Managing User Communities
Workflows With DIRAC

SW and Computing round table, JLab, May 3rd 2022

Federico Stagni
DIRAC technical coordinator
federico.stagni@cern.ch
● What’s DIRAC
● What are the features and attributes that make DIRAC popular
● Why it is not *more* popular
● Current, upcoming and future developments
What’s DIRAC?

- A software framework for distributed computing
- A **complete** solution to one (or more) user community
- Builds a layer between users and resources

- Started as an LHCb project, experiment-agnostic in 2009
- Developed by communities, for communities
  - Open source (GPL3+), [GitHub](https://github.com) hosted
  - Python 3 (python 2.7 kept for current production release)
  - Publicly documented, active assistance forum, yearly users workshops, open developers meetings and hackathons
- The DIRAC **consortium** as representing body
… a few examples of what DIRAC can be used for

- sending jobs to “the Grid”
  - the obvious one...
- interfacing with different sites
  - with different computing elements
    - and batch systems
  - with different storage elements
- interfacing with different information systems
- interfacing with different catalogs
- interfacing with different MQs, DBs
- authenticate through different identity providers (in preparation)

- managing “productions” (e.g. reconstruction, simulation…)
- managing dataset transfers and removals
  - or delegate the task
- interacting with FTS
- providing a failover system
  - your jobs won’t fail because a certain SE is down, nor because of central service are down
- transfer data from the experiment (the “online”) to a Grid SE (the “offline”)
- monitor your resources with a policy-based system
- … and more
Installations and communities (that I know)

A framework shared by multiple experiments/projects, both inside HEP, astronomy, and life science

Experiment agnostic
Extensible
Flexible
Why is DIRAC popular?

- **DIRAC as-a-service** (1 installation, several VOs) available since long time
  - good for medium-small communities
  - national initiatives to provide DIRAC as-a-service (e.g. GridPP, DutchGrid) – very few DIRAC administrators for possibly dozens of communities

- Feature-rich, all-in-one (WMS, DMS, but also Productions and Dataset management, and monitoring)
  - again, good for limited-manpower communities

- Tightly-integrated DIRAC WebApp

- Actively developed and maintained

"Stable"

"Communication is the killer feature of DIRAC"
- **Users** define and submit **jobs**. Jobs have **requirements**. Job descriptions are stored in DIRAC’s Job DB.
- Independently, **Pilots** are started (1) on the sites’ worker nodes (WN).
- Pilots will try to **match** (2) the worker nodes’ capabilities to the jobs requirements.
- Jobs are started on WNs. DIRAC monitors their progress.
**Grids** (nowadays: HTCondor, ARC)

**Clusters** behind a BS
- access through SSH/GSSH tunnel
  - a really thin layer that we call “SSH CE”
- and then talks with batch system

**VMs scheduler**:
- Based on apache libcloud
- Contextualization from standard images
  - with, at least, the DIRAC pilot

**Vacuum**:
- VAC/vcycle resources
- BOINC Volunteer resources
- HLT farm (LHCb)

**HPC sites**
- it often means at least SSH+Slurm
  - more later on
Basics of DIRAC DMS:

- **LFNs**: unique identifier within DIRAC of a file
  
  Logical File Name (described as paths)

- **LFNs** are registered in **catalog(s)**.
  
  and there are implementations like the DFC

- **LFNs** may have **PFNs**, stored in **SEs**.
  
  Physical File Name on Storage Elements
  (and SEs are monitored, within the DIRAC Resource Status System)

  You can access those PFNs with several protocols.

- **DIRAC catalogs** for implementing namespace functionalities
  
  ○ Several **catalogs** can live in parallel
    
    → **DFC**: full replica and metadata catalog – used by all DIRAC installation but Belle2
    
    → **plugins** for LFC, Rucio, DIRAC TS, LHCb Bookkeeping, Belle2 AMGA (?)

