Pilots 2.0: DIRAC pilots for all the skies

F Stagni[1], A Tsaregorodtsev[2], A McNab[3], C Luzzi[1]

[1]PH Department, CH-1211 Geneva 23 Switzerland
[2]C.P.P.M - 163, avenue de Luminy - Case 902 - 13288 Marseille cedex 09, France
[3]The University of Manchester, Oxford Road, Mancherster, M13 9PL, UK

On behalf of the DIRAC consortium and the LHCb Collaboration

E-mail: `federico.stagni@cern.ch`

Abstract. In the last few years, new types of computing infrastructures, 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 opportunistic. Most of these new infrastructures are based on virtualization techniques. Meanwhile, some concepts, such as distributed queues, lost appeal, while still supporting a vast amount of resources. Virtual Organizations are therefore facing heterogeneity of the available resources and the use of an Interware software like DIRAC to hide the diversity of underlying resources has become essential. The DIRAC WMS is based on the concept of pilot jobs that was introduced back in 2004. A pilot is what creates the possibility to run jobs on a worker node. Within DIRAC, we developed a new generation of pilot jobs, that we dubbed Pilots 2.0. Pilots 2.0 are not tied to a specific infrastructure; rather they are generic, fully configurable and extendible pilots. A Pilot 2.0 can be sent, as a script to be run, or it can be fetched from a remote location. A pilot 2.0 can run on every computing resource, e.g.: on CREAM Computing elements, on DIRAC Computing elements, on Virtual Machines as part of the contextualization script, or IAAC resources, provided that these machines are properly configured, hiding all the details of the Worker Nodes (WNs) infrastructure. Pilots 2.0 can be generated server and client side. Pilots 2.0 are the "pilots to fly in all the skies", aiming at easy use of computing power, in whatever form it is presented. Another aim is the unification and simplification of the monitoring infrastructure for all kinds of computing resources, by using pilots as a network of distributed sensors coordinated by a central resource monitoring system. Pilots 2.0 have been developed using the command pattern. VOs using DIRAC can tune pilots 2.0 as they need, and extend or replace each and every pilot command in an easy way. In this paper we describe how Pilots 2.0 work with distributed and heterogeneous resources providing the necessary abstraction to deal with different kind of computing resources.

1. Introduction

This paper illustrates a recent development of DIRAC[1][2]. 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.

Services expose an extended Remote Process Call (RPC) and Data Transfer (DT) implementation. Agents are a stateless light-weight component, comparable to a cron-job. Executors are designed around two components: the Mind knows how to retrieve, store and dispatch tasks, and Executors are the working processes that know what to do depending on the task type. Other types of components that are not developed within DIRAC can be included, and managed as well. DIRAC introduced the concept of pilot jobs already ten years ago. This paper illustrates a new generation of pilots.

Being a community Grid solution, 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. DIRAC provides a layer for interfacing with many CE types, and different catalog and storage types. It also gives the opportunity to instantiate DIRAC types of sites, CEs, storage elements (SE) or catalogs.

DIRAC has been initially developed inside the LHCb[3] collaboration, as a LHCb-specific project, but since 2010 the LHCb-specific code resides in the LHCbDirac extension while DIRAC is VO-agnostic. In this way, other VOs, like Belle II [4], or ILC/LCD [5] have developed their custom extensions to DIRAC.

Section 2 goes through a brief history of grid computing resources, section 3 explains what pilots are, section 4 shows what grid in 2015 means, section 5 explains pilots 2.0 requirements, and design, and section 6 shows how LHCb extended pilots 2.0. Finally, a summary is given in section 7.

2. Brief history of grid computing resources, from a LHCb perspective

The grid, as a distributed computing concept, was born to face large high-throughtput computing requirements, coming primarly from HEP experiments. The concepts became a number of technologies, thanks to middleware initiatives driven by HEP experiments themselves, and by large, public-funded software projects. LHC experiments support was, and still is, assured by the WLCG (Worldwide LHC Computing Grid). WLCG was and still is vastly referred as *"The Grid"*. As of today, experiments use a variety of technologies, but only few years ago most, if not all, of the grid computing resources an LHC experiment could use was limited to WLCG supported resources.

