Università degli Studi di Ferrara

DOTTORATO DI RICERCA IN
MATEMATICA E INFORMATICA

CICLO XXVI

COORDINATORE Prof. Mella Massimiliano

A Web portal to simplify the scientific communities in using Grid and Cloud resources

Settore Scientifico Disciplinare INF/01

Dottorando
Dott. **BENCIVENNI MARCO**

Tutore
Prof**. LUPPI ELEONORA**

Anni 2011/2013

Contents

List of Figures

List of Tables

Introduction

The modern scientific applications demand increasing availability of computing and storage resources in order to collect and analyze big volume of data that often the single laboratories are not able to provide. In the last decade, distributed computing models such as Grid and Cloud have proved to be a valid and effective solution. A proof is the Grid model, widely used in the high energy physics experiments. In this scenario also Clouds are showing an increasing acceptance. Although the Grid has been recognized as a valid solution to allow the sharing of computing and storage resources it was restricted principally by physicists users due to its intrinsic difficulty.
The first obstacle to deal with in the use of Distributed Computing Infrastructures (DCIs) is the robustness of Authentication and Authorization (AA) mechanisms. Many user communities complain about the difficulty of handling digital certificates, the mechanism used in Grid to provide a secure computational environment, and the complexity of the Grid middleware. Those factors, along with the steep learning curve, have undermined the wide adoption of those kind of services.
In the last three years the Italian Grid Initiative (IGI) has started a process of divulgation and support in order to extend the use of these resources to all discipline, but also activities for simplifying the access to Grid resources, covering both the access step and the use of the middleware components. In particular it has developed a Web portal that aims at facilitating the usage of Grid and Cloud. It hides the complexities and acts as an unique access point for different services such as: federated authentication, job submission, workflows, data management in order to make easier the distributed computing resources usage to name a few. Being the Cloud natively easier than Grid, the majority of the services developed is relative to the Grid, especially for the authentication and authorization mechanisms that are the most difficult aspects for the new Grid users.
In order to develop a powerful but sustainable solution, the Web portal has been

connected to some already existing services for the workload management or others internally developed for the authentication and authorization operations and for the data management. This type of solution has two main advantages: the users see only a high level interface hiding them the underlying complexity and the project is sustainable because it relies on several existing services, widely used and well maintained. Our works consists in: configuring correctly the portal and the services used, interconnecting the portal with the external services and developing new ones when needed. Pieces of software are developed only if other valid solutions do not exist or can not be used.
Fig. 1 shows the homepage of the Web portal developed.
The portal URL is: `https://portal.italiangrid.it`

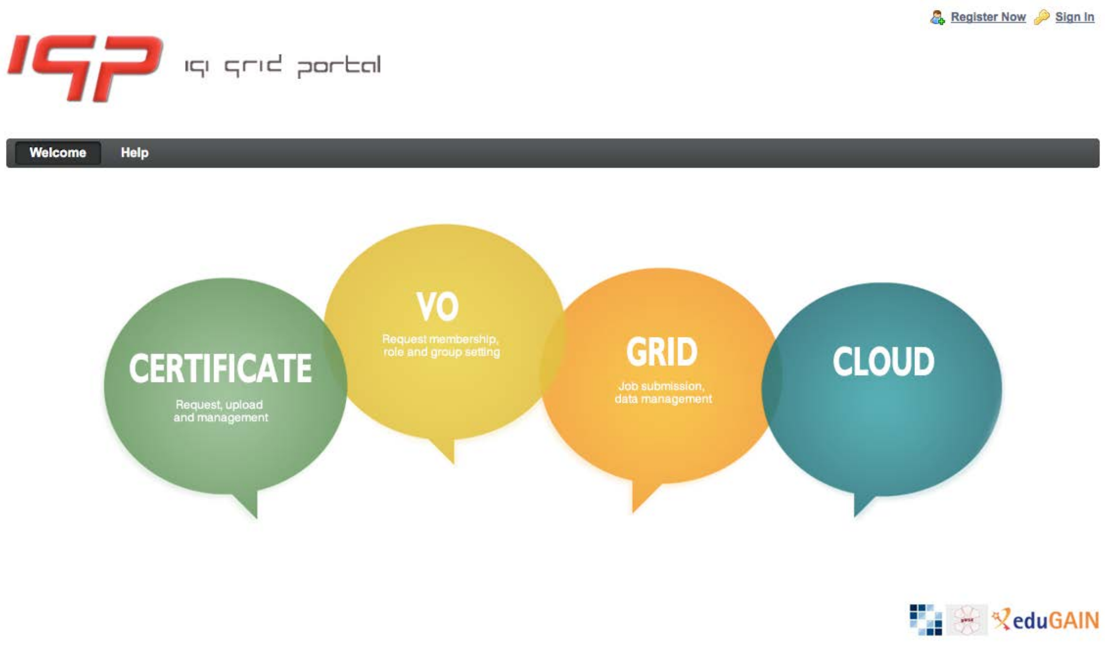

Register Now
Sign In
IGP
igi grid portal
Welcome
Help
CERTIFICATE
Request, upload and management
VO
Request membership, role and group setting
GRID
Job submission, data management
CLOUD
eduGAIN

FIGURE 1: Portal homepage

In *Chapter 1* we focus the attention on the paradigm of distributed computing, in particular Grid and Cloud infrastructures. The Grid, being at the moment the most usable infrastructure through the portal interface, it is introduced with more emphasis, especially the services that have a main role in the portal architecture. It is described also the Cloud definition and some implementations because the future works on the portal will be oriented in this direction.
Chapter 2 we discuss the authorization and authentication mechanisms implemented in the portal in order to hide the user from these complex aspects. We also explain a study to integrate an online Certification Authority (CA) in the

portal architecture as a proof of concept.

In *Chapter 3* it is described which external services have been integrated in the portal architecture to perform different type of computations: jobs, workflows and specific applications.

In *Chapter 4* we focus on the data management aspects and the solution developed in order to avoid the users to learn difficult commands to transfer data from their PC to Grid and viceversa.

In *Chapter 5* we give some examples how we have used the portal for specific applications, developing scripts and interfaces ad hoc for some communities in order to hide the Grid complexity as much as possible.

In *Chapter 6* we explain the services that we are going to use in order to interface the portal with the Cloud infrastructures. Two different use cases have been studied: the Cloud resources used in an interactive way or used as computation resources on demand for the job execution.

In the appendixes are shown pieces of code developed for different purposes: the appendix A shows the bash script used to run an application in Grid. In the appendix B there is the code for moving files or directory from the Grid to different remote storage servers using several protocols. In appendix C is shown the DIRAC configuration used to support multi VOs. Moreover in appendix D is shown the portal logical architecture with all the components described in the thesis.

Chapter 1

Grid and Cloud architecture

1.1 Authentication

The authentication is a fundamental point both for Grid and Cloud because it concerns the security aspects. The European Grid Infrastructure is composed of more than 350 Grid sites distributed all over Europe and for these reasons it is obvious that the security is one of the most important topic because a security breach at one of the Grid site can affect a huge number of other sites. Therefore a robust Authentication mechanism is a fundamental component in Grid.
In Grid the authentication mechanism is based on X.509 digital certificates, defined by RFC 5280. In Cloud the authentication method is not univocally defined, it could be used the X.509 certificates [1] yet but also other solutions as the federated identity. In next paragraphs we introduce the basic concepts and services useful to better understand the following chapters. In particular we are going to explain the Authentication mechanism in Grid and the concepts of Identity Federations.

1.1.1 X.509 Certificate

A public key certificate (also known as a digital certificate) is an electronic document used to identify an individual, a server or some other entity. In this way it is possible to associate an identity with a public key. In cryptography the asymmetric encryption involves a public key and a private key associated with an entity that needs to authenticate its identity electronically. Each public key is published,

and the corresponding private key is kept secret. Data encrypted with your public key can be decrypted only with the corresponding private key[1].
The X.509 is the standard for the digital certificate, it combines a Distinguished Name (DN) with a Public key; the DN is a collection of information about a person in a determined context. These information are in the key/value pair format: where the values depend on the CA which signs the certificate itself. The typical fields of a Distinguish Name in a X.509 certificate are: CN: Common Name, O: Organization, OU: Organization Unit, C: Country, ST: State Or Province Name, L: Locality. In Fig. 1.1 is shown the X.509 structure. It follows a brief description of the fields:

- *Version* is a integer value; the possible alternatives are:
  - 0. default (v1)
  - 1. if present "Issuer unique identifier" or "Subject unique identifier" (v2)
  - 2. in case of extensions (v3)
- *Serial number* in an integer value, unique for each CA; it identifies unambiguously the certificate.
- *Signature algorithm ID*: it represents the algorithm and the hash function used by CA to signs the certificate (e.g. md5WithRSAEncryption, sha-1WithRSAEncryption).
- *Issuer name*: Distinguish Name of the CA which has signed the certificate.
- *Validity period*: it contains the date in which the certificate starts to be valid and the expiration date.
- *Subject name*: the Distinguished Name of the certificate's owner (who has the corrispective private key).
- *Subject's public key information*: Public key of the certificate's owner and relative algorithm (es. rsaEncryption).
- *Issuer unique identifier*: it is used to distinguish univoquely the CA if the same DN is used in two distinct CAs.

[1]https://developer.mozilla.org/en-US/docs/Introduction_to_Public-Key_Cryptography

- *Subject unique identifier*: it is used to distinguish univoquely the certificate's owner if the DN (of the user) it is reused.
- *Extensions*: the extension are divided in 3 categories:
  - key and policy information, for example the certificate utilization scopes,
  - subject and issuer attributes, for example the subject/issuer alternative name,
  - certification path constraints, for example if the subject can act as a CA.

The hash of these fields is signed by the CA private key.

