Dirac integration with a general purpose bookkeeping DB: a complete general suite for distributed resources exploitation

M Chrzaszcz, C De Santis[1], G Donvito[2], A Fella[3], R Grzymkowski, B Santeramo[2,4], L Tomassetti[5,6], M Zdybal

[1] INFN - Sezione di Roma Tor Vergata, Roma, Italy
[2] INFN - Sezione di Bari, Bari, Italy
[3] NFN - Sezione di Pisa, Pisa, Italy
[4] Dipartimento di Fisica dellUniversitá e del Politecnico di Bari, Bari, Italy
[5] Dipartimento di Matematica e Informatica, Universitá di Ferrara, Ferrara, Italy
[6] INFN - Sezione di Ferrara, Ferrara, Italy

E-mail: `bruno.santeramo@ba.infn.it`

Abstract. In the context of High Energy Physics computing field the R&D studies aimed to the definition of the data and workload models have been carried on and completed by the Super$B$ community beyond the experiment life itself. The work resulted of great interest for a generic mid- and small size VO to fulfill Grid exploiting requirements involving CPU-intensive tasks.

We present the R&D line achievements in the design, developments and test of a distributed resource exploitation suite based on DIRAC. The main components of such a suite are the information system, the job wrapper and the new generation DIRAC framework. The DB schema and the SQL logic have been designed to be able to be adaptive with respect to the VO requirements in terms of physics application, job environment and bookkeeping parameters. A deep and flexible integration with DIRAC features has been obtained using SQLAlchemy technology allowing mapping and interaction with the information system. A new DIRAC extension has been developed to include this functionality along with a new set of DIRAC portal interfaces aimed to the job, distributed resources, and metadata management. The results of the first functionality and efficiency tests will be reported.

1. Introduction

2. Description of suite design

philosophy: simple, standard and long term solution
bird's eye view all over the project

3. The Dirac extension

- extension structure description
- Dirac general purpose project short description + Dirac configuration
- service and systems description in detail

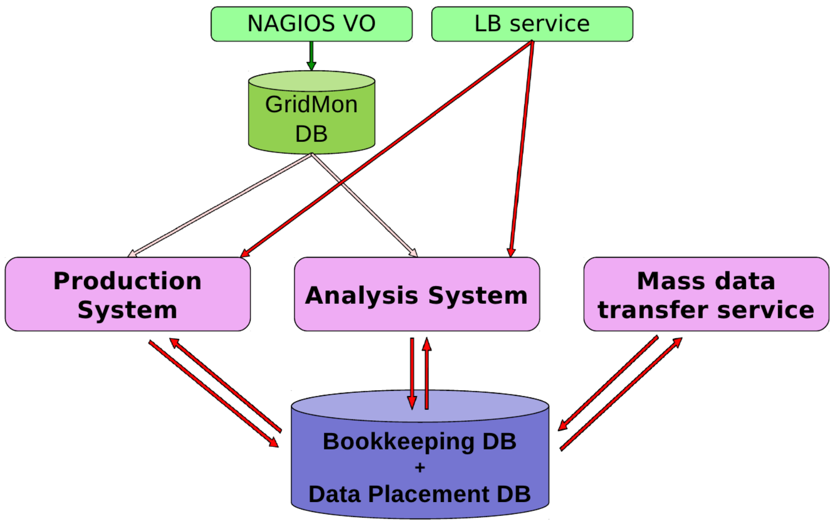


Figure 1. Distributed System bird's eye view.

- web interface components description

4. Bookkeeping DB integration

- Software layer based on SQLAlchemy
– advantages using SQLAlchemy: Object relational Mapping, clean code, fast
– development, change of DB backend
- BK description, highlighting the general purpose characteristics
– session, request, dataset concept

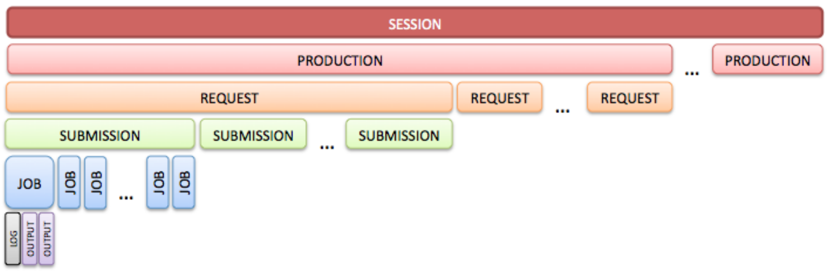


Figure 2. Entities in BookKeeping database.

