SuperB evaluation of DIRAC Distributed Infrastructure

A Fella\textsuperscript{1, 2, 3}, G Donvito\textsuperscript{4}, B Santeramo\textsuperscript{4}, A Gianoli\textsuperscript{5}, E Luppi\textsuperscript{3, 5}, M Manzali\textsuperscript{5}, L Tomassetti\textsuperscript{3, 5}, M Rama\textsuperscript{6}, G Russo\textsuperscript{7, 8}, S Pardi\textsuperscript{7, 8}, D Del Prete\textsuperscript{9}, R Stroili\textsuperscript{10, 11}, M Corvo\textsuperscript{1}, 10, S Longo\textsuperscript{10}, A Perez\textsuperscript{2}, A Di Simone\textsuperscript{8, 12}, F Bianchi\textsuperscript{13, 14}, S Luitz\textsuperscript{15}, F Giacomini\textsuperscript{16}, V Ciaschini\textsuperscript{16}

\textsuperscript{1} CNRS, Italy
\textsuperscript{2} INFN Pisa, Pisa, Italy
\textsuperscript{3} University of Ferrara, Ferrara, Italy
\textsuperscript{4} INFN Bari, Bari, Italy
\textsuperscript{5} INFN Ferrara, Ferrara, Italy
\textsuperscript{6} INFN LNF, Frascati, Italy
\textsuperscript{7} University of Napoli Federico II, Napoli, Italy
\textsuperscript{8} INFN, Italy
\textsuperscript{9} INFN Napoli, Napoli, Italy
\textsuperscript{10} INFN Padova, Padova, Italy
\textsuperscript{11} University of Padova, Padova, Italy
\textsuperscript{12} University of Roma Tor Vergata, Roma, Italy
\textsuperscript{13} University of Torino, Torino, Italy
\textsuperscript{14} INFN Torino, Torino, Italy
\textsuperscript{15} SLAC, CA, USA
\textsuperscript{16} INFN CNAF, Bologna, Italy

E-mail: armando.fella@pi.infn.it

Abstract. The SuperB asymmetric energy $e^+e^-$ collider and detector to be built at the newly founded Nicola Cabibbo Lab will provide a uniquely sensitive probe of New Physics in the flavour sector of the Standard Model. SuperB distributed computing group performed a detailed evaluation of DIRAC Distributed Infrastructure for use in the SuperB experiment based on the two use cases: End User Analysis and Monte Carlo Production. Test aims to evaluate DIRAC capabilities to manage both gLite and OSG sites, File Catalog management, job and data management features in SuperB realistic use cases.

1. Introduction
SuperB [1] is an international experiment aiming at the construction of a very high luminosity asymmetric $e^+e^-$ flavour factory. Together with the Large Hadron Collider (LHC) [2] and eventually the International Linear Collider (ILC) [3], SuperB will provide an important contribution in flavour physics and rare decays knowledge, improving the precision of several measurements. Electrons and positrons, circulating in a vacuum chamber 1.8 km in circumference, located underground, will be brought into collision using the innovative new crab waist design, recently tested at the INFN Frascati Laboratory. The design of SuperB, on the other hand, allows production of a hundred times the existing data sample with power...
consumption below that of the current generation of facilities. Moreover SuperB will be used as a very powerful synchrotron light source useful to visualize both biological or inorganic structures at a nano-meter resolution contributing to solid state physics, biology, nano technologies and biomedicine science fields.

2. Distributed Computing

The SuperB experiment needs large samples of Montecarlo simulated events in order to finalize the detector design, to estimate the analysis performances and to support the Technical Design Report studies. The total computing resources needed for one year of data taking at nominal luminosity are of the same order as the corresponding figures estimated, in the spring of 2010, by the ATLAS [4] and CMS [5] experiments for the 2011 running period. Therefore a distributed production model capable of exploiting the existing HEP worldwide distributed computing infrastructure is needed. So far, the main effort of the computing group has been devoted to the development and the support of the simulation software tools and the computing production infrastructure needed for carrying out the detector design and performance evaluation studies for the Detector TDR. The distributed computing infrastructure, as of January 2012, includes several sites in Europe and North America as reported in Fig 1. Each site implements a Grid flavour depending on its own affiliation and geographical position, EGI [6] and OSG [7].