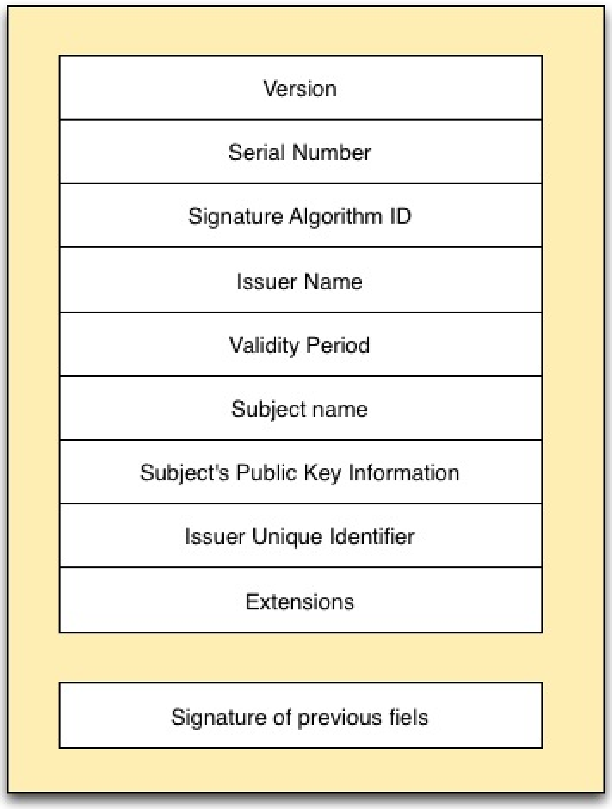
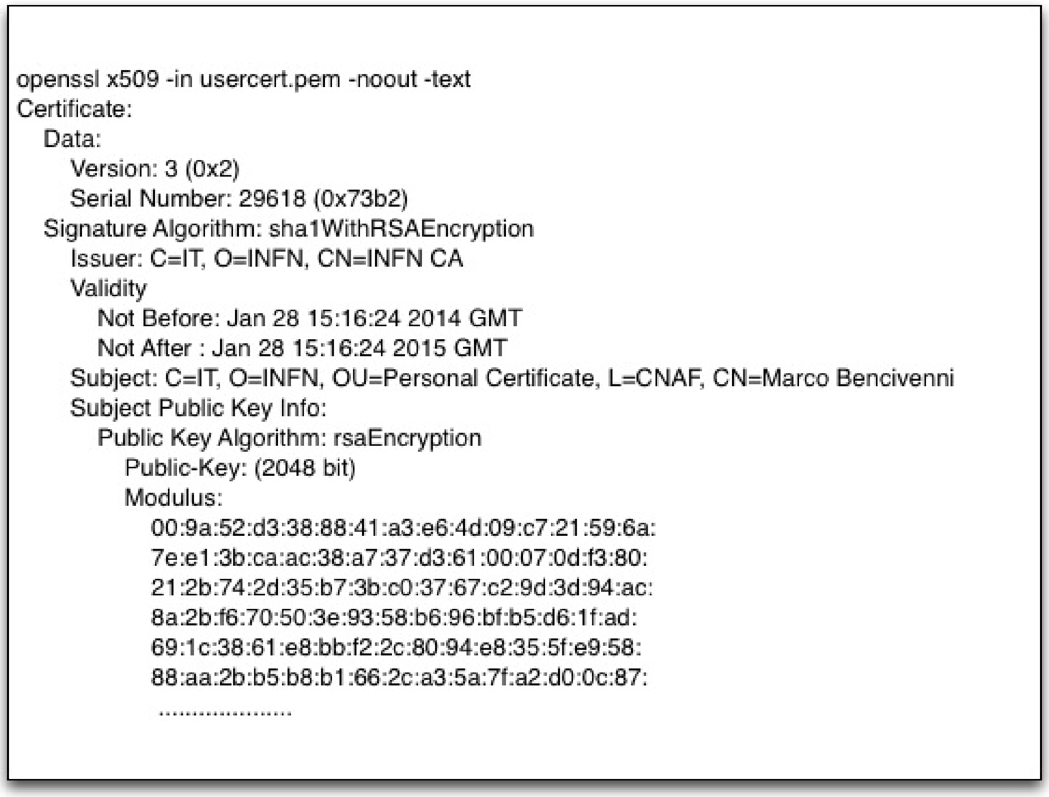


FIGURE 1.1: X.509 Certificate structure

In Fig. 1.2 is shown an example of a certificate. After obtaining a personal X.509 certificate, the user have to store it according to some basic security rules defined by their CA, including:

- encrypting their certificates by means of an appropriate passphrase;

- storing the encrypted private keys on a local file system only;
- applying for the certificates renewal near the expiration date.



```
openssl x509 -in usercert.pem -noout -text
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 29618 (0x73b2)
  Signature Algorithm: sha1WithRSAEncryption
    Issuer: C=IT, O=INFN, CN=INFN CA
    Validity
      Not Before: Jan 28 15:16:24 2014 GMT
      Not After : Jan 28 15:16:24 2015 GMT
    Subject: C=IT, O=INFN, OU=Personal Certificate, L=CNAF, CN=Marco Bencivenni
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
          00:9a:52:d3:38:88:41:a3:e6:4d:09:c7:21:59:6a:
          7e:e1:3b:ca:ac:38:a7:37:d3:61:00:07:0d:f3:80:
          21:2b:74:2d:35:b7:3b:c0:37:67:c2:9d:3d:94:ac:
          8a:2b:f6:70:50:3e:93:58:b6:96:bf:b5:d6:1f:ad:
          69:1c:38:61:e8:bb:f2:2c:80:94:e8:35:5f:e9:58:
          88:aa:2b:b5:b8:b1:66:2c:a3:5a:7f:a2:d0:0c:87:
          ...................
```

Figure 1.2: X.509 Certificate example

1.1.1.1 Certification Authority

A Certification Authority is a third party entity, public or private, qualified to issue distribute and revoke digital certificates X.509 following certification procedures established by international regulations. The authority issues a certificate used to signs the user's certificate. The CA software must run in a specialized and safe hardware with the highest physical and logical security measures. Any CA has a Registration Authority (RA) which is the system for the registration and authentication of the users who are able to ask for a certificate.
A CA can be of 2 types: online or offline. The offline CA is completely disconnected from the network and the certificate issue procedure need the physical intervention of a CA administrator. The online CA typically has a front-end connected to the

network and a back-end connected only to the front-end. Being this CA connected to the network it is more exposed than the previous one so the security mechanisms requested to protect it are stronger. Through an online CA a user can obtain a certificate on-demand in an automatic way.
The access to the EGI infrastructure is restricted to certificates provided by CAs accredited to the International Grid Trust Forum (IGTF[2]) federation. The IGTF role is to establish common policies and guidelines among its Policy Management Authorities (PMAs) members. EUGridPMA[3] is the international organisation coordinating the trust fabric for e-Science authentication in Europe.
According to the IGTF rules it can exist one IGTF-accredited CA per country per type: for example one CA offline e one CA online in Italy. Moreover the IGTF accepts 2 types of certificate profile:

- Short Living Credential Service (SLCS) SLCS: the certificates issued by a SLCS CA have a maximum lifetime of 11 days.
- Member Integrated Credential Services (MICS) [13]: the certificates issued by a SLCS CA have a maximum lifetime of 13 months.

The IGTF requirements for an on line CA are illustrated in the document for the MICS profile. In particular, the user's primary identity vetting must be performed de visu, the assessment of the type of vetting done by the users home institutions is done through a specific attribute (eduPersonEntitlement) released by the IdPs. Moreover, the documents specifies that a certificate can be issued only upon an explicit user's request, that all communications are secure and that the user's private key is kept on the server for the time strictly needed to complete the process. Two types of certificate can be used in a web portal: robot or personal. In the next paragraphs we explain the difference between them.

1.1.1.2 Robot certificate

The Robot Certificate has been introduced in order to allow unattended services to perform automated Grid tasks on behalf of a person. A Robot Certificate is uniquely associated to a specific application and VO. A typical usage is inside a Web Portal where a group of people submits jobs using the same Robot Certificate.

[2]http://www.igtf.net
[3] http://www.euGridpma.org

The certificate subject must be the string “Robot” followed by alphanumeric chars without spaces in order to identify the certificate type and in the commonName must be specified the Distinguished Name of the certificate owner. In this way all the job submitted with the same Robot Certificate [2] belong to the same person even if submitted from other people.
Being that a large number of users could performs Grid actions using a single Robot Certificate: if a security incident happens, the robot certificate revocation could have a very high impact because all its users would be unable to use the Grid resources until a new Robot Certificate is generated and associated to the application and VO.
As specified in the EGI Traceability and Logging Policy document [6], the portals which use Robot Certificates need to provide information on who is doing what at any time. They implement a mechanism to continuously track the real identity of the user that is performing Grid operations. Moreover it has to store these information in order to exactly identify the malicious user’s identity in case of security incident.

1.1.1.3 Personal Certificate

A Personal Certificate is a certificate issued to a person by a Certificate Authority only after a de visu identification. In order to perform Grid operations the user must also be registered into one or more VOs. The name reported in the certificate identifies precisely the certificate owner and the certificate can not be used by other people. The portals which use this type of credential do not have to implement a mechanism for identity traceability because users are already using their personal credential so, in case of security accident, only the specified user is banned from Grid.

1.1.2 Proxy

As we have explained above, Grid users have to provide a certificate in order to prove their identity. Anyway, in Grid it is often necessary that a service acts on behalf of the user, this mechanism is known as delegation. Instead of sending the user’s certificate on the net, it is used a proxy certificate that has limited rights. A proxy is a certificate with a shorter lifetime, typically of 12 hours. In order to create a proxy, a new public/private key pair is created and a new certificate is built. It is signed with the certificate's long-term private key and it contains the public key using a name with the form of the following example:

```
/C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Marco Bencivenni/CN=proxy
```

When a job is submitted, the proxy certificate, the private key for the proxy and the normal certificate (but not the long-term private key) are sent with it. When the job wants to prove its delegated identity to another service, it sends it the proxy certificate and the standard certificate, but not the proxy private key. This information is sufficient to prove that the remote service has the right to use the delegated identity. Proxies usually have a lifetime of only a few hours, so the potential damage is fairly limited[4]. The Fig. 1.3 shows the steps to create a proxy certificate and delegation.

