DIRAC universal pilots

To cite this article: F Stagni et al 2017 J. Phys.: Conf. Ser. 898 092024

View the article online for updates and enhancements.

Related content

- Integration of Cloud resources in the LHCb Distributed Computing
  Mario Úbeda García, Víctor Méndez Muñoz, Federico Stagni et al.

- Monitoring of IaaS and scientific applications on the Cloud using the Elasticsearch ecosystem
  S Bagnasco, D Berzano, A Guarise et al.

- Integrating PROOF Analysis in Cloud and Batch Clusters
  Ana Y Rodríguez-Marrero, Isidro González Caballero, Alberto Cuesta Noriega et al.
**DIRAC universal pilots**

F Stagni\(^1\), A McNab\(^3\), C Luzzi\(^1\), W Krzemien\(^2\)

\(^1\)PH Department, CH-1211 Geneva 23 Switzerland
\(^2\)NCBJ: National Centre for Nuclear Research, 05-400 Otwock, Swierk, Poland
\(^3\)The University of Manchester, Oxford Road, Mancherster, M13 9PL, UK

On behalf of the DIRAC consortium

E-mail: federico.stagni@cern.ch

**Abstract.** In the last few years, new types of computing models, such as IAAS (Infrastructure as a Service) and IAAC (Infrastructure as a Client), gained popularity. New resources may come as part of pledged resources, while others are in the form of opportunistic ones. Most but not all of these new infrastructures are based on virtualization techniques. In addition, some of them, present opportunities for multi-processor computing slots to the users. Virtual Organizations are therefore facing heterogeneity of the available resources and the use of an Interware software like DIRAC to provide the transparent, uniform interface has become essential. The transparent access to the underlying resources is realized by implementing the pilot model. DIRAC’s newest generation of generic pilots (the so-called Pilots 2.0) are the “pilots for all the skies”, and have been successfully released in production more than a year ago. They use a plugin mechanism that makes them easily adaptable. Pilots 2.0 have been used for fetching and running jobs on every type of resource, being it a Worker Node (WN) behind a CREAM/ARC/HTCondor/DIRAC Computing element, a Virtual Machine running on IaaC infrastructures like Vac or BOINC, on IaaS cloud resources managed by Vcycle, the LHCb High Level Trigger farm nodes, and any type of opportunistic computing resource. Make a machine a "Pilot Machine", and all diversities between them will disappear. This contribution describes how pilots are made suitable for different resources, and the recent steps taken towards a fully unified framework, including monitoring. Also, the cases of multi-processor computing slots either on real or virtual machines, with the whole node or a partition of it, is discussed.

1. Introduction

This paper illustrates the recent developments of DIRAC\(^1\)[2] Pilots\(^3\). DIRAC is a community Grid solution, developed in Python, that offers powerful job submission functionalities, and a developer-friendly way to create services, agents, and executors, together with the integration of external components.

DIRAC introduced the concept of pilot jobs more than ten years ago, to overcome instabilities and issues with the previously used “push” model. A profound rewriting happened in 2014, leading to so-called “Dirac pilots 2.0”. This paper illustrates how this new generation of pilots is used for hiding diversities among different types of computing resources. It also gives insights of the ongoing development for the new generation of DIRAC pilots, which we dubbed “Pilots 3.0”, which are already partially used in a production environment.

This paper is organized as following: section 2 gives an overview of modern distributed computing resources, section 3 explains the DIRAC Workload Management System and the role
of pilot jobs in exploiting all types of computing resources, and section 4 explains the recent developments of DIRAC pilots 3.0. Finally, a summary is given in section 5.

2. Distributed computing in 2017

The grid, as a distributed computing concept, was born to face large high-throughput computing requirements, coming primarily from HEP (High Energy Physics) experiments. The GRID concepts became a number of technologies, thanks to middleware initiatives driven by HEP experiments themselves, and by large, public-funded software projects. As of today, experiments use a variety of technologies, including clouds, vacuum sites [4], and volunteers[5].

Within the last years, many LHC and non-LHC communities started adding new types of computing resources to their grids. While for LHC VOs (Virtual Organizations) WLCG (Worldwide LHC Computing Grid) resources still provide the majority of available computing power, this is certainly not true for non-LHC communities. Nowadays DIRAC is used by dozens of VOs with a variety of computing models and resources among them. The most common resources used by the LHC communities are:

- Standard CEs (Computing Elements) provide a uniform front end service to authorize the jobs and submit them to the local batch system. Services like CREAM CEs, with direct pilot jobs submission, but also other CEs, like ARC (NorduGrid) CEs, which are a popular choice among sites, DIRAC CEs.
- WNs (Worker Nodes) that are instantiated Virtual Machines provided in the frame of the “Vacuum model” that VAC [4] exploits, or using existing cloud systems.
- Various forms of opportunistic computing, for example:
  - most LHC experiments have used the High Level Trigger (HLT) farms during the long shutdown, while LHC was not taking data, for running grid jobs.
  - Many have integrated (HPC - High Performance Computing) opportunistic sites, although many of them are using HPC sites as standard grid farms.
  - Volunteer computing, usually achieved via BOINC [6].

