DIRAC update

J. Martyniak, S. Fayer & D. Bauer

Imperial College
London

Introduction

- What is DIRAC and how does it fit within the SwiftHEP remit ?
- SwiftHEP: Things that worked (eventually)
- SwiftHEP: And one thing that didn’t
- What we are going to do next

What is DIRAC ?

- DIRAC was originally developed by LHCb as a Monte Carlo production system.
- The DIRAC consortium was founded in 2014 to enable adoption by other communities. The UK joined the consortium in 2019.
- Open source: https://github.com/DIRACGrid/DIRAC
- Documentation: https://dirac.readthedocs.io/en/latest/
- DIRAC comprises of:
  - Workload Management System
  - File Catalog/Data Management System
  - Workflow Management System
- Provides a standardised user interface to multiple compute (grid & cloud) and storage resources.
  - Outside the SwiftHEP remit: UK contribution has focused on multi-VO and clouds
  - SwiftHEP: PilotLogging, Workflow, User Interface and Integration with WP5 (Analysis Systems)

Status: What worked…..

- Past work: User friendly commands
  - For VOs too small to write their own production frontend, but too big to make do with low level commands.
  - Work done in conjunction with EGI, who faced a similar issue
  - SwiftHEP 1.8: Merged February 2023
- Past work: Pilot logging (though that turned out a bit more complicated than anticipated - see next slide)
  - Pilot logs are crucial to debug site problems
  - DIRAC used a pull model:
    - Pilots logs often lost in crashes: Just when needed the most !
    - No cloud logging
  - SwiftHEP 1.5: Changed to a push model, much safer that way

Things we learned about pilot logging

- An existing, unfinished (but merged!) remote pilot logging system could not really be used as a base of a new development.
- We opted for a system with configurable, pluggable back-ends with a Web Server as an intermediate log consumer.
- This required work on both the pilot (Python 2&3) and the Dirac server.
- We changed the way pilots are configured - this is now done reading a JSON file. This works both for Dirac-submitted pilots and “vacuum” pilots submitted by a shell script (a requirement by LHCb).
- The system is truly multi VO. Logging can be independently configured on a VO-by-VO basis.
- Log file retrieval is configurable and transparent to a user (both cmd line and WebApp).

Status: What worked….

- Recent work: Workflow improvements
  - DIRAC has an inbuilt system to allow users to chain operations together:
    - Metadata driven
    - Decidedly not multi-VO compatible it turned out
    - All fixed now
  - SwiftHEP 1.6: Code merged August 2023
  - Deployed in production at GridPP DIRAC

Status: And what didn't …..

- Workload management system fairly basic:
  - Jobs bound to sites quite early in submission process.
  - Target site immutable after submission and binding.
- Not flexible enough for large infrastructures, e.g.:
  - Unexpected changes in target site capacities (both up and down).
  - Misunderstandings lead to users submitting large batches of jobs to unsuitable target.
- This is not just a UK issue: Belle II sees the same problem.
- As it turns out this is technically infeasible: Once the job description is stored in the database, there is no interface to reprocess or update it; a significant modification to the DIRAC workload manager would be required.
  - This complexity for a basic system is part of the legacy code in DIRAC…

DiracX (the neXt DIRAC incarnation)*

- Last conceptual overhaul in 2008: DIRAC becomes a “multi-VO community solution”

- Regular updates within this framework:
  - python 2 to python 3
  - dashboards: ES/Kibana/Grafana
  - ongoing: implementing token support, identity providers: IAM, Checkin
    - In the UK this will have to be covered under GridPP/Operations for the time being.
  - ongoing: replacing DISET (DIRAC’s very own proprietary protocol for RPC calls) to HTTPS

Imperial College London

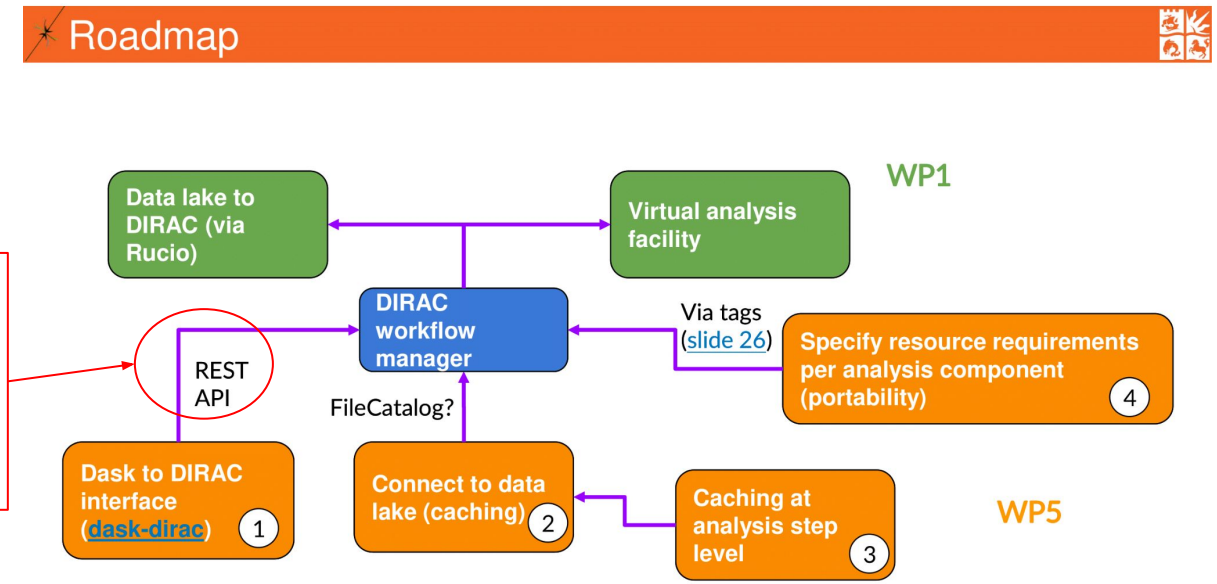
* For the record: We did not have any input on the naming scheme ;-)