(almost ready) ES/kibana/grafana!
● Some VOs using DIRAC would like to use Rucio as DMS
  ○ and maybe some VOs using Rucio would like to use DIRAC
● Discussions started at the 8th DIRAC workshop (May 2018)
● Since January 2021 Belle2 uses DIRAC and Rucio
  ○ from LCG file catalog to Rucio FC
● Developments were (and are being) done on both sides
  ○ not many lines of code
  ○ available: integration of (multi-VO) DIRAC with (multi-VO) Rucio, working without a rucio.cfg
The Transformation System (TS) is used to automate common tasks related to production activities

- A “production” is a transformation managed by the TS that is a “Data Processing” transformation (e.g. Simulation, Merge, DataReconstruction…). A Production ends up creating jobs in the WMS.
- A “Data Manipulation” transformation replicates, or remove, data from storage elements. A “Data Manipulation” transformation ends up creating requests in the RMS (Request Management System), which feeds the DMS.

The Transformation System is finely tuned and can manage millions of jobs and files daily.
• Web apps available for monitoring jobs, files, productions, etc.
• Configurable
• Extendable
"Horizontal" extensibility
- For specific requirements

"Vertical" extensibility
- Community driven
Why not more popular?
(a bit more than a guesswork)

- complex, with high entrance bar
- somewhat cumbersome deployment
- known for (only) its WMS functionalities
- often a bit late on “standards”
  - http services
  - tokens
  - monitoring
- still considered “an LHCb thing”
- “old”-ish design
Some DIRAC developments

- **always valid**: Integrating DIRAC workflows in HPCs
- **always valid**: DMS advancements
- **Done**: Python 3
  - py3 clients supported since version 7.2 (pip installable)
  - py3 server supported since version 7.3 (production)
  - py2 support ends with 8.0 (release is few weeks away)
    - with some obvious exceptions of part of pilots code
- **Ongoing**: dips:// → https://
  - dips: DIRAC proprietary protocol for RPC calls
  - http: based on tornado
  - several DIRAC services already available using HTTP, and adding more
    - http will be the default for all the DIRAC services from version 8.1
- **Ongoing**: token support, and IdP (IaM, Check-in)
- **Ongoing**: ES/kibana/grafana dashboards
- **Started**: running on kubernetes (goal: define a helm chart)
- **Started**: using celery and RabbitMQ (retiring part of DIRAC framework)
Running on HPCs

**Different solutions** must be adopted for different HPCs. DIRAC can only take care of the distributed computing issues, such as:

- For MP WNs: logical partitioning using DIRAC “inner” PoolCE
- Interfacing with the batch system (mostly slurm)
- Interfacing with WNs with limited network connectivity using the *PushJobAgent*

VO actions often needed.

On software challenges: the good news is that py3 versions of DIRAC clients (so, DIRAC pilots) can be installed on *ppc64le* and *aarch64* → but, you need to be able to start the pilots
Tokens and OAuth2

- Current production release depends on VOMS
- DIRAC v8 (due in few weeks) rationalizes many aspects related to AuthN, AuthZ, Tokens and OAuth2 support, and will add experimental support to new Identity Providers (IaM and CheckIn)
- Longer term goals include:
  - externalize (to IdPs) users’ management
  - use tokens (and/or proxies) for interfacing with computing and storage resources (v8.1)
Accounting and Monitoring

Accounting:
- For historic data
  - Jobs
  - Pilots
  - Data Operations
  - Storage
- MySQL backend
- DIRAC Web App dashboard

Monitoring:
- Real Time monitoring and not only
- ElasticSearch (OpenSearch) backend
- Visualize in kibana, grafana, and (partially) DIRAC WebApp
- largely improved within DIRAC v8
Part of the communities

- Participation in WLCG activities (mostly DOMA)
- EGI
  - Check-In
  - Jupyter notebooks
~6 FTE as core developers, a dozen contributing developers

Tests, certification, integration process is a daily work.