In other terms, the Grid “push” model is still in use, but is complemented by IAAS (Infrastructure as a Service) and IAAC (Infrastructure as a Client) models. A typical case of IAAS is represented by cloud systems, while vacuum models can be considered as embracing the IAAC models. Traditional grid sites are also presenting heterogeneous resources due to multiple hardware generations being supported. We can therefore say that the grid is not anymore “The Grid”, as today’s experiments’ distributed systems are made of heterogeneous resources.

Communities are hungry for CPU cycles. If we restrict for a moment our vision to LHC experiments, and we analyze the amount of CPU cycles they used in the last year, we can notice that all of them have consumed more CPU-hours than those official reserved (pledged) to them by WLCG (the Worldwide LHC Computing Grid). Each community found ways to exploit non-reserved CPUs, often on not officially supported resources and computing elements. Such resources may be private to the experiment (e.g. the “online” computing farm - often simply called “High Level Trigger” farm) or public; resources may sometimes be donated free of charges (as is the case for volunteer computing), or not (public commercial cloud providers).

Integrating non-grid resources is an ongoing activity for all communities that have been using WLCG in the past, and still do. Software like DIRAC aims to make this job easy.

3. DIRAC pilots

DIRAC was initially developed inside the LHCB collaboration, as a LHCB-specific project, but since 2010 the LHCB-specific code resides in the LHCBDIRAC[7] extension while DIRAC is
VO-agnostic. Since then, several communities have been using DIRAC as a software of choice for managing their distributed computing requirements, often providing their own extensions.

Being a community Grid solution used by several communities, DIRAC can interface with many resource types, and with many providers. Resource types are, for example, a computing element (CE), a catalog, or a storage element. DIRAC provides a layer for interfacing with many CE types, and hide their differences.

Distributed computing software like DIRAC or PanDA\cite{8} implement, among other functions, a “Workload Management System” (WMS) to submit and monitor jobs on the Grid, or clouds, or whatever other computing resource the community has at its disposal. The WMS is also used for submitting pilot jobs to the CEs.

3.1. Pilot jobs
The Grid model was initially conceived as a “push” model, where jobs were submitted from a queue of jobs, managed by each and every experiment in an independent way through their WMS software. The “push” model proved to be inefficient and error prone. To face these issues DIRAC introduced, back in 2006, the so-called pilot jobs, which are not “jobs”, but, instead, scripts that, once run, look for “jobs”:\this action is often called “matching” a job. A pilot job that fails prior to having matched a job causes no particular troubles. The advantages of the pilot job concept are now well established: pilots are not only increasing the aggregate users’ job throughput efficiency, but also helping to manage the heterogeneous computing resources presenting them to the central services in a uniform coherent way. Each LHC VO has, since then, moved to the pilots model, which is now a standard solution.

In an ideal situation, VOs have an unlimited amount of WNs at their disposal, and all these nodes are set up in a way that is convenient for the VO itself. The reality is that WNs are often shared among several VOs and, at least before virtualization became popular, all presented a stock operating system. So, another reason why pilot jobs have been invented is because VOs wanted their “own” machines for their Grid jobs, or, at least, to privatize them for some hours. In a way, pilot jobs became the first way of privatizing grid WNs, because, at a minimum, that had to:

- set the environment
- install the middleware

With “middleware” we mean DIRAC itself. The first version of pilots were introduced in the paper \cite{9}, that was presented at CHEP 2006. The second generation of DIRAC pilots, that we dubbed “Pilots 2.0” was instead presented in paper \cite{3} and its concept is visualized in figure 1.

For pilots 2.0 it has also been wrote that:

- A pilot is what creates the possibility to run jobs on a worker node.
- A pilot 2.0 is a standalone script
- Can be sent, as a “pilot job”, to all “Grid” CE types
- Can be run as part of the contextualization of a Virtual Machine, or whatever machine
- Can run on almost every computing resource, provided that:
  - Python 2.6+ is installed on the WN (pilots 2.0 are not coded for python 3)
  - It hosts an Operating System onto which DIRAC can be installed

Pilots 2.0 have been implemented using a command pattern. Each command is realizing an atomic function. A toolbox of pilot capabilities (“pilot commands”) is available to each and every pilot. DIRAC pilots can also be extended by communities that want to provide their own commands.
Figure 1. Pilots 2.0 form an overlay layer hiding the underlying diversity: pilots should be started on every Worker Node, in order to match a Job.

Figure 2. Pilots and the pilots commands.

3.2. Pilots 2.0 as overlay layer
A DIRAC pilot role is to install and configure the full DIRAC environment on the WN. After the successful installation, the DIRAC pilot invokes a special agent called “JobAgent”. The
“JobAgent” will try and fetch (match, in fact) a job from the central DIRAC jobs queue, which resides in the DIRAC WMS. A DIRAC pilot should be able to run on each and every computing resource type. Therefore, effectively the set of running pilots form an overlay layer hiding the underlying diversity of resources.