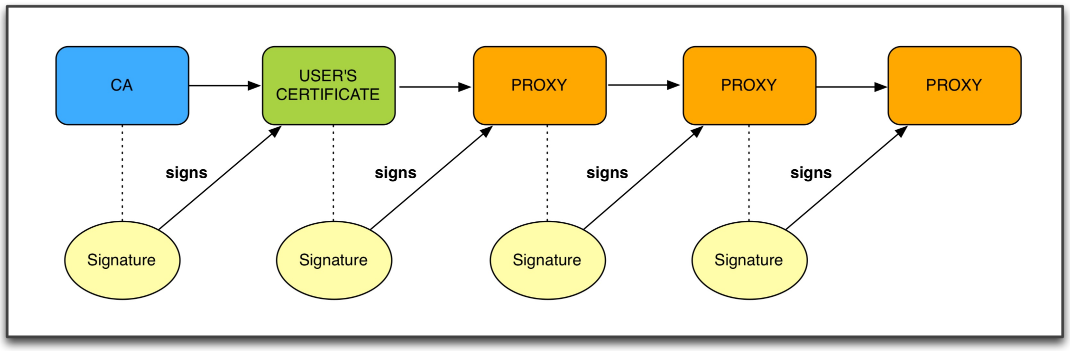


FIGURE 1.3: Grid delegation

1.1.2.1 MyProxy

MyProxy is open source software for managing certificates and private keys. It can be configured to store encrypted private keys with a password chosen by the user [10].
The typical interaction between a MyProxy and a Web Portal is that the portal contacts the MyProxy server to obtain credentials so it can access Grid resources on user's behalf. The users has to enter his username and password on the portal page, these credential are used by the portal to login to MyProxy on user's behalf. In this way the portal is authorized to retrieve user's credential. The schema of this interaction is show in Fig. 1.4.

[4]http://eu-egee-org.web.cern.ch/eu-egee-org/fileadmin/documents/UseCases/ProxyCerts.html

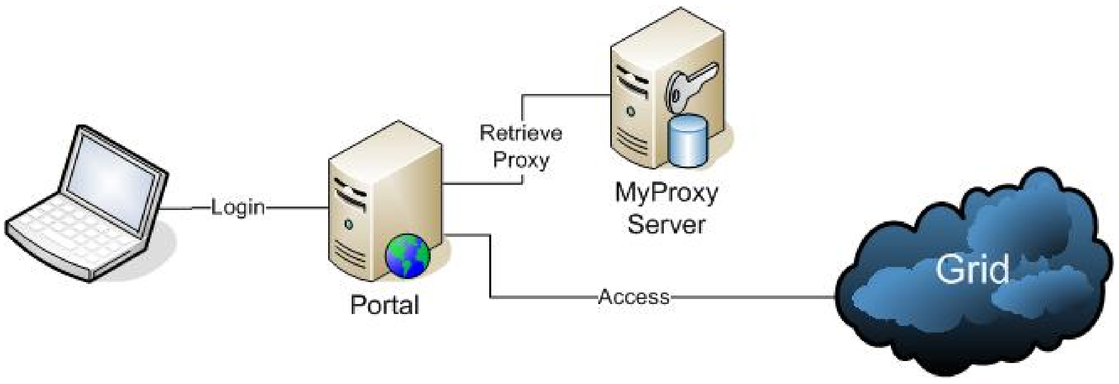


FIGURE 1.4: Interaction between MyProxy and Portal[5]

Another important use of MyProxy is for credentials renewing.
Users can store a proxy credential in the MyProxy Server using the myproxy-init command. Whenever users needs a credential, they can retrieve a short-lived proxy from the MyProxy with a specific command[6]. In this way it is possible to access the credentials without needing to copy the real and long user's certificate and private key between systems, which can be the cause of security problems. The schema of this interaction is show in Fig. 1.5.

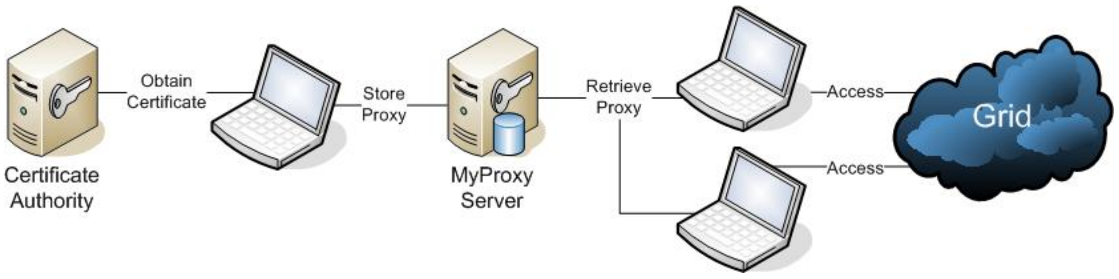


FIGURE 1.5: MyProxy used for proxy renewal

1.1.3 Identity Federation

We are now going to explain another type of authentication not adopted in Grid yet, but used by the portal to authenticate the users. A federation can be considered as an agreement among organizations and resource providers. The organizations mutually trust the information exchanged during the authentication and authorization procedures following detailed rules in order to maintain this trusted relationship.

[6]http://eu-egee-org.web.cern.ch/eu-egee-org/fileadmin/documents/UseCases/BasicDataMgt.html

In order to avoid that the users have to authenticate themselves every time they access a service, an Authentication and Authorization Infrastructure (AAI) has been introduced. The task of this infrastructure is to manage the authentication and authorization procedures among the users, their organization and the services offered introducing the federation identity mechanism.

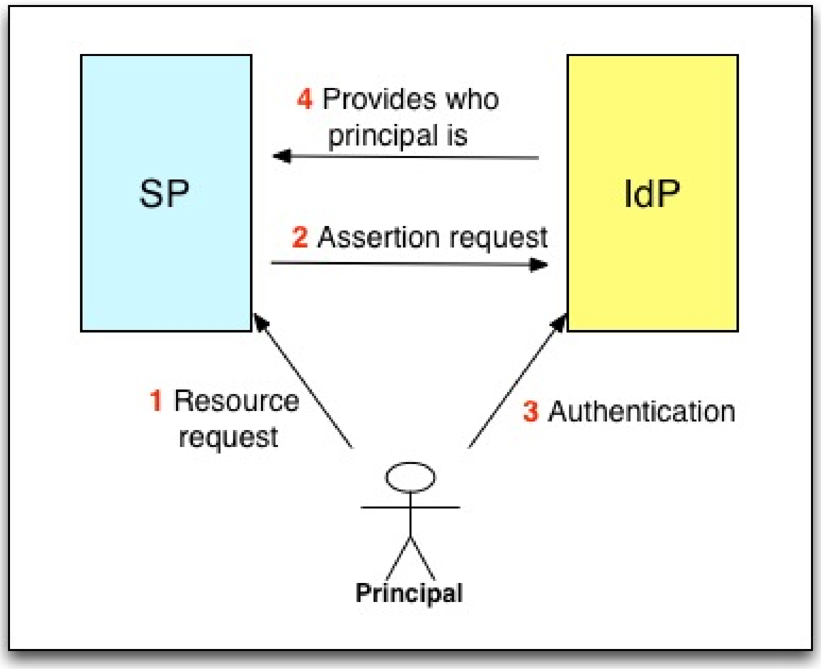


FIGURE 1.6: Identity Federation model

The Security Assertion Markup Language (SAML) is the protocol used for exchanging authentication and authorization data between parties. In particular it is an XML-based open standard data format, and it is used to exchange data between the service that manage the user's identity and a generic service provider. Its principal scope is to provide the Single Sign On (SSO) property.
In the SAML protocol are defined three roles: the principal (the user), the Identity Provider (IdP), and the Service Provider (SP) as shown in Fig. 1.6. The typical scenario is:

1. the principal requests a service from the SP;
2. the SP requests an identity assertion from the IdP;
3. the principal authenticates himself on the IdP which issues an assertion to the SP;

4. the SP, on the basis of this assertion, decided whether the principal can access to the services.

In a federation there are many SPs and IdPs: one IdP may provide SAML assertions to many SPs and viceversa one SP may rely on and trust assertions from many IdPs.
Shibboleth and SimpleSamlPHP are SAML implementations used to build SP and IdP. In particular while Shibboleth is written in Java and uses Tomcat as container in which the IdP runs, SimpleSamlPHP is written in PHP. At present several Identity Federations exist, in the next paragraph we introduce 2 of them because they have been interfaced to the portal.

1.1.3.1 eduGAIN and IDEM

eduGAIN is a service developed within the GÉANT project and its scope is to interconnect identity federations around the world, simplifying access to services and resources for the global research and education community. eduGAIN coordinates the elements of the federations and provides a policy framework that controls this information exchange.

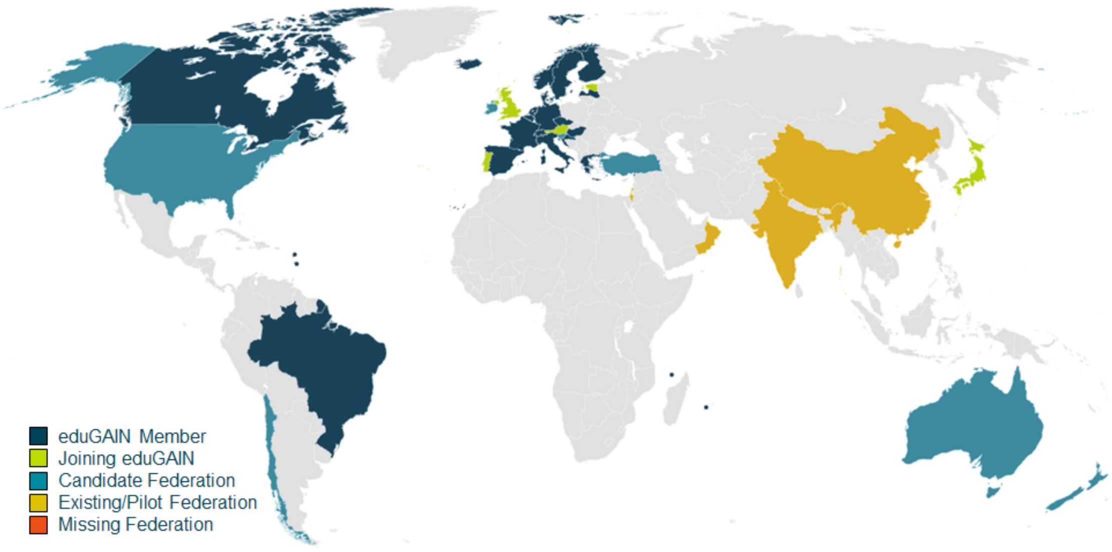


FIGURE 1.7: Map of eduGAIN members[7]

eduGAIN has 17 active participant federations (as shown in Fig. 1.7), one of which is IDEM: the Italian identity federation of universities and research institutes for authentication and authorization. At present IDEM is composed of 74 IdPs and 90 SPs.

