DIRAC
Distributed Computing Resources

GSoC 2019 Project Presentation

Amarnath Karthi
Report Link

Mentors
Federico Stagni
Christophe Haen
Zoltan Mathe
What is DIRAC

**DIRAC** is a software framework for distributed computing.

- builds a layer between the users and the computing grids
- offers a common interface to a number of heterogeneous providers, integrating them in a seamless manner, providing interoperability
- **Ensures optimized, transparent and reliable usage of the resources.**
Problem Statement

**Resource Status System (RSS)** is responsible for maintaining the status of each computing resource (think storage, compute units, memory etc.).

The aim of my project as a whole was to revamp the RSS to be more robust by providing the following:

- A unified SQLAlchemy database backend
- Ensure that RSS is ready for Python 3
- Unit, Integration and Performance tests for RSS
Things delivered

- Base SQLAlchemy module for DB connections
- Migration of EmailAction and EmailAgent from SQLite to SQLAlchemy + MySQL
- Resource Status Notifications on Slack and Mattermost
- Removing SMS notification action.
- Integration test for RSS EmailAction and EmailAgent ready for Jenkins
- Python 3 compatible code ensured using pylint3k
- **Performance test for RSS in Locust and Taurus tool**
Base SQLAlchemy Module

2 connection modules based on SQLAlchemy:

1. ResourceStatusDB
2. ResourceManagementDB

A lot of code redundancy
Base SQLAlchemy Module

1. Add a base SQLAlchemy module to provide the basic DB functionality.
2. Both the connection modules extend base module

No more code redundancy
Notifications on Slack and Mattermost

- Slack and Mattermost are popular work chat applications. RSS now integrates with both to provide status change notifications.
- This is done using HTTP POST APIs and Webhook integrations provided by the Apps.
Performance Testing 101

- Helps in measuring and testing a service’s scalability.
- Helps study the impact of future changes, like adding a Cache.
- Required for DIRAC RSS, since it has several services.

Steps:

1. Add load to service by adding Virtual Users.
2. Each user repeatedly queries the service.
3. Response time is measured.
4. Use response time to plot Throughput, Latency etc.
Migrating to Locust

- Currently DIRAC uses MultiMechanize as its performance testing library.
- Although MultiMechanize is a really powerful tool, it is available only in Python 2.
- DIRAC will need to move to Python 3 by 2020.
- Support for MultiMechanize has ended already.

Locust is a good candidate for replacing MultiMechanize:

- Supports both Python 2 and 3
- Distributed & scalable: Use multiple machines for load generation
- Proven and battle tested in the industry: A huge community support
Taurus + Locust

Taurus tool is an Open Source test automation framework, providing simple YAML-based configuration. This will be used in the future to run and visualise performance tests on Jenkins.

Currently we use it to run Locust tests on CLI only.
Future Work

- Shift all Multi-Mechanize Performance tests to Locust and Taurus for Python 3 compatibility
- Complete Jenkins integration for Locust and Taurus tests with Metric plotting
- Explore Caching solutions for RSS