The Grid model initially advertized was a push model, where jobs were submitted from a *queue* of jobs (managed by each and every experiment in an independent way) to:

(i) the (W)LCG resource broker (RB, also known as WMS - Workload Management System), which we can consider as a complex *queueing* system used for dispatching jobs to

(ii) Computing Elements (CEs), managed by the destination sites. A CE has, at a minimum, to implement a *queue* for dispatching jobs to

(iii) batch *queues* (like LSF, or SLURM, just to name a few) that handle the submission to Worker Nodes (WNs), where the jobs are finally running.

So, if a job was lucky enough to survive to the 4 queues it had to go through, it could finally run and hopefully produce correct results.

This model proved to be inefficient. First of all, WLCG inefficiencies were exposed to end users. Also, in case of complex JDLs (Job Descriptions), experiments could experience high load on the WLCG brokers. To face these issues, DIRAC, that was first developed by LHCb, introduced the so-called pilot jobs. An explanation of what is a pilot can be found in the next section.

Within few years, CEs implementations evolved from being "LCG CEs", to "CREAM CEs". From an experiment's point of view, the biggest change was represented by the fact that jobs could be pushed to CREAM CEs without needing to go through the LCG RBs, thus removing an intermediate queue.

3. What was, and what still is, a pilot

As explained in the previos section, pilot jobs came to solve the practical issue of avoiding inefficiencies. A "pilot job" is not a "job": instead, it is a script that, once run, will look for a "job": this action is often called "matching" a job. So, a pilot job that fails prior to having matched a job causes no particular troubles. End users care about seeing their jobs running, and the fact that pilot jobs might fail does not interest them. The advantages of the pilot job concept are now well established. The pilots are not only increasing the efficiency of the user jobs but also help to manage the heterogeneous computing resources presenting them to the central services in a uniform coherent way.

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 came 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

Where, with "middleware", we mean DIRAC itself. DIRAC, or its extensions, was installed in every job via the pilot jobs by getting from a webserver a tarball containing a full, pre-compiled version of DIRAC. The application software used by the jobs was installed with SAM[] jobs. In the paper [6], that was presented at CHEP 2006, the term pilot does not apper yet. Instead, it was said that "agents form an overlay layer hiding the underlying diversity" and as thus embracing the network paradigm.

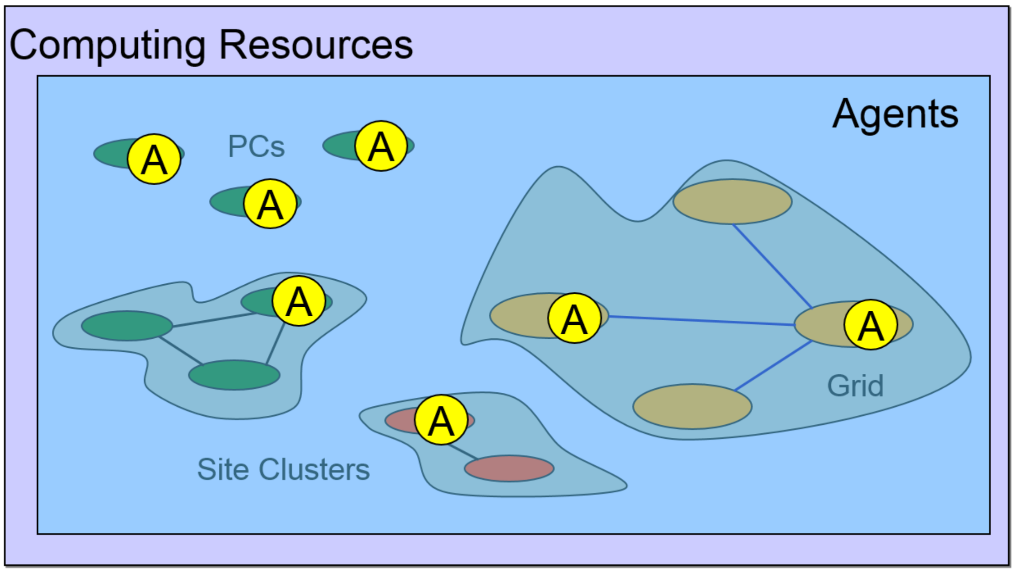

Figure 1. Agents were conceived as forming an overlay layer hiding the underlying diversity

Figure 1 is also taken from [6] and was putting in graphic the network paradigm that we just explained. We can imagine to substitute the word "agent" with the word "pilot". If this is the case, looking at this picture, it is already clear that pilots were conceived to be run on different types of computing resources: this concept is, as we shall see in next sections, one of the driving forces for which we developed pilots 2.0.