1.2 Grid Architecture

The Grid infrastructures were born and are still being used to face one of the most challenging scientific collaborations, such as the ones running the experiments at CERN's Large Hadron Collider (LHC[8]). The Grid relies on advanced software, called middleware, an interface between resources and applications. A typical usage scenario is: many users of different organizations geographically distributed (Virtual Organizations VOs) requesting high computational and storage capacities and collaborating with each others where the computational resources (computing and storage) belongs to different institutions but are transparently accessible. The users join a VO, each VO shares Grid resources with other VOs accordingly to several policies.
There are several middleware distributions: gLite, Arc, Unicore, etc. The rest of this document we consider only the gLite solution which was developed in the context of the Enabling Grids for E-sciencE (EGEE) project [28]. At the end of EGEE, the development has been carried on by the European Middleware Initiative (EMI) [25] as components of the EMI distribution. Now its key services are maintained as independent projects by the interested research institutions.
In order for the user to exploit and share these resources, the gLite middleware is composed of different services. Some of these are:

- *User Interface* (UI): the user entry point.
- *Workload Management System* (WMS): a set of services having the scope to find the best available computing element where to submit user's job.
- *Logging and Bookkeeping* (LB): it keeps track of user job execution in terms of status: Ready, Scheduled, Waiting, Running, Done.
- *Computing Element* (CE): it is the service responsible of the computation tasks. It is the entry point to several types of servers handled by a job queue management system.

[8]http://home.Web.cern.ch/about/accelerators/large-hadron-collider

- *Worker Node* (WN): the servers where jobs are executed and managed by the CE queue management system.
- *Information System* (IS): the service for maintaining information about the available Grid resources and their health status.
- *Virtual Organization Management Service* (VOMS): the service for managing the authorization to use the Grid resources. Furthermore VOMS allows the VO managers to define different access rights to VO's resources.
- *Storage Element* (SE): the service responsible to manage Grid files.
- *LCG File Catalogue* (LFC): it offers a mechanism for users and jobs to easily locate the files stored in the SEs.
- *Site-BDII*: it collects the information gathered by the various services at a site level.
- *Top-BDII*: it collects the information from different sites.

Being some of these services directly connected to the portal or used by it, in the next paragraphs these will be described in more detail.
Fig. 1.8 shows the Grid architecture and the components interaction. Although the Grid Middleware services should hide the users from the underlying infrastructure complexity, it is common belief that these services are not easy to use. The middleware services are usually accessed via the Command Line Interface (CLI) but while this is a trivial tasks for expert users, it may be very complex for not skilled user communities.

1.2.1 VOMS

The VOMS is the Grid system for managing VO user authorization information [11].
VOMS provides a database for sorting users into groups and keeping track of their roles and other attributes. These information are used to issue trusted attribute certificates and VOMS extension which are used in the Grid environment for authorization purposes. Through a command line available in the UI, users can generate a local proxy credential based on the user's information in the VOMS

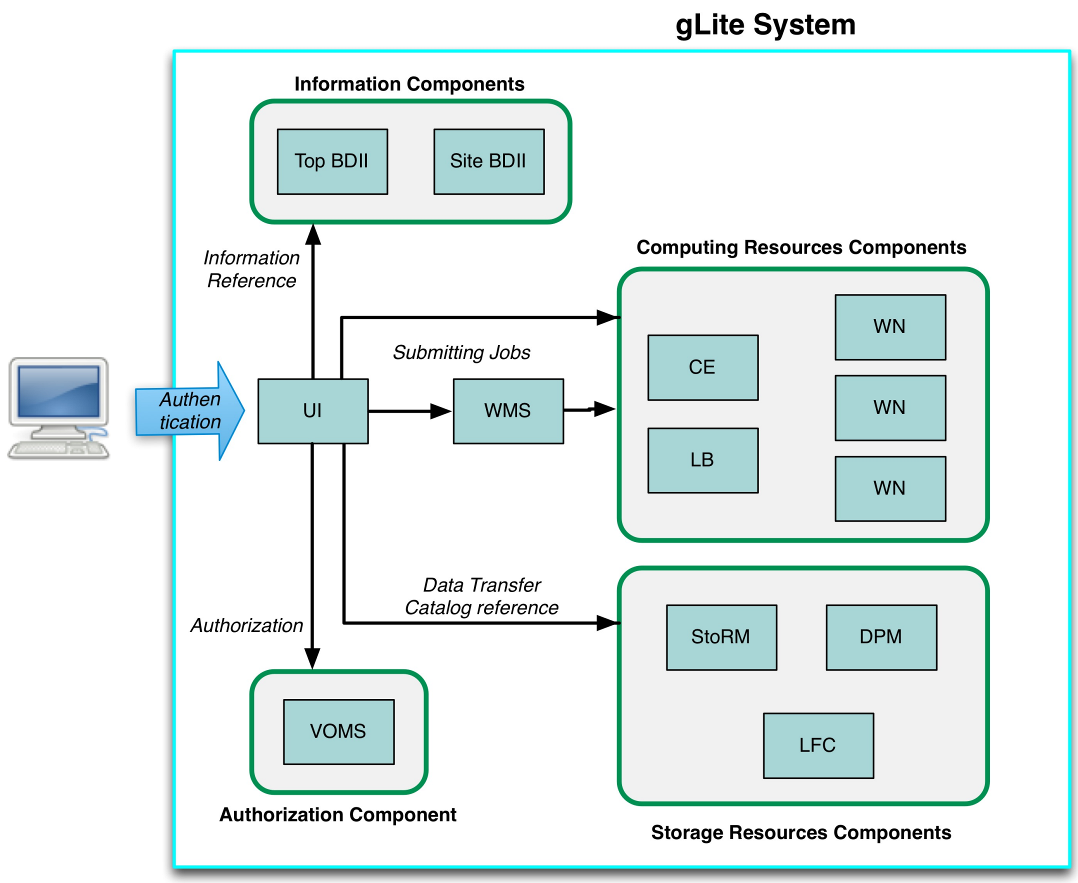


FIGURE 1.8: Grid architecture

database. At the time the proxy is created, one or more VOMS servers are contacted and they return an Attribute Certificate (AC) which is signed by the VO and contains information about group membership and any associated roles within the VO.
The information exchange between Client (UI) and the VOMS Server follows the workflow shown in Fig. 1.9 and described by the following steps:

1. Mutual authentication between client and server.

2. Client sends a signal request to server that checks the user's identity and if the request is syntactically correct.

3. Server signs the authorization information and returns it back. The Client checks the consistency and validity of the information returned, creates a proxy certificate that includes the information returned by the VOMS servers.

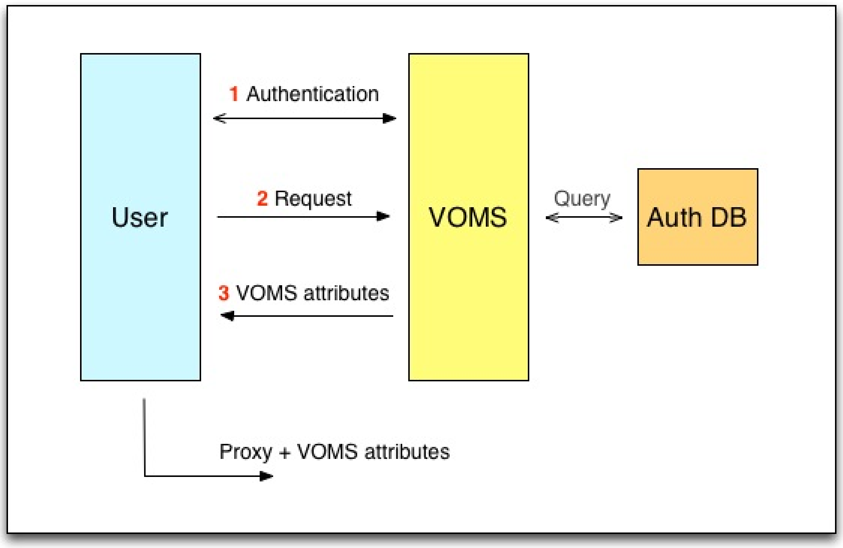


FIGURE 1.9: Interactions between Client and VOMS

At the end of these process the credential includes the basic authentication information (Grid proxy credential) but also group and role information from the VOMS server, as shown in Fig. 1.10. The VO administrator can assign roles to user and manage user's information.

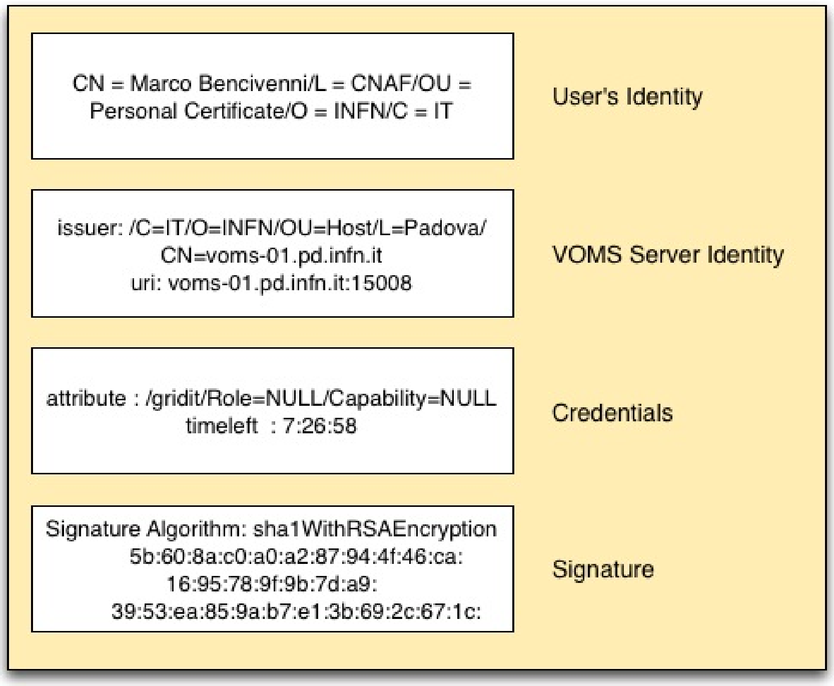

CN = Marco Bencivenni/L = CNAF/OU = Personal Certificate/O = INFN/C = IT
User's Identity
issuer: /C=IT/O=INFN/OU=Host/L=Padova/ CN=voms-01.pd.infn.it
uri: voms-01.pd.infn.it:15008
VOMS Server Identity
attribute : /gridit/Role=NULL/Capability=NULL
timeleft : 7:26:58
Credentials
Signature Algorithm: sha1WithRSAEncryption
5b:60:8a:c0:a0:a2:87:94:4f:46:ca:
16:95:78:9f:9b:7d:a9:
39:53:ea:85:9a:b7:e1:3b:69:2c:67:1c:
Signature