5. Job wrapper component

- general workflow
- data management policy: stage-in and stage out strategies

6. Simulation production use case: the SuperB experience

Simulation Production is designed to manage huge MonteCarlo productions. A webportal, named WebUI[4], is available to manage the entire stack of operations related to this use case:

user management, definition of new "Sessions", "Productions" and "Request", job submission and monitoring, sites management for productions. Production manager users can add and delete sites, add and delete CEs and SEs, set a site a site as enabled/disabled and supported/unsopported for a given session, manage "Sessions", "Productions" and "Request". These actions involve only BookKeeping database, building up automatically required tables. Shifter users can submit and monitor jobs for a given Request. Job submission in WebUI is performed via Ganga[1], while configurations files for submission are generated by WebUI itself taking data from SBK. Ganga submit jobs to grid via WMS using standard gLite commands. Job execution on WNs is driven by Severus (see section 5), while job status updates in SBK is performed via REST interface. Stagein and Stageout as well as LFC registration of output files is performed even by Severus. See figure 3 for Simulation Production workflow.

SuperBDIRAC goal is porting WebUI functionalities in DIRAC to manage jobs, distributed resources and metadata related to a MonteCarlo production using a BookKeeping database. Job submission is directly managed by DIRAC. Grid computing resources can be used via WMS or direct submission to CREAM CEs, computer clusters can be accessed via ssh connections, Cloud resources are available via VMDIRAC module, even desktop computers can be used via Boinc. User authentication via X509 certificates and authorization VOMS-role based are builtin in DIRAC. SuperBDIRAC enhance DIRAC integrating the generic bookkeeping database SBK via SQLAlchemy. DIRAC webportal in order to provide an interface for Production manager actions as well as shifter tasks. At present time not all WebUI functionalities are yet ported in SuperBDIRAC, but site and job monitoring from SBK are available. In section 7 are reported results of first SuperBDIRAC functionality test.

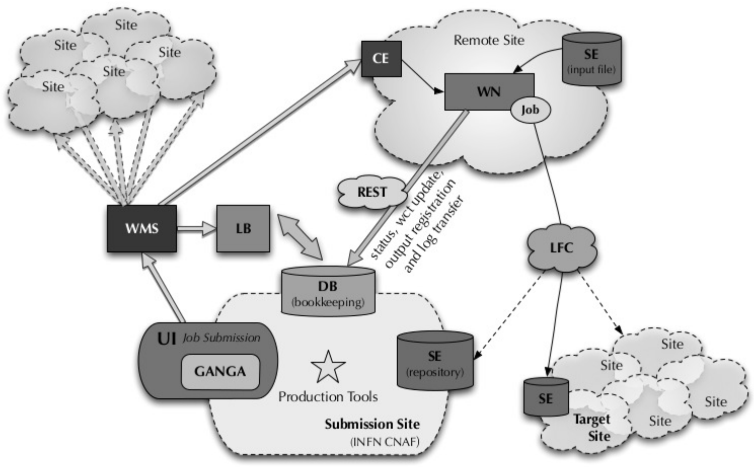
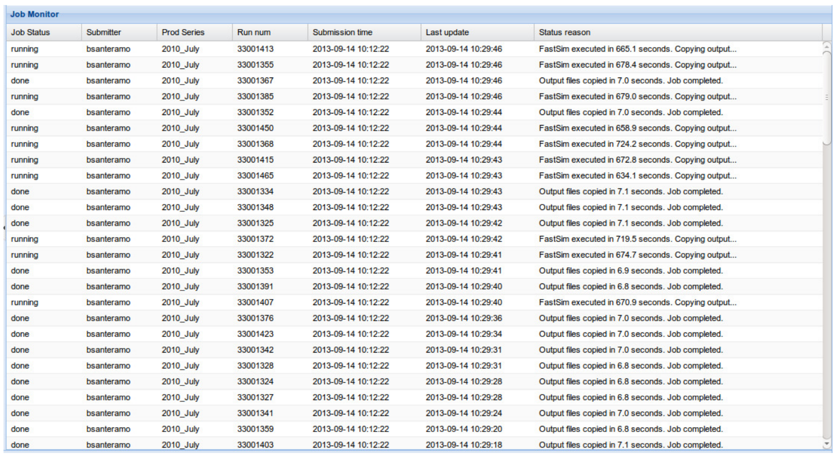