Pilots 2.0 have been very successful among DIRAC communities, with every installation adopting them. Some VO extended the DIRAC base pilots with their own commands, including LHCb (DIRAC’s main user, maintainer and developer), which uses by default CVMFS[10] (CERN Virtual Machine File System) for installing DIRAC, instead of fetching a DIRAC tarball. CVMFS is a distributed read only File System based on FUSE, optimized for the distribution of software.

Staying with the LHCb example, Pilots 2.0 have been used for each and every computing resource since 2014 (DIRAC release v6r12). This version could be used also for IAAS and IAAC resources, including “vacuum” types of resources like Vac[4] and clouds: pilots 2.0 became the core of the Virtual Machines contextualization scripts. The simplicity of DIRAC pilots 2.0 with respect to other pilots used by other LHC experiments became clear with the CERN cloud extension usage: as shown in figure 3 LHCb DIRAC pilots could be run directly within the instantiated VMs without the need for a batch service. This is in sharp contrast with what other VOs not using DIRAC need to do (see figure 3).

Figure 3. The picture shows how the the CERN cloud extension was exploited, by the WLCG experiment in 2015. LHCb DIRAC’s approach is visually simpler than the other shown experiments.

4. Latest DIRAC pilots developments: pilots 3.0
The bulk of changes done for pilots 2.0 is serving as a base for the new generation of DIRAC pilots, which we dubbed (of course!) “pilots 3.0”. Development and testing of the next generation is underway, and includes more general-purpose pilot commands.
4.1. The DIRAC pilots repository

DIRAC pilots’ code have historically always resided together with DIRAC’s code. It was recognized that coding for the pilot is rather different from coding for DIRAC, as it uses a different PYTHONPATH, different dependencies, different python support, and that the pilots’ code may evolve independently from DIRAC.

So, as the first action we decoupled the development of pilots from the development of the bulk of DIRAC. We then created a “Pilot” repository, which, contrary to all other repositories on the GitHub hosting, is not a DIRAC extension.

This move also gave us the possibility to make selective tests, using a technique that can be regarded as exposure control: we can use the pilots 3.0 in production, prior to full tests and even prior to releasing, by enabling it to run on a specific subset of computing resources. This is happening already, and in fact, as of today, about 10% of LHCb computing nodes run pilots 3.0 code.

4.2. Monitoring system

The scalable monitoring system based on the Message Queue (MQ) architecture is foreseen to be included as an option to the Pilot code. The monitoring system running as early as the pilot scripts itself, before any job matching, will be able to provide information about the WN environment and possible errors occurring during the installation and configuration phases, by sending logging messages to the dedicated servers.

The communication channels between the pilot and the monitoring server can be independent from the DIRAC architecture (DIRAC-agnostic) e.g. the MQ server can be a RabbitMQ or ActiveMQ instance without any knowledge of the DIRAC environment. It makes the solution more flexible and scalable. The transmission security is provided by the digital SSL certificates. The prototype model of the so-called Pilot Logger has been developed and tested using the RabbitMQ logging system.

4.3. Resources specialization

The list of commands that a pilot runs became, from Pilots 3.0, easily configurable: Pilots 3.0 can be configured to run a set of command, in whatever order, with any configuration. We have also given the possibility to easily activate and deactivate commands based on the type of WN, or CE, that the pilot is running on. The list of commands is configured at runtime, or in other words discovered only when the pilot starts. Some commands may be needed only for volunteer computing resources, or only for a specific setup, and so on.

4.4. The bootstrap issue

The so-called “bootstrap issue” is the name we gave to the process of starting a pilot. While for traditional Grid CEs the pilot jobs are sent to what is simply described as a service in front of a batch queueing system, a pilot, on a VM, starts blind: in the vacuum. So, pilot wrappers need to be supplied with few info:

- Where to get the pilot script(s)
- Resource names: Site, CE, Queue

Figure 3 shows that already pilots 2.0 were solving the bootstrap issue. The advantage is clear: if pilots can be started “in the vacuum”, we eliminate the need for a queuing system for starting the pilots themselves.
4.5. **SAM tests via the pilots**

We create a new pilot command for running SAM[11] (Service Availability Monitoring) tests at sites, and report the outcomes required by WLCG. This approach allows SAM tests to be run on resources (such as VMs created spontaneously in the Vacuum model) that have been previously inaccessible to the WLCG site testing framework. This example demonstrates the universal nature of the DIRAC pilots, as the same framework is made available on all flavours of resources.

5. **Summary and prospects**

Pilots 2.0 have been available to all DIRAC communities as of DIRAC v6r12, and have been used by all DIRAC installations as a way for running DIRAC jobs. They have also been adopted for running jobs on non-grid computing resources. Their flexibility made it possible to easily extend them.

The Pilots code is still actively developed, now as a functionally independent system from the DIRAC core. A new generation of pilots is underway, and already in use in production via exposure control.

**References**