FIGURE 1.10: Proxy with VOMS extension

1.2.2 WMS

The WMS is the middleware component responsible for submitting and managing jobs in Grid.
WMS typically receives requests for a job execution from a client (UI for example) and, on the base of job requirements, finds the most appropriate resources, submits the job to the resource and follows the job submitted until its completion. In case of failure the WMS can also resubmit, if the job description requires it [20]. The WMS is able to handle different job types (shown in Fig. 1.11):

- Batch: single job.
- Directed Acyclic Graphs (DAG): a set of jobs where the input/output/execution of one of more jobs may depend on one or more other jobs.
- Parametric Jobs: multiple jobs with one parameterized variable,
- Collections: multiple jobs with a common description but each one independent from the others.
- Parallel jobs: a set of jobs each one dependent from the others commonly used to handle the communications between tasks in parallel applications.

Jobs are described via a flexible, high-level Job Definition Language (JDL).

1.2.3 Computing Element

CE is the services that provides an uniform interface to the Grid to allow the jobs to be executed in a Local Resource Management System (LRMS), like Load Sharing Facility (LSF), Portable Batch System (PBS) or Sun Grid Engine (SGE). The Computing Resource Execution and Management (CREAM) Service is a lightweight service for job management. CREAM accepts job submission requests and other job management requests (e.g. job cancellation, job monitoring, etc) via WMS or directly from the end-user. The job submitted directly are described with the same JDL language used to describe the jobs submitted to WMS.
Once the job has been received by the CE it is scheduled on the LRMS. It waits there for the proper resources, based on the queue, user privileges, etc to be free and then it is sent to a WN. The WN is essentially a UI that includes commands

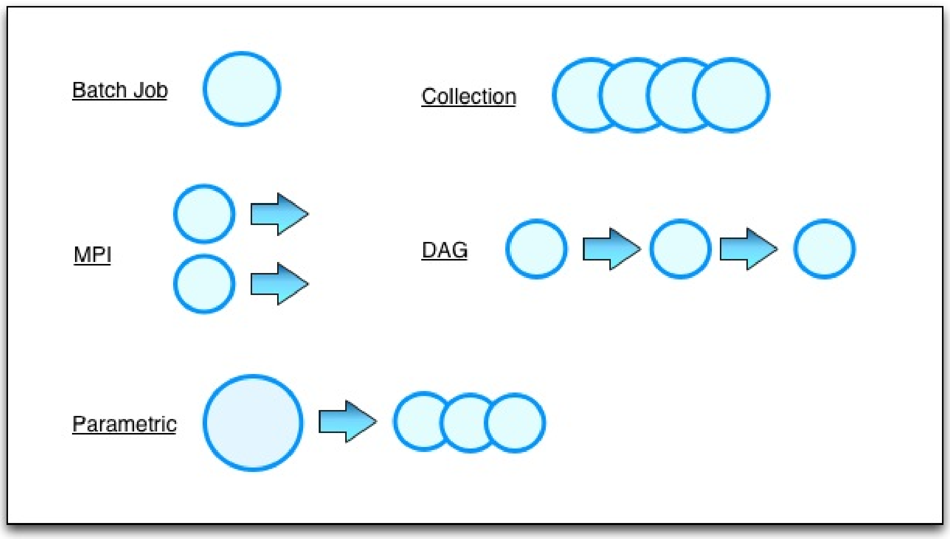


FIGURE 1.11: Job Types

to manipulate the jobs and has access to a software area where the required programs are installed the. The job is executed by mapping an appropriate local user to the Grid user.

1.2.4 Storage Element

A Storage Element is the Grid service that allows the users to store and manage files in the storage resources deployed in the Grid site. Using standard interface it is able to provide the users a uniform interface hiding them from the heterogeneity of the resources: disk servers, Storage Area Network (SAN), tape server, etc.
As shown in Fig. 1.12, in order to make the storage available in a Grid infrastructure, the storage resource is wrapped by the storage service.
In gLite several implementations are available:

- Storage Resource Manager[9] (StoRM): it takes advantage of the parallel file systems to access disk areas [23].
- dCache[10] : by means of disk buffer, a server acts as frontend for a complex mass storage system.

[9]http://italiangrid.github.io/storm/
[10]http://www.dcache.org/

- Disk Pool Manager[11] (DPM): a lightweight solution for disk storage management.

The standard interface of all these implementations is the Storage Resource Manager [21] (SRM) which is responsible of interacting with the underlying storage hiding the implementation details and providing a Grid interface to the outside world. SEs also use standard transfer mechanisms for remote access such as the File Transfer Protocol in Grid Computing Networks[12] (GridFTP) that grants high-performance, secure connection, reliable data transfer, and is optimized for high-bandwidth wide-area. The data in a SE can be accessed through several standard protocols: Remote File I/O (rfio) to directly access the remote files in the SE, Data Link Switching Client Access Protocol[13] (dcap) and file.

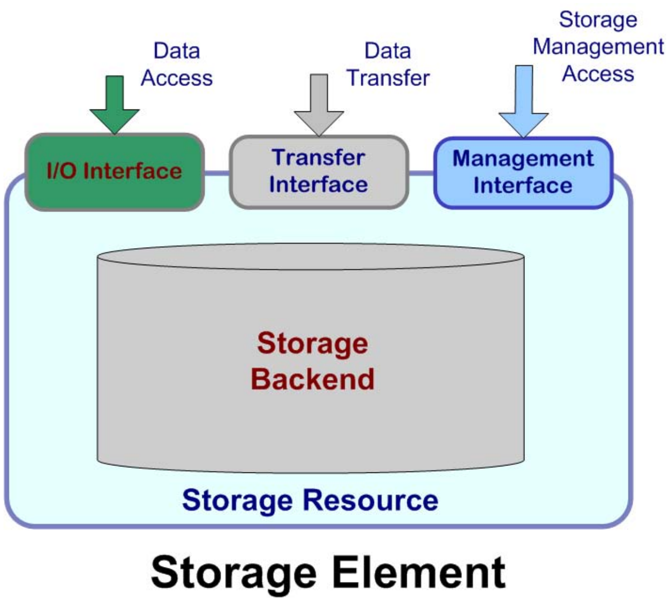


FIGURE 1.12: Storage Element architecture

As well as for the other services, the users must have the valid credentials to access the SE. The system can allow or deny access and manage the space with quotas, prevent the deletion of file with pinning, preallocate a storage space per VO with space reservation and set a specific lifetime on the file.
The Grid service that manage the transfers is the File Transfer Service (FTS) [33]. It interacts with the SRM source and destination using the gridFTP protocol.

[11]https://www.gridpp.ac.uk/wiki/Disk_Pool_Manager
[12]http://www.ogf.org/documents/GFD.47.pdf
[13]http://www.networksorcery.com/enp/rfc/rfc2114.txt

This allows the transfer of huge amount of data using multiple streams, queue of transfers, detection of errors and rollback.

1.2.4.1 StoRM Implementation

StoRM solution has been designed, developed and maintained by the INFN-CNAF development team. It has been designed in order to take advantage of the high performance of parallel file systems used in Grid. Several file system are supported: General Parallel File System[14] (GPFS), Lustre[15], Posix FS[16] etc.
The StoRM architecture (shown in Fig. 1.13) encompasses different elements. It is possible to deploy each one of them in a different server:

- Front-end: it exposes the SRM interface (written in C++).
- Back-end: it executes the SRM requests (written in Java).
- Database: it stores the requested data and StoRM metadata (MySQL based).

1.2.5 LFC

A relevant problem in a distributed architecture as Grid is to uniquely identify a file but also identify replicas of the same file in different location and with different names. Each file in Grid is univocally identified by a Global Unique Identifier (GUID), a non-human-readable fixed-format string that identify an item of data. Each file can be replicated in different locations and each replica is identified by a Site URL (SURL) or Physical File Name (PFN). The Transport URL (TURL) provides the necessary information to access a file in a SE; it is a string composed by the access protocol and the temporary locator of a replica.
To access the desired file the users define the Logical File Name (LFN). The LFN, as the GUID, is independent from the location of the file.
LFC servers are databases containing a list of correspondences between LFN, GUID and SURL. When accessing a SE, the standard protocols use the TURL address, given upon request by the SE itself, to write or retrieve a physical replica.