A DIRAC pilot has, at a minimum, to:

- install DIRAC
- configure DIRAC
- run an agent: the "JobAgent"

"JobAgent" is the DIRAC name for an agent that will try and fetch (match, in fact) a job (or, more than one job) from the central DIRAC jobs queue, which resides in the DIRAC WMS. We believe that the term "JobAgent" is common also for non-DIRAC implementations of pilot jobs.

In DIRAC, a pilot has to run on each and every computing resource type. We can then say that Pilots form an overlay layer hiding the underlying diversity. Or at least, they should. The first version of pilots was not doing this jobs well enough, for example the pilots monitoring was possible for only a limited amount of types of resources (i.e. WLCG types of computing resources). This is one of the reasons why Pilots 2.0 have been developed.

4. Grid computing in 2015

Within the last years, many LHC and non-LHC communities started adding different types of computing resources to their grid. For example, while for LHC VOs WLCG resources still provide the majority of available computing power, this is certainly not true for non-LHC communities. DIRAC is used by tens of VOs with rather different computing models and resources between one another. At the same time, most LHC communities now include resources like:

- Standard CEs, like *CREAM* CEs, with direct pilot jobs submission, but also other implementations of CEs, e.g. ARC (NorduGrid) CEs are a popular choice among sites. And also DIRAC types of CEs.
- WNs that are instantiated Virtual Machines, that the experiment provides for example using the "Vacuum model" that VAC [7] 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, althought many of them are using HPC sites as standard grid farms.
  - Volunteer computing, usually achieved via BOINC [8] [9].

In other terms, the Grid "push" model that experiments have used, and still use, is complemented by using IAAS (Infrastructure as a Service), or 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. We can say that the grid is not anymore "The Grid". Today's experiments' grids are made of heterogeneous resources.

Heterogeneity complicates things. Just for the sake of showing this concept, we can try the following exercise: monitor pilots in today's grids. For example, getting the *Logging Info* of a pilot. Depending from where this pilot has been executed, we have different libraries and commands to use:

- If the pilot ran on a *LCG CE*, the command to use for retrieving its logging output would be: `edg-job-logging-info v 2 --noint <ref>`.
- On a *gLite CE*: `glite-wms-job-logging-info v 3 --noint <ref>`.
- *CREAM*: `glite-ce-job-status L 2 <ref>`.
- *ARC*: this functionality is for the moment a feature request.
- *DIRAC CEs*: not available.

- *CLOUDs*: it depends from cloud to cloud and from what you expect.
- *BOINC*: not available.
- *HLT*: ...not?
- *Opportunistic* : ...
- ... *resources of tomorrow*: ??

If instead of the logging info we request for the full log output file, we would probably have a very similar story as the one just exemplified. So, how to work in such a heterogeneous environment? There are basically two possibilities: the first means keeping adding support for each and every type of resources. The second is a better option: we can just embed this functionality in the pilot.

5. Pilots 2.0 as overlay layer

While re-engineering DIRAC pilots, we had a few requirements to satisfy:

- 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 every computing resource, provided that:
  - Python 2.6+ is installed on the WN
  - It is an OS onto which we can install DIRAC

The same pilot, which we repeat, is a standalone script with minimal dependencies, can be used everywhere. Also on tomorrow’s resources. Figure 2 is similar to figure 1, just projected in 2015.