DiracX

- Technological debt in DIRAC codebase threatened to overwhelm available maintenance effort.
- Aim to use current standards in place of old bespoke solutions:
  - REST/HTTPs interfaces for everything in place of DISET
  - SQAlchemy instead of custom written SQL statements
  - Designed with current requirements rather than grown organically
- https://github.com/DIRACGrid/diracx/discussions

How does DiracX fit in with WP5 ?

That bit needs some work.



Imperial College London

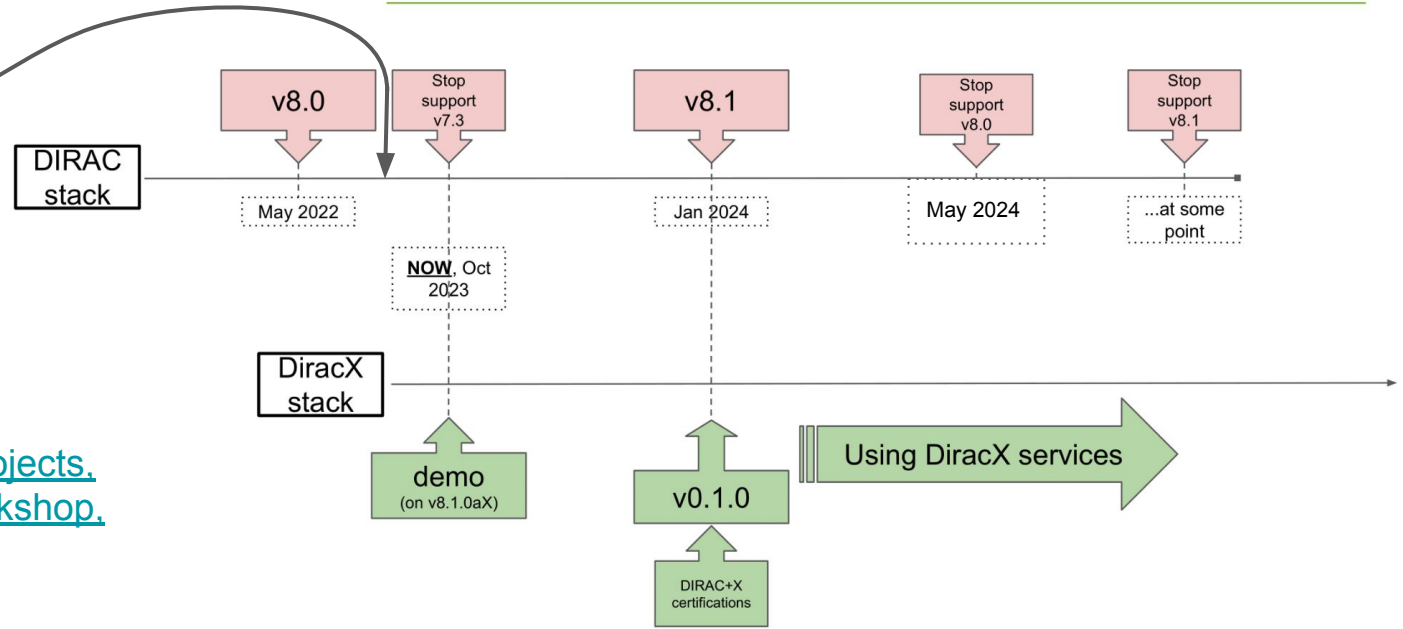
from Sam Eriksen

DiracX

- DiracX ties in with SwiftHEP work-packages:
  - Interoperability is key in tying everything together:
    - “SwiftHEP is all about integrating stuff”
  - Partial redesign allows us to include requirements and use-cases which didn’t easily fit into old code base.
- Example: WP5 Analysis
  - Currently uses prototype REST interface on DIRAC for workload management, basics are there but multiple difficult edge cases:
    - Proxy & sandbox management still require special consideration and support.
  - DiracX will have REST for all interfaces, no special cases.

DIRAC release timeline

July 2023:
GridPP DIRAC
Upgrade to v8.0



from:
F.Stagni,
Status of DIRAC projects,
DIRAC & Rucio Workshop,
Oct 2023

Imperial College London

Conclusions

- WP1.5, 1.6 & 1.8 Complete.
- Original WP1.7 plan not achievable.
- Replacement for WP1.7 proposed:
  - Use new DiracX services to provides REST interfaces for external tooling.
  - Links very well with WP5 requirements in both scope & timeframe.