Bookkeeping DB integration in DIRAC

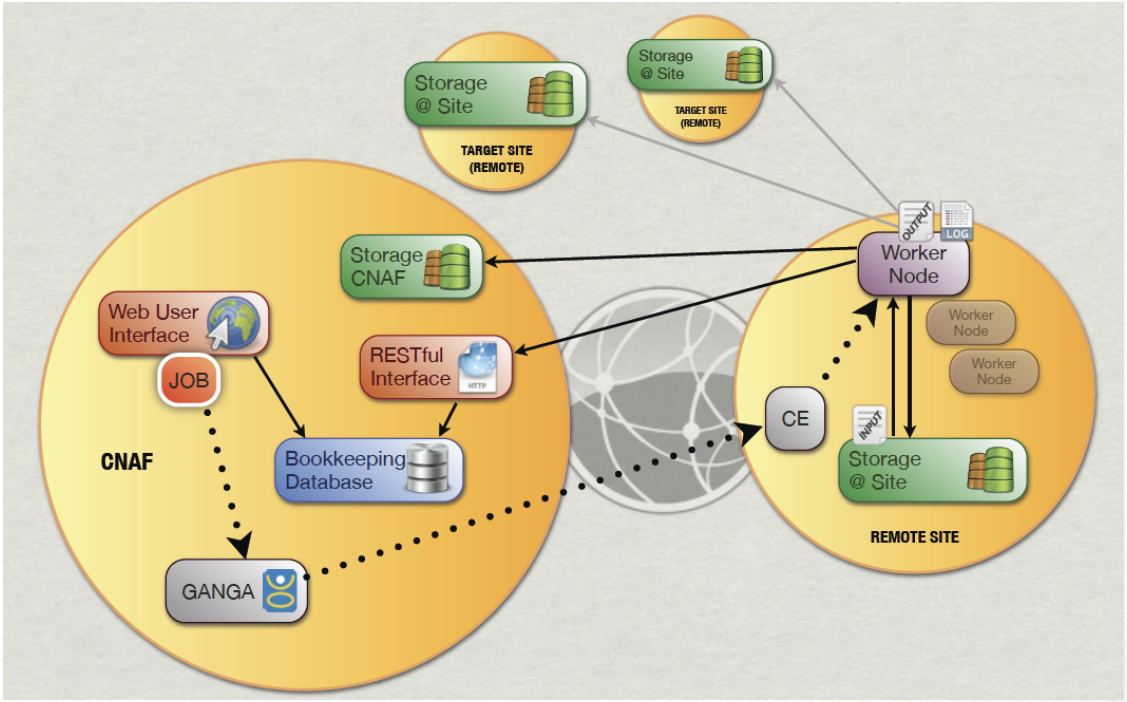
Miłosz Zdybał
Institute of Nuclear Physics PAN, Kraków, Poland

Overview

- Bookkeeping database – SBK
- Integration with DIRAC
  - SQLAlchemy
  - DIRAC Service
- Status update
- Discussion

Bookkeeping database – SBK5

- Central database for distributed computing model
- Collects all information about Fastsim and Fullsim jobs



DIRAC integration

- SuperB is moving to DIRAC, so SBK too
- Communication between other parts of distributed system and SBK is needed
  - Accessing PostgreSQL database
  - Exporting operations on database

SQLAlchemy

- Object-relational mapper
- Open source
- For Python
- Very powerful and easy to use
- Works with variety of SQL backends
- SQLAlchemy provides "a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language"