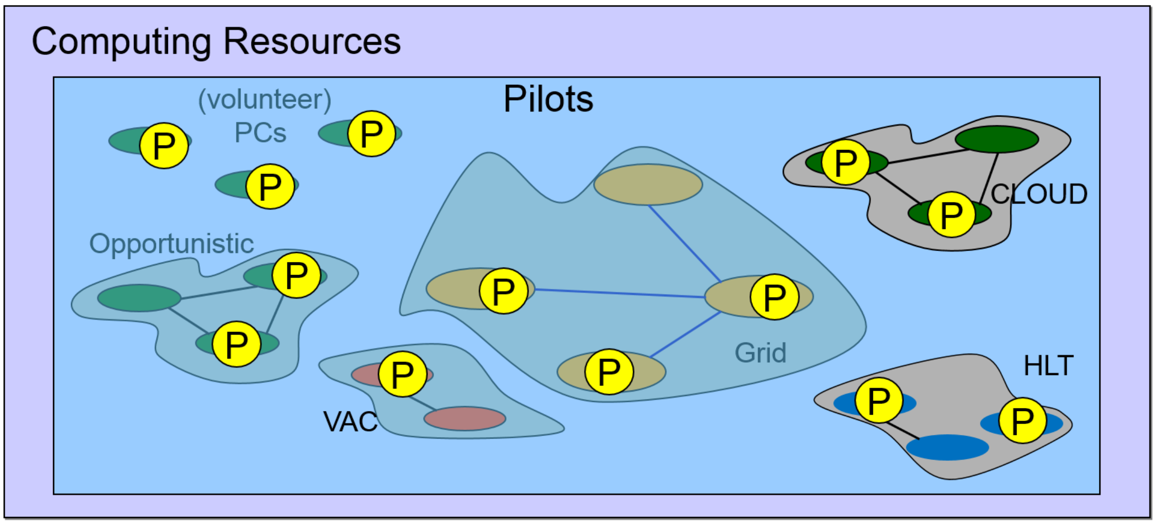

Figure 2. Pilots 2.0 form an overlay layer hiding the underlying diversity

5.1. Coding rules for Pilots 2.0
The coding rules for Pilots 2.0 have been defined collectively in a DIRAC RFC (`https://github.com/DIRACGrid/DIRAC/wiki/Pilots-2.0:-generic,-configurable-pilots`). A toolbox of pilots capabilities (that we will call "commands") is available to the pilot script. Each command implements a single, atomic, function, e.g.:

- Run an environment test
- Install DIRAC (or its extension)
- Configure DIRAC
- Run the JobAgent
- Run monitoring thread
- Report usage
- ... and whatever is needed

Communities can easily extend the content of the toolbox, adding more commands. If necessary, different computing resource types can run different commands, since everything is configurable.

In order to start a pilot, 4 python scripts are needed:

- `dirac-install.py`
- `dirac-pilot.py`
- `pilotTools.py`
- `pilotCommands.py`

If you manage to get these files on the Worker Node, then running the pilot can be as simple as:

```
python dirac-pilot.py \
 --debug \
 --project LHCb \
 -o '/LocalSite/SubmitPool=Test' \
 --configurationServer dips://lhcb-conf-dirac.cern.ch:9135/Configuration/Server \
 --Name "$CE_NAME" \
 --name "$DIRAC_SITE" \
```

While several switches are available for possible customizations.

6. LHCb pilots 2.0
Pilots 2.0 can be easily extended. An example of such extension is represented by the LHCb pilots 2.0.

CVMFS[10] is a strong requirement for LHCb: CVMFS should be present on each and every WN LHCb wants to run jobs on. CVMFS is used for the distribution of all LHCb software, including LHCbDIRAC. LHCb Pilots will try to "install" (set up) LHCbDIRAC from CVMFS, if it fails (e.g., its a just deployed version, and the cache is cold or a test version), will fall back installing the old way, which means downloading a tarball with pre-compiled binaries to the worker node and then launching a dirac script for installing DIRAC. Another difference is that LHCb pilots wont download CAs (Certification Authorities files), using instead what can be found on CVMFS.

7. Summary and prospects

As exemplified in figure 3, pilots 2.0 are the pilots to fly in all the skies, since they can hide the diversities between all the distributed computing resources. Pilots 2.0 are the real federators, and WNs will look the same everywhere. Make a pilot to run, and you will monitor it the same way as all the others.