The LHC Computing Grid (LCG) architecture [8] was adopted to provide the minimum set of services and applications upon which the SuperB distributed model could be built. To give a shape of the computing efforts SuperB experiment will involve, the estimation of network, CPU and disk space resources consumption have been reported:

- 5Gbyte/s of network bandwidth as experiment output
- 1000 PB amount of data collected after 5 year of experiment life
- about 50000 CPU Cores needed for all the activities for the first year of data taking

The three main computing intensive experiment tasks will be modelled to accomplish a fully distributed computational environment: Montecarlo (MC) simulation production, data reprocessing and chaotic final user data analysis. In parallel with the Detector TDR definition support, the Computing group works are focusing on a dense R&D program permitting to face technology and resources in 5 years future scenario. It is of worth to cite the R&D plan involving three computing centres located in the South of Italy with the aim of accessing resources as located in a single, large, computing facility [9].
3. About DIRAC

DIRAC (Distributed Infrastructure with Remote Agent Control) [10] was initially developed as a MC production tool using pilot jobs for for LHCb experiment [11], afterwards also data management functionalities were added and it became the default Data and Workload Management System for LHCb. DIRAC introduced the now widely used concept of Pilot Agents. This allows efficient Workload Management Systems (WMS) to be built. Pilot Jobs are nothing more than empty resource reservation containers that are sent to the available computing resources with the aim of executing the most appropriate pending payload in the central WMS queue. There are several advantages in this approach with respect to a more classical payload pushing scheduling mechanisms:

- Pilots allow to perform basic sanity checks of the running environment prior to any binding with a given payload. In case problems are found they can be reported to the DIRAC WMS framework for further actions. The user payload has not been assigned yet, so there is no impact from these problems onto the payloads.
- Pilots allow to create an overlay network that masks the central DIRAC WMS components from the heterogeneity of the underlying resources. Once the DIRAC Pilots are deployed on the Worker Node, they all present the same interface to DIRAC.
- Pilots allow an effective implementation of the pull scheduling paradigm. Once they are safely running in the computing resource, they contact central WMS servers for a late binding of the resource to the payload. This overcomes many intrinsic problems of the push mechanism when running on a world-wide distributed environment with all the unforeseeable delays in the delivery of the pushed payload to the executing resource, as well as with those issues derived from an exact knowledge of the current and near future local situation at each and every target resource, an impossible pre-requisite for a proper functionality of a such type of scheduling when attempted in this kind of environments. When presenting their grabbed resources Pilots produce the most accurate picture of those currently available.
- Pilot jobs can works in filling mode: in a single CPU time slot, a pilot can run several jobs.

Every DIRAC Pilot Job performs an in situ DIRAC installation including a full download of the most current version of the configuration. In this fresh installation and after checking the working conditions (i.e.: exact location where execution is taking place, available disk space, memory, cpu, available Grid environment, running platform) a DIRAC Job Agent is executed. This Job Agent is responsible for placing the payload request to the central DIRAC WMS server and for the later execution of the received payload. To provide a common execution environment for all payloads, the Job Agent instantiates a Job Wrapper object, last responsible of the payload execution. At the same time it also instantiates a Watchdog to monitor the proper behaviour of the Wrapper. The watchdog checks periodically the situation of the wrapper, takes actions in case the disk or available cpu is about to be exhausted or the payload stalls, and reports to the central WMS. It can also execute management commands received from the central WMS, like killing the payload. The wrapper retrieves the input sandbox, checks availability of required input data and software, executes the payload, reports success or failure of the execution, and finally uploads output sandbox and output data if required. The DIRAC wrapper takes care in an uniform way of all these details providing the user with a well defined and common execution environment across different computing resources. In order to efficiently determine the most appropriate payload to be handled for execution to a Pilot Job at any given time, pending tasks are organized into TaskQueues. The workload of the community is optimized in the central TaskQueue. The WMS is carefully designed to be resilient to failures in the ever changing Grid environment.
The DIRAC framework can be profitably used to manage all the SuperB needs in terms of Grid infrastructure and management:

- Data and Workload management
  - Production activities (simulation, reconstruction, re-processing)
  - User data analysis
  - Data storage, replication, movement, catalogue, integrity check, FTS transfers [12]
  - Software distribution
  - Monitoring and statistics
- Single point to control/manage entire system (e.g. Sites, CEs, SEs, File Catalogues, software components)
- Native use of pilot job
- Two File Catalogues:
  - LHC File Catalogue (LFC) [13]
  - DIRAC File Catalogue (DFC) [14]
- Easily extensible via Agents and Services (highly customizable)
- Interaction gLite and OSG Grid flavour
- Interaction with cloud (via VMDIRAC module)
- Ganga can use DIRAC as backend for job submission

The documentation about DIRAC working principles and its components can be found at DIRAC project web site [10]. Not all the aspects have covered in such a documentation repository, but it is sufficient to cope server and client installation adnd configuration.