- We use (lots of) GitHub Actions, and Jenkins for some bits
- We run certification hackathons every 2nd week
The 2022 DUW is next week!

https://indico.cern.ch/e/DUW11

(on zoom, free, CEST mornings)

we’ll record the sessions
**Questions/comments?**

- [dirac.readthedocs.io](https://dirac.readthedocs.io)  
  - including [code documentation](https://dirac.readthedocs.io)
- Ops and general questions: Google [forum](https://github.com) – but we prefer [github discussions](https://github.com)
- Dev and DevOps issues: on [github](https://github.com)
- Bi-weekly developers meetings (and/or hackathons): [BILD](https://github.com)
**Storage Elements**: abstraction of the storage endpoints

- fully described in Configuration Service (CS)
- several abstractions of the same physical endpoint are possible
- Multi-protocol
  - DIP: DIRAC custom protocol
  - File: offers an abstraction of the local access as an SE.
  - RFIQ (deprecated): for the rfio protocol.
  - Proxy: to be used with the StorageElementProxy.
  - S3: for S3 (e.g. AWS, CEPH)
  - GFAL2_SRM2: for srm
  - GFAL2_XROOT: for xroot
  - GFAL2_HTTPS: for https
  - GFAL2_GSIFTP: for gsiftp
- SpaceOccupancy plugins:
  - BDIIOccupancy, WLCGAccountingJson, WLCGAccountingHTTPJson

**SEs definitions are sync-ed from DIRAC CS to Rucio RSE via a DIRAC agent**
[RSS] Resource Status System

- Stores info on the status of Resources (e.g. SEs)
- An autonomic computing tool evaluates a few policies to determine the status of the resources. E.g.:
  - space left < threshold $\rightarrow$ ban for writing
  - endpoint in downtime in GocDB $\rightarrow$ ban r/w
  - ...
- DIRAC SEs states are sync-ed from DIRAC RSS to Rucio via a DIRAC agent
A generic system, which can be used for queueing (also) DMS operations

files

/lhcb/user/f/fstagni/shoppingList.txt
/lhcb/user/f/fstagni/presents.docx

files

/lhcb/user/f/fstagni/beach.jpg
/lhcb/user/f/fstagni/sunset.jpg
/lhcb/user/f/fstagni/beers.jpg

operations

1. Register in catalogs

Request #12345

operations

1. COPY from A to B
2. REMOVE from A

Request #12346

Operation types:
- ReplicateAndRegister (e.g. using FTS)
- RemoveFile/RemoveReplica
- ...others (not useful for this pres)
- ...add your own (e.g. ReplicateUsingAnotherExternalSystem)

and the Request Executing Agent will... execute the requests.
A **generic** system for queueing similar *operation types* on certain *datasets* and forward them to the appropriate systems.

<table>
<thead>
<tr>
<th>An <em>operation type</em> can be, e.g.:</th>
<th>A <em>dataset</em> is split into groups, based on criteria defined by <em>plugins</em>, e.g.:</th>
<th>A <em>system</em> is either (today) the DIRAC WMS (for productions) or the DIRAC RMS (for dataset management operation types):</th>
</tr>
</thead>
<tbody>
<tr>
<td>● a simulation workflow</td>
<td>● split by size</td>
<td>• split by size</td>
</tr>
<tr>
<td>● a reconstruction workflow</td>
<td>● by destination</td>
<td>• by destination</td>
</tr>
<tr>
<td>● a replication</td>
<td>● by metadata</td>
<td>• by metadata</td>
</tr>
<tr>
<td>● a removal</td>
<td>● … [code it]</td>
<td>• … [code it]</td>
</tr>
<tr>
<td>● ...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**E.g.** Take all my holidays pictures from 2018 with tag='sunset', make sure that there is one copy on tape and one on disk, distributed on all the sites according to free space, and group the operations by group of at most 100 files.