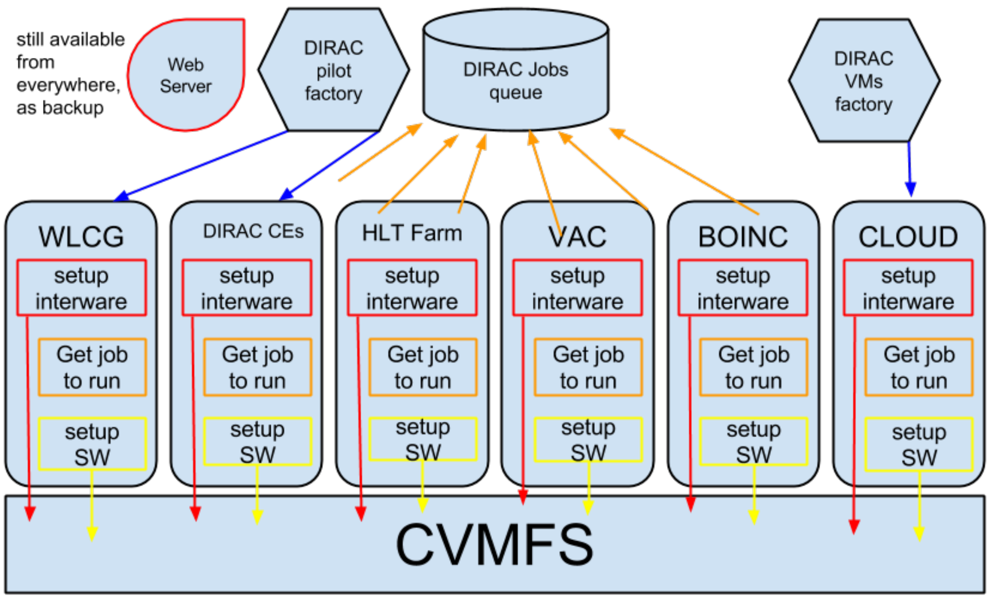

Figure 3. Pilots 2.0 run on each and every computing resource of LHCb

Pilots 2.0 are available to all DIRAC communities as of DIRAC v6r12, and are compatible with the previous generation of pilots, so it is possible that some communities may have not noticed the change. Pilots 2.0 have been implemented using vastly the command patten. Each command is realizing an atomic function, and can be easily activated and de-activated based on the WN type. Pilots 2.0 are easy to extend, as demonstrated within LHCb, and are actively developed also today. Prospects include adding more generic commands, one of these will be for running SAM jobs.

References

[1] Tsaregorodtsev A and the Dirac Project 2014 *Journal of Physics: Conference Series* **513** 032096 URL `http://stacks.iop.org/1742-6596/513/i=3/a=032096`

[2] Casajus A, Ciba K, Fernandez V, Graciani R, Hamar V, Mendez V, Poss S, Sapunov M, Stagni F, Tsaregorodtsev A and Ubeda M 2012 *Journal of Physics: Conference Series* **396** 032107 URL `http://stacks.iop.org/1742-6596/396/i=3/a=032107`

[3] Stagni F, Charpentier P, Graciani R, Tsaregorodtsev A, Closier J, Mathe Z, Ubeda M, Zhelezov A, Lanciotti E, Romanovskiy V, Ciba K D, Casajus A, Roiser S, Sapunov M, Remenska D, Bernardoff V, Santana R and Nandakumar R 2012 *Journal of Physics: Conference Series* **396** 032104 URL `http://stacks.iop.org/1742-6596/396/i=3/a=032104`

[4] Kuhr T and the Belle Ii Distributed Computing Group 2014 *Journal of Physics: Conference Series* **513** 032050 URL `http://stacks.iop.org/1742-6596/513/i=3/a=032050`

[5] Grefe C, Poss S, Sailer A, Tsaregorodtsev A, the Clic detector and physics study 2014 *Journal of Physics: Conference Series* **513** 032077 URL `http://stacks.iop.org/1742-6596/513/i=3/a=032077`

[6] Tsaregorodtsev A, Brook N, Sanchez M, Smith A C, Stokes-Rees I, Casajus A, Charpentier P, Closier J, Garonne V, Graciani R, Kuznetsov G, Paterson S and Saborido Silva J 2006 URL `https://cds.cern.ch/record/1397928`

[7] McNab A, Stagni F and Garcia M U 2014 *Journal of Physics: Conference Series* **513** 032065 URL `http://stacks.iop.org/1742-6596/513/i=3/a=032065`

[8] Anderson D P 2003 *Conference on Shared Knowledge and the Web, Residencia de Estudiantes, Madrid, Spain* URL `http://boinc.berkeley.edu/madrid.html`

[9] Himyr N, Blomer J, Buncic P, Giovannozzi M, Gonzalez A, Harutyunyan A, Jones P L, Karneyeu A, Marquina M A, Mcintosh E, Segal B, Skands P, Grey F, Gonzlez D L and Zacharov I 2012 *Journal of Physics: Conference Series* **396** 032057 URL `http://stacks.iop.org/1742-6596/396/i=3/a=032057`

[10] Blomer J, Buncic P, Charalampidis I, Harutyunyan A, Larsen D, and Meusel R 2012 *Journal of Physics: Conference Series* **396** 052013 URL `http://stacks.iop.org/1742-6596/396/i=5/a=052013`