[14]http://www-03.ibm.com/systems/software/gpfs/
[15] http://users.nccs.gov/ fwang2/papers/lustre_report.pdf
[16] http://www.opengroup.org/austin/papers/posix_faq.html

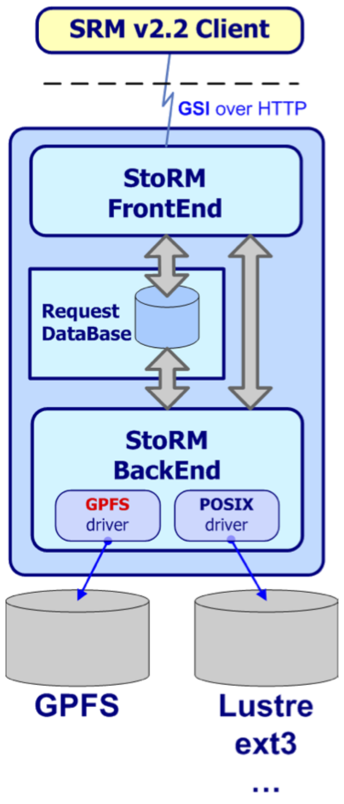


FIGURE 1.13: StoRM architecture

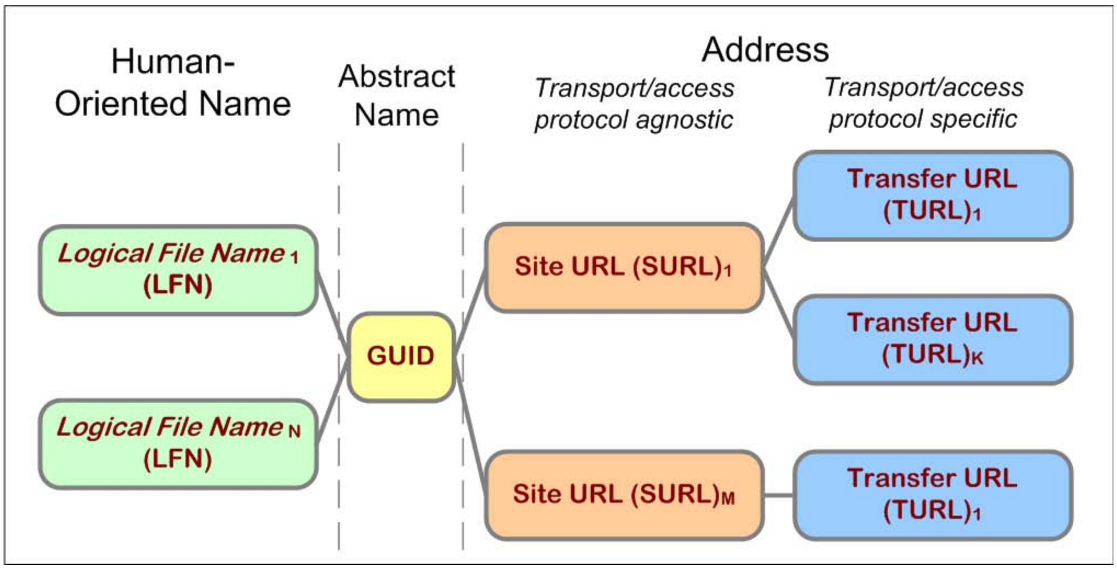


FIGURE 1.14: Grid Data Naming

1.3 Cloud

By definition the Cloud computing is a model to conveniently enable on-demand network access to a shared pool of configurable computing resources (e.g.: networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
The Cloud is another type of distributed computing infrastructure over the net. Using the Cloud, the users connected to a Cloud Provider which can perform operations as use of computing or storage resources and retrieval of data or software through a web browser.
A user can execute remote software and store the resulting data in the online storage made available by the provider.
It is possible to distinguish 3 main topologies of Cloud computing services:

- IaaS (Infrastructure as a Service) - It consists of using remote hardware resources. This type of Cloud is similar to the Grid paradigm but with a main difference: the resources used are available on demand, they are not preassigned independently of their utilization (e.g Amazon EC2).
- PaaS (Platform as a Service) - On a remote server is executed a software platform that could be composed by several services: programs, libraries etc (e.g Google App Engine).
- SaaS (Software as a Service) - It consists of using software installed in a remote server through a web server (e.g Google Apps).

In the IaaS the Cloud provider manages everything from the data centers, network, storage, servers, and operating systems–leaving the customer to manage their own applications and data. In other types of Cloud services, the Cloud provider could manage the entire system all the way up to and including applications and data[17]. Fig. 1.15 shows a high-level view of the provider and customer roles for IaaS, PaaS, and SaaS.

The Cloud Computing system encompasses 3 main actors:

- Cloud provider: the entity who offers the services (virtual servers, storage, applications), usually on the base of a ”pay-per-use” model.

[17] http://mycloudblog7.wordpress.com/2013/04/19/who-manages-cloud-iaas-paas-and-saas-services/

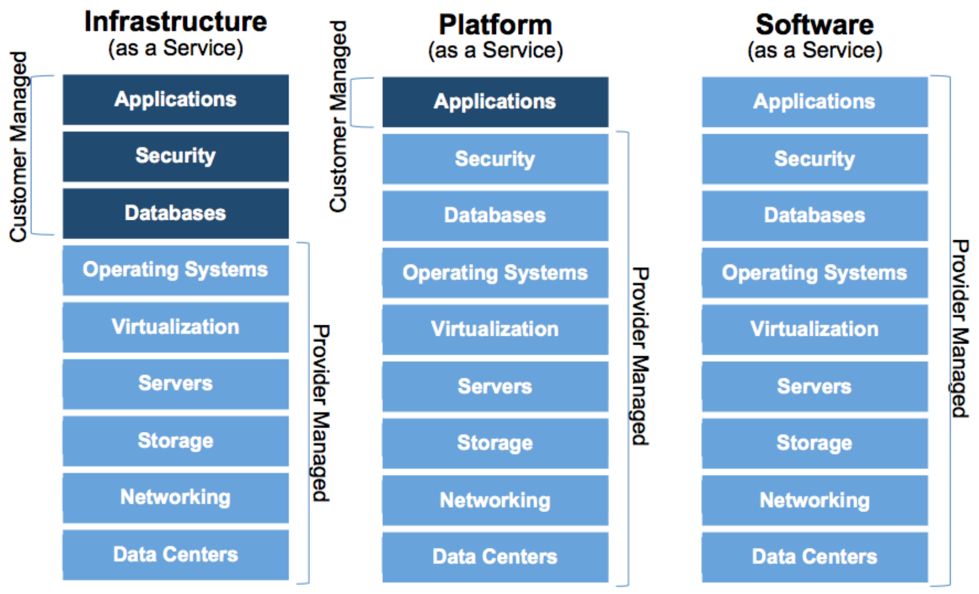


FIGURE 1.15: Cloud Topologies

- Administrator Client: the entity who chooses and configures the services. Usually it is his task to install the necessary software for the final user.
- Final Client: the entity who uses the services opportunely configured by the administrator client.

In some use case the administrator client and the final client could be the same entity: for example a client uses a storage service to backup his data and it is also responsible for configuring the service itself.
From an architectural point of view the Cloud Computing consists of one or more physical servers, generally in a high availability configuration and located in the data center of the service provider.
As shown in Fig. 1.16, the Cloud provider offers interfaces to catalog and manage the available services. The client administrator uses these interfaces to select the requested service (for example a complete virtual server or only the storage component) and to administer it (configuration, activation, deactivation).The client uses the service preconfigured by the administrator. For the final user the features of the physical servers are not relevant.

The Cloud solutions are various, in the next paragraphs we describe only those that the portal has been interfaced to.

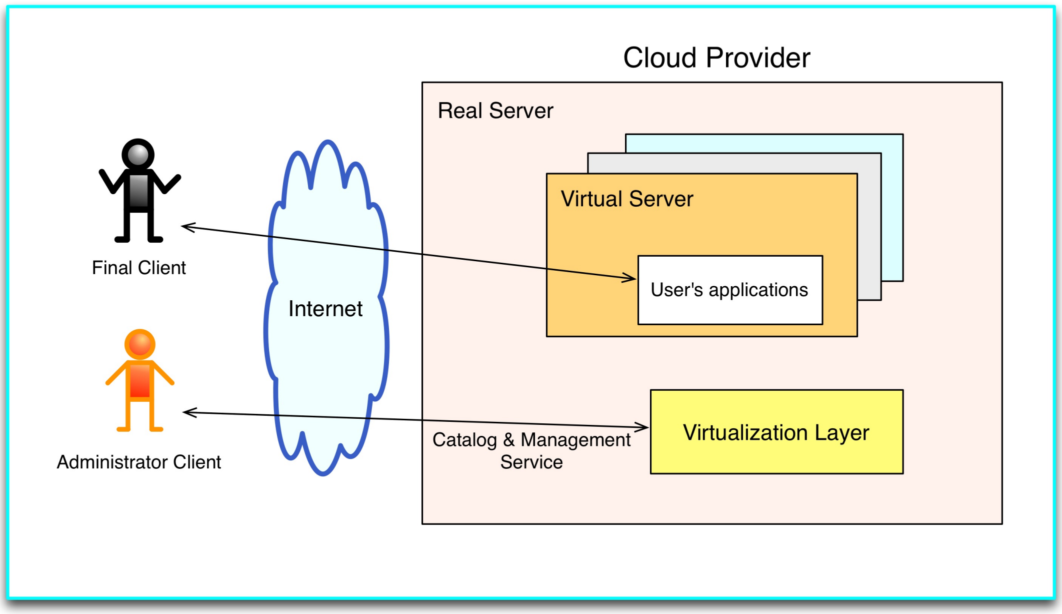


FIGURE 1.16: Cloud Architecture

1.3.1 Openstack

OpenStack[18] is a Cloud-computing project that acts as IaaS platform. It is free and open-source software released under the terms of the Apache License[19]. The project is managed by the OpenStack Foundation. More than 200 companies have already joined the project.
The software has a modular structure composed of a series of interrelated components that control pools of processing, storage, and networking resources throughout a datacenter, able to be managed or provisioned through a web-based dashboard, command-line tools, or a RESTful API.

1.3.2 OpenNebula

OpenNebula[20] is an open source Cloud management toolkit that can be used to build different types of Cloud infrastructures. It is very widespread, has a modular structure and is able to scale from a single node cloud to thousands of physical nodes. OpenNebula manages storage, network, virtualization, monitoring, and

[18]https://www.openstack.org/
[19]http://en.wikipedia.org/wiki/Apache_License
[20]http://opennebula.org/

security technologies to deploy virtual machines on distributed infrastructures, combining both data center resources and remote cloud resources.

1.3.3 WNoDeS

Worker Nodes on Demand Service (WNoDeS) [24] is a Cloud solutions developed by INFN and released as a service in EMII. It uses a batch system, integrated in a local farm, to define the resources allocation policies; the batch systems supported are: IBM Platform LSF, Torque/Maui, SLURM. A study to integrate some components of OpenStack inside WNoDeS is in progress. An important feature, called mixed mode[21], is the possibility to manage jobs that use physical resources and virtual resource at the same time on the same hardware.