DIRAC framework is mainly developed above gLite (and now EMI2) middleware. Considering the EMI2 current product list [15], DIRAC components cover the management of BDII queries, CREAM CE job operations, EMI-UI interaction, FTS operations, GFAL/lcg Util commands, gLExec job running, LFC management, Storm-SE data operations and VOMS interactions.

Started inside LHCb community, today several VOs and computing center are using DIRAC (see table 1) and some of those are also actively involved in its development.

Figure 2. working principle of pilot jobs
4. Testbed setup
In order to evaluate whether the characteristics of DIRAC are able to fulfil the Super\(B\) computing model needs, a test environment was configured at INFN-T1 and INFN-BARI sites.

Three Virtual Machines (VM), equipped with Scientific Linux (SL) and hosting a User Interface (UI), were configured as DIRAC servers (see table 2).

The following distributed resources have been configured to be DIRAC enabled.

- 23 gLite sites
- 1 OSG site
- 50 CEs
- 19 SEs
- 1 DIRAC File Catalogue

Main goal of the test was the evaluation of job and data management DIRAC capabilities. The following DIRAC components have been installed and configured:

**Framework:** Core system of DIRAC

**Data Management:** Storage elements and data management, file catalogue, FTS transfers

**Workload Management:** Jobs management

**Accounting:** Monitoring functionalities

**Configuration:** Computing elements automatic discovery and configuration
First test was conducted using DIRAC v5rX, but since September 2011 the new v6rX release is under test.  

DIRAC v6rX introduces several enhancements, some of these in accomplishment with features requested by our group:

- introduced concept of “project” to easier build a customized DIRAC version
- single DIRAC installation can manage several VOs
- more configurable parameters
- define local SEs for sites
- improvements in proxy management (autoupload, autorenew, automatic alerting)
- maximum number of parametric jobs configurable
- multiple File Catalogues
- more depth levels in DIRAC File Catalogue
- DFC and LFC interoperability
- improvements in Resource Status System
- improvements in FTS transfers
- increased pilots efficiency

Initially documentation available was very poor, but thanks to strong interaction with DIRAC’s developers and recently improvements in technical documentation, several installation procedures were performed and a detailed documentation specific for SuperB was written, documenting installation and configuration procedures, including configuration files.

Configuring a computing element in DIRAC is extremely simple because a dedicated agent performs all discovery and configuration action, so admin has only to instruct DIRAC with two commands, one to add a site and another to enable site:

adding a site:

```
dirac-admin-add-site LCG.INFN-BARI.it INFN-BARI cream-ce-2.ba.infn.it
  cream-ce-1.ba.infn.it Grid-ce-01.ba.infn.it
```

enabling a site:

```
dirac-admin-allow-site LCG.INFN-BARI.it 'added INFN-BARI'
```

Storage elements and VOMS servers can be easily configured via web interface provided by DIRAC. DIRAC user management permits a fine grain permission configuration based on groups and every group can be mapped via VOMS attributes. Both INFN-T1 and INFN-PADOVA VOMS servers were configured, wms-multi.Grid.cnaf.infn.it was used as WMS server in the testbed.

INFN-T1 has two endpoints in SuperB dedicated storage elements due to different write permission setup: in DIRAC these two endpoints were configured as two different storage elements at INFN-T1 with groups permission to use them driven by VOMS role attribute.

To enable filling mode it is necessary to install a dedicated agent that enables this feature. To use DIRAC in test phase the selected machines were adequate. During the tests up to 10k jobs were submitted and more than 2k jobs were running simultaneously.
5. Analysis test

