the Director gets a complete picture of the pending activities currently eligible for execution.

(ii) Based both on the priority of the TaskQueues and the total number of pending tasks on each of them, the Director attempts the submission of a configurable number of Pilot Jobs per iteration. For each TaskQueue the availability of matching resources is checked by looking at the number of free slots on the queues or alternatively looking for a relatively small locally pending queue of Pilots. The number of previously submitted Pilots that have so far not matched any payload are also evaluated.

(iii) All this information is put together, the number of Pilot Jobs to be submitted for the TaskQueue is evaluated and submission is attempted. The TaskQueue definition is rewritten in the language required by the underlying system, ie Requirements in a gLite JDL file. If successful, the pilots are registered in the system for later reference.

group that is embedded in the proxy by means of special DIRAC extension (see [4] for further details)

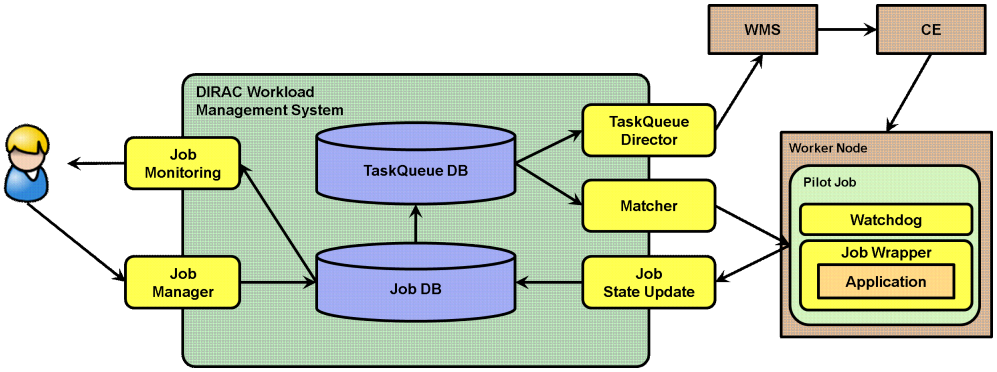

Figure 1. DIRAC Workload Management System.

To handle the inefficiencies intrinsic to the underlying distributed and varying resources, each Director Agent is accompanied by a corresponding Pilot Status Agent. This Agent monitors the submitted Pilots flagging them in case of failures and declaring them stalled if they have not managed to execute after a certain amount of time. Given the protections included in the Directors to keep a reasonable rate of Pilots submitted to pending tasks, it is essential to prevent the system from stalling if a certain resource misbehaves, failing all Pilots received or keeping them in the local queues for an unreasonable amount of time. Any Pilot Job receiving a payload for execution is immediately flagged in the central WMS server.

3. The full DIRAC Workload Management System picture

Using the above components DIRAC WMS can be described using the picture from figure 1. On the left side, the user initiates the whole procedure by submitting a new payload. For the sake of simplicity, only one Director submitting to WLCG via gLite WMS servers is shown. After submission, the payload description is inserted into the DIRAC job repository, the JobDB. This description is used to add an entry in the TaskQueueDB, adding the payload to an existing TaskQueue if a matching one is found or creating a new TaskQueue otherwise. Based on the complete list of pending payloads, Director Agents are used to populate the available computing resources with Pilot Jobs. After deploying DIRAC system and checking the local environment on the WN, Pilots request a payload to the central DIRAC WMS server. The payload is finally executed inside a DIRAC Wrapper under the supervision of a DIRAC Watchdog.

The use of central TaskQueues together with the Pilot Job approach provide the user payload with a transparent and efficient access to the underlying computing resources. At the same time it allows to take into account global policies for the whole user community, like priorities, fair shares and quota mechanisms. With all these capabilities DIRAC allows an overall optimization of the resource usage for the complete user community, including a large variety of different payloads and use patterns. The resulting system has shown very nice scalability, being able to either fill up all the available resources for LHCb (over 15 K cpu cores) with 1 day long Monte Carlo simulation jobs or managing job rates well in excess of 1500 job/hour for short users data analysis payloads, see [6].

3.1. Private and Generic Pilots

The Pilot Job concept has recently covered a new evolution step. In its initial implementation Pilot jobs include as part of their description the full identity of the user to which the pending payloads in the corresponding TaskQueue belong. This pilots are what it is called “Private” Pilot Jobs, and once running on the computing resource are only able to match payload from this identity. The new “Generic” Pilot Job concept is now fully supported in DIRAC. For computing

resources accepting this type of pilots, DIRAC submits Pilots using the same strategy defined above with a special user/group combination: the DIRAC pilot group (mapped in the WLCG world to the VOMS extension `lhcb:/lhcb/Role=pilot`). This special identity, although it does not own any payload, is allowed to match any pending task in the system irrespectively of the user/group identity it might belong to, just following the priorities, quotas and/or shares globally defined. Once the payload is matched and the identity of the owner is known, a delegated limited proxy is requested and payload is fully executed with this identity.

DIRAC supports the use of gLexec mechanism, currently being deployed at the WLCG sites, to switch from the pilot to the payload identity prior to the DIRAC Job Wrapper execution. At the same time it also provides the same functionality for sites where this new mechanism is not yet installed by means of properly setting the environment prior to the DIRAC Job Wrapper execution. The actual identity switch provided by gLexec is site configurable and may range from a simple environment change like the one DIRAC is doing, to a full identity switch to a complete new execution environment for the payload with different local unix user and group ids, and new working directory fully isolated from the original pilot environment.

4. Summary

DIRAC, the LHCb solution for managing all computing activities on the Grid, includes a full featured Workload Management System based on the Pilot Job paradigm to implement a pull scheduling mechanism with late binding of resources to payloads. This solution provides an efficient system to apply any priority, fair-share and quota system decided by LHCb as a whole without need of any further tools or effort at the site, middleware or Grid levels to enforce them.

At the same time, this solution provides a layer in between the computing resources and both the DIRAC WMS and the user payload that gives uniform and transparent access to the resources. This simplifies the task of the Workload Management and shields the user applications from the underlying complexity inherent to a world-wide distributed computing environment.

References

[1] Tsaregorodtsev A *et al.* 2003 *Computing in High-Energy Physics and Nuclear Physics 2003*
[2] Antunes-Nobrega R *et al.* (LHCb) CERN-LHCC-2003-030
[3] Antunes-Nobrega R *et al.* (LHCb) CERN-LHCC-2005-019
[4] Casajus A and Graciani R 2009 *Computing in High-Energy Physics and Nuclear Physics 2009*
[5] Casajus A and Graciani R 2007 *Computing in High-Energy Physics and Nuclear Physics 2007*
[6] Paterson S and Closier J 2009 *Computing in High-Energy Physics and Nuclear Physics 2009*