[21] http://web.infn.it/wnodes/index.php/mixed-mode-faq

Chapter 2

Architecture

2.1 Portal

Before describing in detail the architecture of the portal developed, we introduce some concepts: a general defenition of web portal and which are its components. In the subsequent paragraphs we describe the type of Grid portal and we explain in detail the architecture of the one developed.
A web portal can be considered as a set of web pages in which the contents are composed of information originated from different sources and displayed in a uniform way. Usually, each source has its dedicated area on the web page for displaying information (the portlet); often, the user can choose which information to display. Modern Web portals offer several services such as e-mail, news, information from databases and even entertainment content. The features available may be restricted in case of authorized and authenticated users (employee or member) or anonymous site visitor.

2.1.1 Portlet

A portlet is a Java technology to develop modular component that are managed and displayed in a web portal. Portlets produce fragments of markup code (HTML, XHTML, WML) that are aggregated into a portal. Typically a portal page is composed of a collection of single portlet and the same portlet can be used in different pages. Through the portlet it is possible to build dynamic contents and

manage the contents settings.
Portlet standards are JSR168 [8] and JSR286 [9] and enable software developers to create portlets that can be reused into any other portal that support the standards.

2.1.1.1 Portlet Container

The portal container is the component responsible to manage the single portlet instance and the distribution of the code fragments generated from the portlet to the portal server where they are aggregated.
The portal container is responsible for the instantiation of the methods to manage the portlets lifecycle and to provide them the correct execution environment

2.1.1.2 Portal Server

A portlet is a single web element. In order to create a web portal page it is necessary a component that aggregates all the fragments code generated by the portlet that need to be shown in the same page, this task is carried out by the Portal Server. The portal server is responsible for the communication with the portal container of all user's request done through the web portal page To sum up the portal container builds the web page contents (a content for portlet), while the portal server aggregates all these contents in a single page providing a uniform style.
There are several solution that offers portal servers/containers, commercial and opensource like: Liferay, GridSphere, iGoogle etc. In the next paragraph we describe in more detail the Liferay solution because it is the one chosen to build the portal.
Fig. 2.1 shows the web portal architecture and how the components described (portal container, portal server and portlet) interact with each other to create a dynamic web page.

2.1.1.3 Liferay

Liferay[1] is a free and open source portal written in Java and distributed under the Lesser General Public License (LGPL). It is worldwide distributed and used

[1] http://www.liferay.com

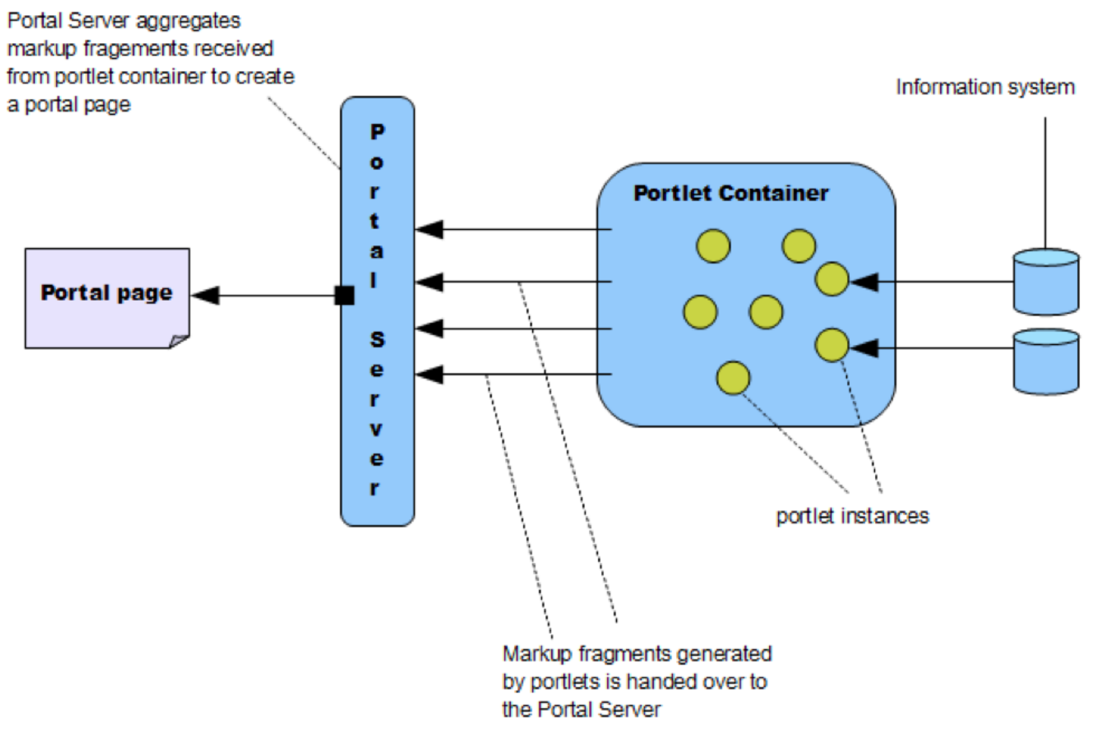

Portal Server aggregates markup fragements received from portlet container to create a portal page
Information system
Portal Server
Portlet Container
Portal page
portlet instances
Markup fragments generated by portlets is handed over to the Portal Server

FIGURE 2.1: Portlet Container

by thousands of organizations and company and it is a leader among the open source portal. Liferay is composed of an engine and applications executed by the engine. it provides also an easy to use development framework for new applications or customization. New developed applications can be integrated with the native Liferay applications. Users access native and new applications as the same set. It provides natively 60 portlets with general functionalities: content manager, wiki, forum, blog and many others. It supports the JSR168 and JSR286 portlet standards and utilities as Service Builder to build automatically interfaces for database. The use of standards makes easier and faster the implementation of new functionalities and also the reuse of the same portlets on other portals that use the same technology.

2.2 Grid Portal Classification

The Virtual Organisations Portal Policy [4] is the document produced by the EGI Security Policy Group in which portals are grouped in different classes according to the executable code, executable parameters and the input data provided by the

Portal Class	**Credentials**	**Executable**	**Parameters**	**Input**
Simple one-click	Robot Certificate	provided by portal	provided by portal	provided by portal
Parameter	Robot Certificate / Personal Cerificate	provided by portal	chosen	chosen from repository
Data processing	Robot Certificate / Personal Cerificate	provided by portal	chosen	provided by user
Job management	Personal Certificate	provided by user	provided by user	provided by user

Table 2.1: VO portal policy summary.

users or portal. There are four portal classes: Simple one-click, Parameter, Data processing, Job Management. Each portal class allows a different set of actions:

- *Simple one-click portals*: the submitted jobs use executable codes, parameters and input data provided by the portal and can not be modified by the users.
- *Parameter portals*: the submitted jobs use executable codes provided by the portal. The user can only choose parameters from a list and select input data from a list of files saved in the Portal repository.
- *Data processing portals*: the submitted jobs use executable codes provided by the portal. The user can only choose parameters from a list and provide personal input data.
- *Job Management portals*: the submitted jobs use executable codes provided by the portal. The user also choose parameters and provide personal input data

All portals operated by (or on behalf of) a VO, which comply with the general EGI Virtual Organisation Operations Policy [5], have to deal with the limitations defined in the VO Portal Policy. Table 2.1 describes the user privileges and the required credentials for each portal class.

In particular the portal developed belongs to the "Job Management" class since all users provide their personal certificate. In this way the users can provide their computational jobs with the executables, input data and parameters. For this reason the portal is as flexible and powerful as a Grid UI. Other solutions, such as, Science Gateways [34], relying on the Robot Certificate, belong to the Data Processing class detailed in Table 2.1. Within these portals users must provide the input data while the computational job executable and parameters are fixed and pre-configured within the portal.

2.3 Portal Architecture

The IGI Web portal is based on the Liferay Framework[2] and it has been designed and developed to allow an easy and fast access to Grid and Cloud resources. The main tasks of this kind of portals are to hide and automate some procedures that for not expert users could be too complex, such as: authentication mechanism, job submission and monitoring, data transfer etc. Moreover it is also possible to improve some Grid functionalities that the normal gLite UI is not able to offer such as: workflow and specific application submission, Cloud integration etc. In the next chapters, each of these functionalities will be described in detail. Being the portal based on Liferay Framework, it grants a modular and flexible structure. The portal modules are the portlets and each one of them implements a specific function. This modular structure makes very simple to extend the portal functionalities by simply adding new portlets. Fig. 2.2 shows the overall portal architecture where dark box are third part components while light box are the components we developed.

The Web portal is the high-level interface that hides the underlying complexity and through which users can interact with the connected services running in the background. In the center of Fig.18 there is the portal composed of several blocks, the portlets, that implement the interaction with external services which are:

- MyProxy and VOMS for storing credentials and requesting VO attributes.
- IdPs for managing user authentication,
- CA-Bridge to request the creation of certificates on demand from an online-CA,

[2] http://www.liferay.com

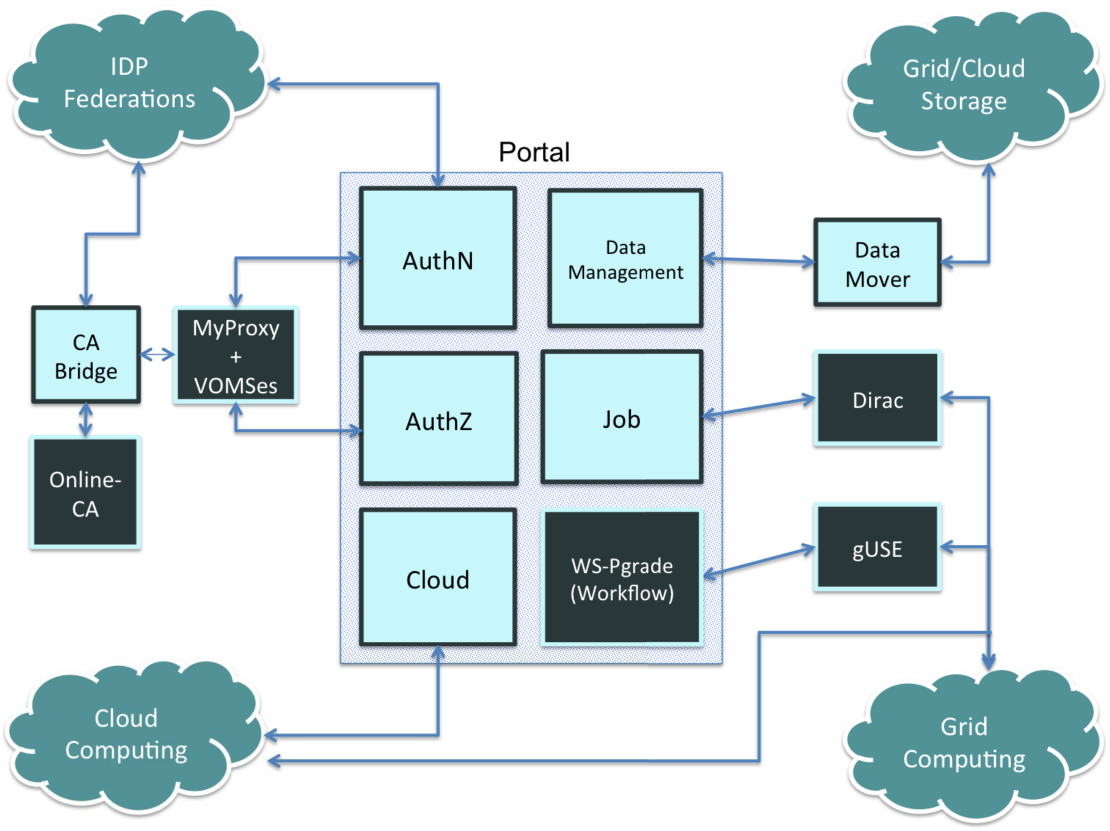


FIGURE 2.2: Portlet architecture

- Grid and Cloud services for requesting computing and data resources.