The first use case that we ported into DIRAC is the distributed analysis. The analysis use case was implemented using a real users application reading Monte Carlo data that performs an analysis on \(B^+ \rightarrow K^+\pi^-n\) and \(B_0 \rightarrow K^0_s\pi^+n\) with \(K^0_s \rightarrow K^+\pi^-\) channel. The output of the application is a small ntuple file plus postscript file with the resulting plot. This application is compiled using standard ROOT [16] libraries, and the SuperB libraries. During the test we exploited the libraries already installed in the Grid software area in each of the farms used for the test. This simplified the job submission, as we should not install the needed software for each job. The DIRAC File Catalogue (DFC) was exploited in this test. The Dataset used for the test is quite small; indeed a dataset of about 200GB of root files was exploited, divided over about 1260 different files. Each of the files contained in the dataset was quite small in size and this could introduce a big inefficiency in dealing with Grid data transfers. In order to avoid this we packed 10 files together and registered the packed files into DFC. The full dataset (only packed files) was replicated into two different Grid site (INFN-TIER1- CNAF e INFN-Pisa), and the file catalogue is updated accordingly. This was done in order to emulate a production environment where the data are distributed on several sites and the jobs submitted by the users could run in more than one site. One analysis job was submitted for each of the packed files, this means that each job processed one input file of about 1.6Gbyte and produced 3 output files (1 root file plus 2 text files). In order to test the complete set of features provided by DIRAC the automatic procedure to copy the input and the output files was configured. Moreover DIRAC was configured in order to copy the input files using the close SE (the SE of the site where the job is executed). The stage output procedure is configured in order to use up to three different storage elements in case of failure. In fact, DIRAC is able to automatically take care of failures in the stage input or output using a different storage provided by the user. This is completely transparent to the end-users as the files are accessed using the Logical File Name on the catalogue that does not depend on the physical location of the file. The user should not learn how to copy files from/to Grid storage elements, but it should only configure in the JDL file, the DIRAC File Catalogue Logical File Name for the input file and the name of the expected output files together with a list of SEs for staging the output. In order to analyse the whole dataset 126 jobs were created using the features provided by DIRAC called parametric job. Using this feature in fact the user is able to submit a unique job to DIRAC, and the server will produce all the needed files in order to process all the requested files. The user could also ask for the status in a very easy way: asking for one single job, the user will know the status of each sub-job. The results of the execution was that the jobs where submitted by DIRAC on both site that hosts the dataset: 60% of the jobs were executed on INFN-CNAF-Tier1 while the 40% of them were executed on the INFN-Pisa farm. In order to complete the analysis job the user should use a classical bash script that takes care only of:

- Untar the input file
- Set-up the needed environmental variable in order to use the SuperB and ROOT libraries
- Run the executable itself with the correct parameters

Thanks to the fail-over solutions that DIRAC provides the final user do not see any failures and all the jobs were completed successfully.

6. MonteCarlo production test

Aiming to measure DIRAC performance in a production-like task, a comparison test was performed to compare the current production stack (WebUI [17]) and DIRAC.

Test involved 10 sites, submitting 100 jobs for each site, for a total of 1,000 jobs submitted. Each jobs simulated 10,000 events for a running time of about 2-3 hours. Every job has 5 input
The test result is encouraging, because although the DIRAC configuration has not been optimized for a Monte Carlo production, DIRAC reached a performance comparable to the ones reached by WebUI production system.

Some considerations on the outcome of the test about DIRAC
- Up to 753 jobs simultaneously running
80 jobs failed at IN2P3-CC due to wrong CPUTime value definition in JDL file

All output data stored at INFN-T1 SE and registered into DFC

LFN:/superbvo.org/test_dir/output_testbed/<job_output_dir>/<file_name>.root

The DIRAC capabilities in stage out operations have been successfully tested. In case of transfer failures DIRAC fallback mechanism works properly

7. Data management test

DFC was tested in order to understand the available features that DIRAC framework provides to users in the field of data management. DIRAC File Catalogue provides a set of advanced functionality with respect to the LFC on file metadata administration. All the SuperB Storage Elements were configured, setting the contact point where SRM service is available and the path to use. Each of the storage element configured in Dirac is mapped to the site name. In this way the end user could easily refer to each of the SE when copying or replicating the data. Firstly standard features were tested: adding, replicating and deleting files. The DFC provides simple interfaces that allow the end user to add a file to the catalogue at a given SE, replicate files on different SE and retrieve file’s replica status. DFC automatically stores a complete and detailed set of metadata information per file. This is a really interesting feature as it is automatic and transparent to the end-user. DFC gives also the possibility to use an interactive shell in order to browse the file catalogue content and to exploit all the available features. The last advanced features tested on DIRAC file catalogue is about the capability to deal with end user metadata and file ancestor. The end user could easily tag a directory with arbitrary metadata moreover file searches can be performed using SQL-like queries on metadata. Massive data transfer is a very important task in SuperB computing model and DIRAC FTS capabilities are expected to fulfil the experiment requirements. At present time, there are three FTS servers configured for SuperB VO: INFN-T1, IN2P3-CC and RAL-LCG2. FTS servers and working channels have been shown in figure 5. DIRAC framework were successfully tested in transferring data to/from the highlighted channels.