Figure 3. Simulation Production workflow.

7. Functionality Test

A functionality test was performed to demonstrate the capability of SuperBDIRAC to integrate the monitoring of a bookkeeping database in DIRAC webportal.



Job Monitor

Job Status	Submitter	Prod Series	Run num	Submission time	Last update	Status reason
running	bsanteramo	2010_July	33001413	2013-09-14 10:12:22	2013-09-14 10:29:46	FastSim executed in 665.1 seconds. Copying output...
running	bsanteramo	2010_July	33001355	2013-09-14 10:12:22	2013-09-14 10:29:46	FastSim executed in 678.4 seconds. Copying output...
done	bsanteramo	2010_July	33001367	2013-09-14 10:12:22	2013-09-14 10:29:46	Output files copied in 7.0 seconds. Job completed.
running	bsanteramo	2010_July	33001385	2013-09-14 10:12:22	2013-09-14 10:29:46	FastSim executed in 679.0 seconds. Copying output...
done	bsanteramo	2010_July	33001352	2013-09-14 10:12:22	2013-09-14 10:29:44	Output files copied in 7.0 seconds. Job completed.
running	bsanteramo	2010_July	33001450	2013-09-14 10:12:22	2013-09-14 10:29:44	FastSim executed in 658.9 seconds. Copying output...
running	bsanteramo	2010_July	33001368	2013-09-14 10:12:22	2013-09-14 10:29:44	FastSim executed in 724.2 seconds. Copying output...
running	bsanteramo	2010_July	33001415	2013-09-14 10:12:22	2013-09-14 10:29:43	FastSim executed in 672.8 seconds. Copying output...
running	bsanteramo	2010_July	33001465	2013-09-14 10:12:22	2013-09-14 10:29:43	FastSim executed in 634.1 seconds. Copying output...
done	bsanteramo	2010_July	33001334	2013-09-14 10:12:22	2013-09-14 10:29:43	Output files copied in 7.1 seconds. Job completed.
done	bsanteramo	2010_July	33001348	2013-09-14 10:12:22	2013-09-14 10:29:43	Output files copied in 7.1 seconds. Job completed.
done	bsanteramo	2010_July	33001325	2013-09-14 10:12:22	2013-09-14 10:29:42	Output files copied in 7.1 seconds. Job completed.
running	bsanteramo	2010_July	33001372	2013-09-14 10:12:22	2013-09-14 10:29:42	FastSim executed in 719.5 seconds. Copying output...
running	bsanteramo	2010_July	33001322	2013-09-14 10:12:22	2013-09-14 10:29:41	FastSim executed in 674.7 seconds. Copying output...
done	bsanteramo	2010_July	33001353	2013-09-14 10:12:22	2013-09-14 10:29:41	Output files copied in 6.9 seconds. Job completed.
done	bsanteramo	2010_July	33001391	2013-09-14 10:12:22	2013-09-14 10:29:40	Output files copied in 6.8 seconds. Job completed.
running	bsanteramo	2010_July	33001407	2013-09-14 10:12:22	2013-09-14 10:29:40	FastSim executed in 670.9 seconds. Copying output...
done	bsanteramo	2010_July	33001376	2013-09-14 10:12:22	2013-09-14 10:29:36	Output files copied in 7.0 seconds. Job completed.
done	bsanteramo	2010_July	33001423	2013-09-14 10:12:22	2013-09-14 10:29:34	Output files copied in 7.0 seconds. Job completed.
done	bsanteramo	2010_July	33001342	2013-09-14 10:12:22	2013-09-14 10:29:31	Output files copied in 7.0 seconds. Job completed.
done	bsanteramo	2010_July	33001328	2013-09-14 10:12:22	2013-09-14 10:29:31	Output files copied in 6.8 seconds. Job completed.
done	bsanteramo	2010_July	33001324	2013-09-14 10:12:22	2013-09-14 10:29:28	Output files copied in 6.8 seconds. Job completed.
done	bsanteramo	2010_July	33001327	2013-09-14 10:12:22	2013-09-14 10:29:28	Output files copied in 6.8 seconds. Job completed.
done	bsanteramo	2010_July	33001341	2013-09-14 10:12:22	2013-09-14 10:29:24	Output files copied in 7.0 seconds. Job completed.
done	bsanteramo	2010_July	33001359	2013-09-14 10:12:22	2013-09-14 10:29:20	Output files copied in 6.8 seconds. Job completed.
done	bsanteramo	2010_July	33001403	2013-09-14 10:12:22	2013-09-14 10:29:18	Output files copied in 7.1 seconds. Job completed.

