DIRAC in Large Particle Physics Experiments
(and not only)

Federico Stagni
Andrei Tsaregorodtsev, André Sailer
Luisa Arrabito, Xiaomei Zhang, Takanori Hara
The DIRAC Interware

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

And many others
just some examples:

→ more details
GridPP Dirac
Accommodating requirements

Each VO may have different requirements, how to accommodate all of them?

- Open source (real!)
- Extensibility
- Flexibility
An open source project

● Developed by communities, for communities
  ○ Open source (GPL3), GitHub hosted, python 2.7
  ○ No dedicated funding for the development of the “Vanilla” project
  ○ Publicly documented, active assistance forum, yearly users workshops, open developers meetings
  ○ 5 FTE as core developers, a dozen contributing developers

● The DIRAC consortium as representing body
  ○ CERN, CNRS (Marseille, Montpellier), UB, KEK, IHEP as members
  ○ First users (after LHCb) in 2009
Accommodating different requirements:
Developed with extensibility in mind

“Horizontal” extensibility
- For specific requirements

For specific requirements

“Vertical” extensibility
- Community driven

A DIRAC release is composed by all the projects (strong dependency)
Each project is independently versioned

DIRAC

WebAppDIRAC

RESTDIRAC

VMDIRAC

COMDIRAC

externals

DIRAC

WebAppDIRAC

VMDIRAC

COMDIRAC

Pilot

| more details |
| ...next presentation |

Core project

VO

DIRAC

VO

WebAppDIRAC

WebAppDIRAC
Flexibility: just 1 example

- DIRAC does not impose you any computing model
- Tiers level may not mean anything to certain VOs
- The (fixed) computing model is dead

- Example: full mesh computing model
  - Every job can run everywhere
- A real world computing model is a full mesh with limits
- All configurable in few clicks
DIRAC first developers, main maintainers

Uses DIRAC for all distributed computing needs (almost)

Computing resources:
- Grid: CREAM, ARC, HTCondorCE
- Private Clusters: SSH/GSISSH tunnels
- Batch: LSF, SGE, Condor, Torque
- Clouds: LHCbDIRAC pilots spawned by vcycle
- Vac, opportunistic (BOINC, HLT, HPC) with LHCbDIRAC pilots
  - Req: CVMFS

Storage, DataManagement:
- SRM (gfal, gfal2), DIP, FTS3

Catalogs:
- DFC (DIRAC File Catalog)
- Bookkeeping (data provenance)

Extended:
- LHCbDIRAC - most of systems
  - Massive data production and manipulation with chained “transformations”
- LHCbPilot
- LHCbWebAppDIRAC

End Users interface with LHCbDIRAC client or with Ganga
- Computing resources:
  - Grid: CREAM, ARC, HTCondorCE
  - Private Clusters: SSH/GSI/SSH tunnels
  - Clouds: VMDIRAC, CloudScheduler, Dynamic Torque
  - Opportunistic (e.g. BOINC) with DIRAC pilots

- Storage, Data Management:
  - SRM (gfal, gfal2), XRootD, DIP, FTS3

- Catalogs:
  - LFC (Replica) + AMGA (Metadata)
  - DFC (in development instance)

- Extended:
  - Automated "Production system" based on Transformation System

[Image of Running jobs by Country and Transferred data by Destination graphs]

**Highlights of Belle2 Computing**
- Use cases similar to other HEP experiments
  - Massive data-production and processing
    - 360M HS06 (DB12) CPU hours
    - 2 PB produced (disk/tape)
  - Need to automatise complex workflows
- DIRAC evaluation started 5 years ago
  - DIRAC-based setup built from scratch
  - Use DFC as replica and meta-data catalog
  - Use Transformation System to manage large
    - ‘productions’
- Extended
  - Interfaces to easily configure CTA applications
- DIRAC contribution
  - Current development to achieve a fully ‘data-driven’
    - Transformation System
  - Aim to develop a ‘Production System’ general enough
    to be useful to many communities
Shared DIRAC instance for Linear Collider detector studies; CLICdp, ILD, SiD collaborations

- Computing Resources
  - OSG&WLCG
- DFC (metadata too)
- Storage: SRM, XRootD
- Last 12 months: 3k CPU years, 4M Jobs, 2PB

→ more details
Using OSG Computing Resources with (iLC)DIRAC

- Extended: Job Submission and Workflow Modules for Linear Collider Software
  - Easy chaining of applications
  - Simple job description via python

```python
from DIRAC.Core.Base import Script
Script.parseCommandLine()
import UserJob
import Marlin
import DiracILC

d = DiracILC()
j = UserJob()
j.setOutputSandbox("recoEvents.alcio")
m = Marlin()
m.setVersion("ILCSoft-01-17-09")
m.setSteeringFile("Steering.xml")
m.setInputFile("SimEvents.alcio")
j.append(m)
j.submit(d)
```

- Very happy users running lots of jobs
Distributed computing system needed for peek needs of BESIII
- Set-up should be easy and flexible enough with few manpower and experience
- Start since 2012 and put into production in 2014
  - Jobs reached 1.29M in 2015

Extended:
- Task submission and management system
- Massive Data transfer system
- Site Monitoring system based on DIRAC Resource Status System (RSS)

Computing resources:
- 10% Grid, 65% Batch, 25% Clouds
- Integrate Clouds with VMDIRAC

Catalogs:
- Use DFC for Replica Catalog, Metadata Catalog, Dataset Catalog

Shared DIRAC instance since 2015: other IHEP experiments, eg. JUNO, CEPC, LHAASO, etc

→ more details
A lightweight task submission and management infrastructure
Summary

- Actively used and developed
- Satisfies the requirements and use cases of several communities
  - Extensibility at several levels, plugging resources, flexibility
  - Not everything is singing & dancing, especially for multi-VO support
- Can be extended to accommodate experiment-specific use cases
  - Common use cases:
    - Interfacing to experiment software
    - Productions handling

Come on board!
Questions/comments