DIRAC Service

- How to put our code inside SuperBDIRAC?
  - DIRAC Service
- Why Service approach?
  - Most intuitive for me
  - LHCb handles bookkeeping this way
- How does it work?
  - Service is a class inheriting RequestHandler class
  - Functions with prefix *export_* are accessible as a service functions
  - Lists of argumets types for all service methods (type checking when calling methods in DIRAC?)

```
""" SBKService - testing service for SuperBDIRAC
"""

__RCSID__ = "$Id: $"

from types import *
from DIRAC.Core.DISET.RequestHandler import RequestHandler
from DIRAC import gLogger, S_OK, S_ERROR
from manager.site import site # my class, based on sqlalchemy mappers

from

def initializeSBKHandler( serviceInfo ):
    return S_OK()

class SBKHandler(RequestHandler):

    def initialize(self):
    """ Handler initialization
    """
    pass

    types_hello = []
    def export_hello(self):
    """ 'HELLo world' - the simpliest test is service working
    """
        return S_OK('HELLo World!')

    types_listSitesNames = []
    def export_listSitesNames():
        try:
            return S_OK(site.listNames())
        except:
            return S_ERROR('Error listing sites names')
```

DIRAC Service sample

Status update

- Environment for SQLAlchemy – done
- Mapping SBK in SQLAlchemy – mostly done
- Figuring out how to make DIRAC Service – mostly done
- Deployment of DIRAC Service – to be done

```
.
|-- database
|   |-- __init__.py
|   |-- entities
|   |   |-- __init__.py
|   |   |-- analysis_dataset.py
|   |   |-- analysis_dataset_site.py
|   |   |-- background_frame.py
|   |   |-- background_frame_site.py
|   |   |-- ce.py
|   |   |-- fastsim_dataset_site.py
|   |   |-- fastsim_env_var.py
|   |   |-- fastsim_job.py
|   |   |-- fastsim_prod_request.py
|   |   |-- fastsim_prod_site.py
|   |   |-- fastsim_production.py
|   |   |-- fastsim_request.py
|   |   |-- fastsim_soft_ref.py
|   |   |-- fastsim_stat.py
|   |   |-- fastsim_submission.py
|   |   |-- fullsim_dataset.py
|   |   |-- fullsim_dataset_site.py
|   |   |-- fullsim_job.py
|   |   |-- fullsim_log.py
|   |   |-- fullsim_output.py
|   |   |-- fullsim_output_type.py
|   |   |-- fullsim_parameter.py
|   |   |-- fullsim_parameter_values.py
|   |   |-- fullsim_prod_request.py
|   |   |-- fullsim_prod_site.py
|   |   |-- fullsim_production.py
|   |   |-- fullsim_request.py
|   |   |-- fullsim_soft_ref.py
|   |   |-- fullsim_stat.py
|   |   |-- fullsim_submission.py
|   |   |-- fullsim_subparameter.py
|   |   |-- se.py
|   |   |-- session.py
|   |   |-- session_site.py
|   |   |-- session_site_soft.py
|   |   `-- site.py
|   `-- mappers.py
`-- manager
    |-- __init__.py
    `-- site.py
```

SBK SQLAlchemy mapping structure

database – database related classes

entities – classes used for mapping of tables

mappers.py – definitions of mappers

manager – classes to be used by other modules

SQLAlchemy mappers

```
orm.mapper(fullsim_job, tables['sbk.fullsim_job'], properties = {
    'soft_release_obj': orm.relationship(fullsim_soft_ref, backref=orm.backref('fullsim_jobs')),
    'site_obj': orm.relationship(site, backref=orm.backref('fullsim_jobs')),
    'fullsim_submission_obj': orm.relationship(fullsim_submission, backref=orm.backref('fullsim_jobs')),
    'fullsim_prod_request_obj': orm.relationship(fullsim_prod_request, backref=orm.backref('fullsim_jobs'))
})

orm.mapper(fullsim_soft_ref, tables['sbk.fullsim_soft_ref'])

orm.mapper(site, tables['sbk.site'])
```

Manager class

```
from database import Session
from database.entities.site import site

def listNames():
    names = []
    for s in Session.query(site).all():
        names.append(s.site)
    return names
```

Discussion

- Deployment of Services under development
  - Procedures
    - Deploying/installing Services and restarting DIRAC (?) when changes commited to repository
  - Environment
    - Repository for the code
    - DIRAC instance for development
- ?