Figure 4. Monitoring job data from SBK in DIRAC.

7.1. Goal description

DIRAC capabilities and performance as job management tool are well documented (insert some reference). Super*B* distributed simulation stack (WebUI + SBK + Severus) was successfully used in 3 simulation campaigns (ask to Armando for confirmation and references).
Functionality Test goal is to demonstrate if SuperBDIRAC is capable of substitute WebUI as monitoring tool for job submission, showing data from a bookkeeping database. Test "Q-factor" is the exact correspondence of information as stored in SBK and displayed in SuperBDIRAC, like in WebUI monitoring page. Correct execution of simulation jobs is not important while the bookkeping information are properly managed.

7.2. Testbed description

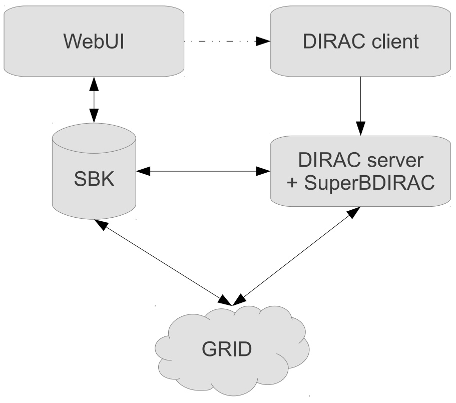


Figure 5. Testbed schema.

At present time, not all WebUI functionalities are implemented in SUperBDIRAC, in particular "Submission" job for a given "Request". The following procedure was followed to perform this test. FastSim job submission is created via WebUI interface. Once submission is created, a set of scripts and configuration files are created whose path is reported by WebUI interface: in particular the path of php submission script is taken to be be parsed. This php submission script, generated by WebUI, is made of an array with all relevant parameters and commands to UI in order to submit jobs in grid via standard glite commands. Since WebUI is still linked with SBK, this portal could be used as well as monitoring portal, useful for a check-cross between info displayed in SBK, WebUI and SuperBDIRAC.

The php submission script is parsed by a python script (mc_production.py), in particular the params array, in order to retrieve all needed paramenters to properly submit jobs: production series, session name, min and max runnumber, configuration files location, physical parameters, events to simulate. Once taken all these parameters, mc_production.py uses DIRAC API to prepare and submit jobs via DIRAC client.

DIRAC server receive jobs from DIRAC client, than starts the normal workflow for job management in DIRAC: scheduling, pilot submission, payload retrieval and execution, stageout. Since DIRAC server is equipped with SuperBDIRAC, even bookkeeping monitoring is performed by this component.

In SBK the job submission insertion is performed by WebUI, while later info are updated by Severus script executed with every job submitted. Severus interact with SBK via REST interface, while WebUI and SuperBDIRAC have direct access to SBK since they are in the same LAN area (ask to Armando for confirmation).