Managing FTS transfers via DIRAC offers some interesting features:

• automatic catalogue updated once FTS transfer is executed
• automatic proxy and myproxy server functionality management
• simplified FTS configuration management, e.g.: user set target and source SE using their name, not their url
• source file passed using its LFN, not PFN or SURL

FTS transfer shell commands are easier via DIRAC rather than via gLite standard CLI. Two diverse examples have been presented here to show different commands used via DIRAC or via gLite CLI in order to perform the same FTS transfer:

DIRAC :

dirac-dms-fts-submit /superbvo.org/data/datasetsXXX.root
INFN-T1_ANALYSIS GridPP
8. Filling mode test

Filling mode is one of the more interesting features of DIRAC. Filling mode permits a single pilot job, during its execution life on a worker node, to execute more than one user’s job (payload job). Figure 6 shows the number of payload jobs vs pilot jobs executed on each site using filling mode strategy. The filling mode feature has been enabled using default configuration in the submission of 10000 jobs. The maximum number of payload jobs accepted per single pilot is five. The adopted pilot submission strategy is the default one: the number of pilot job submissions is the double of the number of payload jobs submitted by the user. Job payload duration is 3600 seconds. The number worker nodes involved in the test is 2005.

To measure gains offered by this feature, a specific test was performed, comparing results obtained enabling and disabling filling mode. Test was performed three times, submitting at same sites 100, 1000 and 10000 jobs, the results are shown in figures 7, 8 and 9. As can be expected, enabling filling mode permits a significant decrease in total jobs execution time, this effect is more evident increasing the number of submitted jobs. Filling mode capability permits a more efficient usage of distributed resources both from Grid infrastructure and VO point of view. A great advantage of pilot jobs is the smaller time spent in matching computing elements with jobs requirements.
Figure 6. Filling mode efficiency graph per site

Figure 7. Run with 100 jobs
Figure 8. Run with 1000 jobs

Figure 9. Run with 10,000 jobs
9. Conclusions and future works

The SuperB experiment at present time is in an intense R&D phase, that will provide the needed experience and information to write a Computing TDR, that is expected to be released around middle of 2013. The activities carried on within the SuperB experiment in this phase are directed both in the direction of writing missing piece of code and in evaluating already available solution that could be of help in order to solve SuperB use-cases. In this phase of the life of a new collaboration it is important to evaluate all the available solutions as there are at least the LHC collaboration that has similar use-cases and that are producing huge amount of good solutions for many of the problems that SuperB is facing. The job submission and data management handling is surely two of the areas where already available software like DIRAC could be of help. Moreover the DIRAC framework is used at the moment by LHCb experiment and the roadmap of the tool is surely interesting for a Virtual Organization like SuperB that will use a geographically distributed Grid infrastructure in order to solve the computational problems of an HEP Experiment that will produce a huge amount of data in the next years. Giving those consideration, the work that we have carried on, already give us a good feeling on the capabilities of the DIRAC framework, and we will go on testing all the features that it makes available, in order to have the opportunity to understand if this tool could solve the use cases of the DIRAC community. One of the most important future test will be the DIRAC capability to handle massive data transfer among Grid sites. In this case for example DIRAC will be compared to PHEDex [18] that is the CMS tool for massive data transfer.

References

[1] The SuperB Collaboration, SuperB Progress Report, Detector, http://arxiv.org/abs/1007.4241v1
[2] http://lhc.web.cern.ch/lhc/
[3] http://www.linearcollider.org/
[4] The ATLAS Collaboration, ATLAS Detector and Physics Performance Technical Design Report, http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/TDR/access.html.
[5] The CMS Collaboration, CMS Detector Technical Design Report, http://cmsdoc.cern.ch/cms/cpt/tdr/.
[6] http://www.egi.eu
[7] http://www.opensciencegrid.org
[8] http://lcg.web.cern.ch/LCG
[9] Donvito G, 2nd SuperB Collaboration Meeting @ INFN-LNF, 2011, Frascati (Italy) http://agenda.infn.it/contributionDisplay.py?contribId=201&sessionId=37&confId=4107
[10] http://diracgrid.org
[11] http://lhcb.web.cern.ch/lhcb/
[12] https://twiki.cern.ch/twiki/bin/view/EGEE/GLiteFTS
[13] https://twiki.cern.ch/twiki/bin/view/LCG/LfcAdminGuide
[14] https://github.com/DIRACGrid/DIRAC/wiki/DataManagementAdvanced
[15] http://www.eu-emi.eu/products
[16] http://root.cern.ch/drupal/
[17] SuperB Simulation Production System http://indico.cern.ch/contributionDisplay.py?contribId=301&confId=149557
[18] https://cmsweb.cern.ch/phedex/