Jobs were submitted via grid by DIRAC server. Submission was performed using WMS instead of using direct submission to CREAM CE.

Submission site was INFN-T1, which ensured CPU time to execute latest simulations related to Super$B$ once the experiment closure.

7.3. Test description

Every job simulated 3000 events: this value is set to have an execution time quite longer than 10 minutes. Physical parameters are the same of other official productions. 3 main bunch submission of 400 jobs were performed at INFN-T1 to obtain a total of 1200 simulations. Status in SBK were "prepared", "running" and "done". In addition, 2 bunch submission of 10 jobs were performed, again at INFN-T1, to force some failure message in SBK and catch it even in SuperBDIRAC monitoring. First failure sample was obtained setting a not-existing site as destination for stageout: error was detected during preliminary check, so status in SBK were "prepared" and "failed". In second failure sample, jobs were submitted using a proxy without Role=ProductionManager, so error occurred in stageout phase: status in SBK were "prepared", "running" and "failed". In summary, 1220 jobs were submitted for test.

Since test goal, several screenshots of WebUI and SuperBDIRAC monitoring were saved, in addition to SBK data screenshots.

7.4. Results and conclusions

For each tests, all jobs were submitted at same time. Job execution depends on queue occupancy. Since we were interested in bookkeeping database status updates and not in measuring DIRAC performance in job execution, no dedicated CPU slots were asked for. While failing jobs were all executed simultaneously (10 dedicated CPU were always available for Super$B$ at INFN-T1), for 400 jobs tests simulation execution was spread on a greater time interval (see table 1).

Mean and max time could be interesting for performance measurements, but for this functionality test min time must be considered since it's related to execution time of job ran once submitted. First 3 good test have similar min time, compatible with estimated 13 minutes for simulation. For failure test, in first case the low min time is due to forced error in preliminary check phase (jobs fail without executing simulation code), in second case jobs execute simulation

Table 1. Execution time calculated as difference between Submission time and End Time in DIRAC scheduling system.

test	mean time (sec)	min time (sec)	max time (sec)
good-1	2203,66	859,00	5048,00
good-2	1102,92	832,00	2597,00
good-3	1188,62	849,00	4716,00
failure-1	165,10	161,00	172,00
failure-2	901,50	874,00	935,00

code but fail trying stageout (due to wrong Role-based permission in writing INFN-T1 storage area), so min time is longer as expected.

Table 2. Job status in SBK, WebUI and SuperBDIRAC.

test	jobs	site	SBK status	WebUI status	SuperBDIRAC status	success rate
good-1	400	INFN-T1	3	3	3	100%
good-2	400	INFN-T1	3	3	3	100%
good-3	400	INFN-T1	3	3	3	100%
failure-1	10	INFN-T1	2	2	2	100%
failure-2	10	INFN-T1	3	3	3	100%

BookKeeping database was properly updated for every job in every test. All status change were promptly displayed as well in SuperBDIRAC as in WebUI, without any appreciable delay between two portals. SQLAlchemy didn't introduced any appreciable delay or information loss, at least in this functionality test. Table 2 reports, for every test, how many status were saved in SBK and displayed in WebUI and SuperBDIRAC. Success rate was established as ratio between status saved in SBK and status displayed in SuperBDIRAC: its value was 100% in all submissions. SuperBDIRAC could be considered good enough to integrate in DIRAC the capability of monitoring jobs metadata from a bookkeeping database.

8. Conclusions

- We are offering a Dirac extended suite capable to satisfy the needs on small and mid size VOs in terms of distributed resource exploitation....

References

[1] `http://ganga.web.cern.ch/ganga`

[2] Super*B* Technical Design Report, `http://arxiv.org/abs/1306.5655`

[3] Fielding R T 2000 *Architectural Styles and The Design of Network-based Software Architectures* , PhD Thesis, University of California Irvine

[4] A.Fella, E.Luppi, L.Tomassetti *A General Purpose Suite for Job Management, Bookkeeping and Grid Submission.* International Journal of Grid Computing & Applications (IJGCA) Vol.2, No.2, June 2011. DOI: 10.5121/ijgca.2011.2202.