Some of these services are solution already existing and widely used, the reuse of existing and maintained solutions is key for its sustainability.

2.4 Database

An additional MySQL database, called Portal DB, has been added to the default Liferay DB. The Portal DB is used to collect user's information. Not only does it store user's personal data such as name, surname and mail, but also data concerning the Grid and Cloud like certificate information and VOs membership. These information as described in the next chapter are collected during the registration phase and and are validated before being recorded in the database.
Fig.8 shows the Portal DB relational schema which is composed of 5 tables:

- *userInfo*: this table collects the personal user information, through an external key it is possible to store the information if the user has completed the registration procedure
- *certificate*: this table collects the informazioni about user's certificate. The tables has an external key to link the certificate to the user identifier. The data collected are: subject, issuer and expiration date and not the whole certificate itself for security reasons. In addition to these information, here is also stored the username used to save the user's proxy certificate in the MyProxy server (section 1.1.2.1).
- *VO*: this table collects the information about the VOs supported.
- *userToVO*: this table allows to collect relations among 3 entities: userInfo, VOs and certificate. The information saved in each tuple are: the user's VO and the user's certificate. Moreover, since a user for each VO can be a member of more groups and have more roles, these information are saved here.
- *notify*: this table collects the information about the proxy lifetime in order to notify the users, some hours before the proxy expiration and give them enough time to renew it in case of running jobs.
- *sshKeys*: this table collects the keys pair, in encrypted form, used for the connection to virtual machines managed in the Cloud section.

Fig. 2.3 shows the Portal DB schema.

The database has been designed to be useful during the registration phase in order to help the user with dynamic information in drop-down menu on the base of his choices. In appendix D is shown a detailed portal architecture with all the components described in the next chapters.

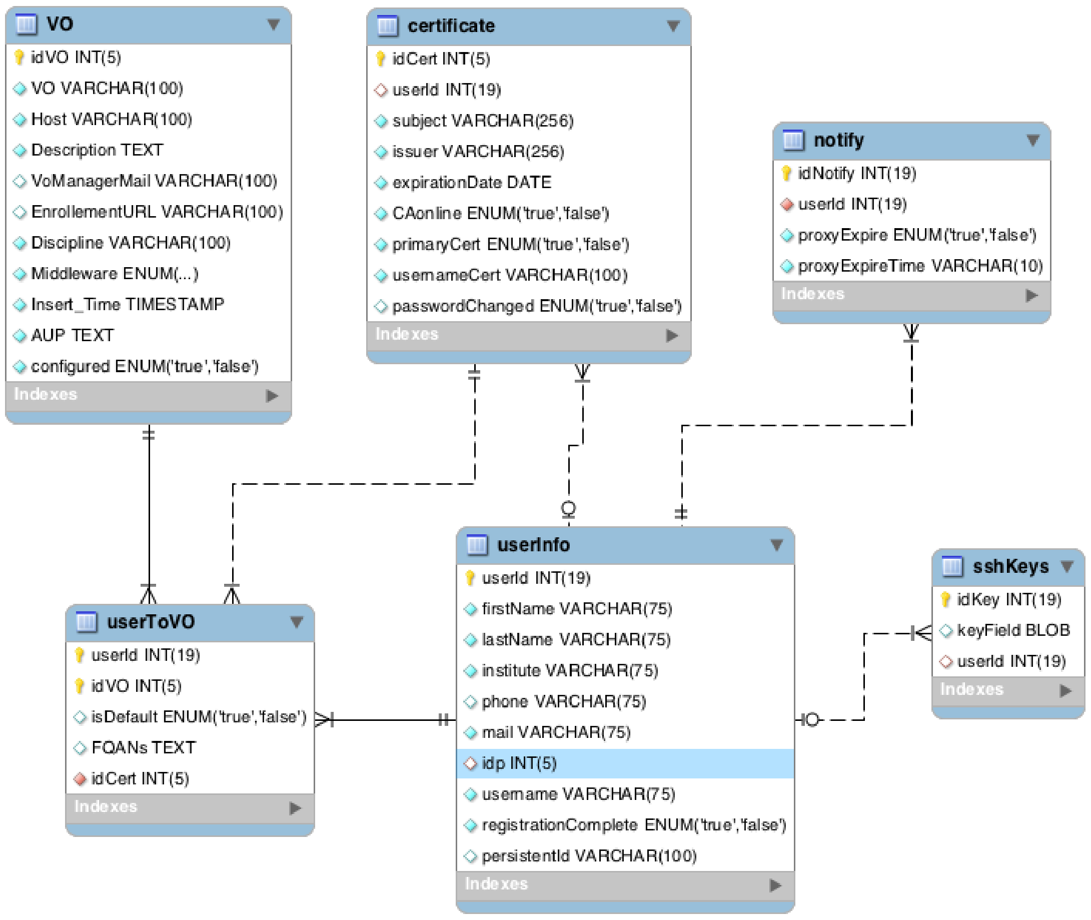


FIGURE 2.3: Database schema

Chapter 3

Authentication and Authorization

Authorization and Authentication processes in the portal are interconnected: the mechanism of the authentication depends on some choice made during the Registration and one step of the Registration uses the authentication mechanism. For this reasons it is not completely correct to describe one before the other. We have chosen to introduce first the Authorization then the Authentication because the first time users access the portal they have to register themselves proving all the necessary credential, the next times they only have to perform the authentication. For both these aspects we have tried to simplify the procedures, moreover we have done a study to integrate an online-CA in order to provide certificate on-demand.

3.1 Related Works

Before the beginning of the work, several existing solutions which provide on-demand certificate creation, have been examined in order to understand if one of these solutions could be adopted. The softwares analized were: GridCertLib[1] [7], CILogon[2] and Terena Certificate Service[3]. Unfortunately, none of these solutions was completely satisfactory for the reasons explained in the next paragraphs.

[1]https://code.google.com/p/Gridcertlib
[2]http://www.cilogon.org
[3]http://www.terena.org/activities/tcs

3.1.1 GridCertLib

GridCertLib is a Library developed in SWITCH [4] and is based on short-lived (11 days) certificates. Every time the users login on to the portal, using their federation identity, it contacts a SLCS online CA in order to provide on-demand new personal certificate. The weakness is that jobs longer than 11 days would fail. To allow users to submit jobs longer than 11 days, the portal has to implement a mechanism to trace any jobs and when they are near to expiration, it has to notify the users to refresh their credentials. This is the main reason why GridCertLib has not been considered as a good solution. This issue could be addressed by adopting long-lived credentials (13 months) in this way the users should be notified for example one month before the expiration of their credentials to renew them through the portal interface and this operation would be only once a year.

3.1.2 CILogon

The CILogon is the solution developed by University of Illinois. It provides different types of long lived certificates (11 months) based on the authentication mechanism used to request the certificate. Users can obtain an *openID* certificate using weak authentication (e.g., their Google account) but these certificates is not accredited by EUGridPMA so they can be used only on local resources. The U.S. students and researchers, who are members of the InCommon federation[5] and are identified with a specific Level of Assurance (LoA)[6], can obtain "silver" certificates which are recognized in EUGridPMA. Here the problem lies in the fact that this service is based on the concept of LoA that is implemented only in the InCommon Federation (U.S.)

3.1.3 Terena Certificate Service

The Terena Certificate Service is the solution developed on the context of the Geant Project. It consists of a Web portal where the users can authenticate themselves using their federated credentials and then obtain X.509 credentials. Although in this way the users can obtain easily a Terena personal Certificate (13

[4]https://www.switch.ch
[5]http://www.incommon.org/federation
[6]http://www.incommon.org/assurance

months of validity) they have yet the problem to manage it because the enrollment certificate process contemplate that the certificates have to be saved locally (i.e on the browser) and finally uploaded to portal. This procedure is quite unfriendly for many users and also browser-dependent. It would be preferable if the online-CA and portal would be directly connected in order to hide completely the users of certificate management complexity.

3.2 Authorization

Now we are going to explain the Authorization mechanism implemented in the portal. The first access to the portal, the so called "Registration", is the most important phase because users have to prove they have all the necessary requirements to access Grid or Cloud services and the Portal has to register this information in a permanent way in order to automatically configure the correct user's environment every time the users log in. Users must provide the following information:

- a X.509 certificate;
- a VO membership;
- the IdP to which they belongs.

The portal trusts: the X.509 certificate issued by the CAs that are member of EU-GridPMA, the VO recognized by the EGI project and all the IdPs belonging to the EDUgain federation plus one additional IdP embedded into the portal (Portal IdP), which is for the exclusive use of the Portal itself. In future it will be possible to trust other federations, at the moment eduGAIN covers the majority of the institutes involved in the EGI project.
For the user with only some of these credentials the portal can provide the missing ones: Table 3.1 summaries credentials combinations necessary to successfully conclude the registration phase (i.e. x=absent, o=present).

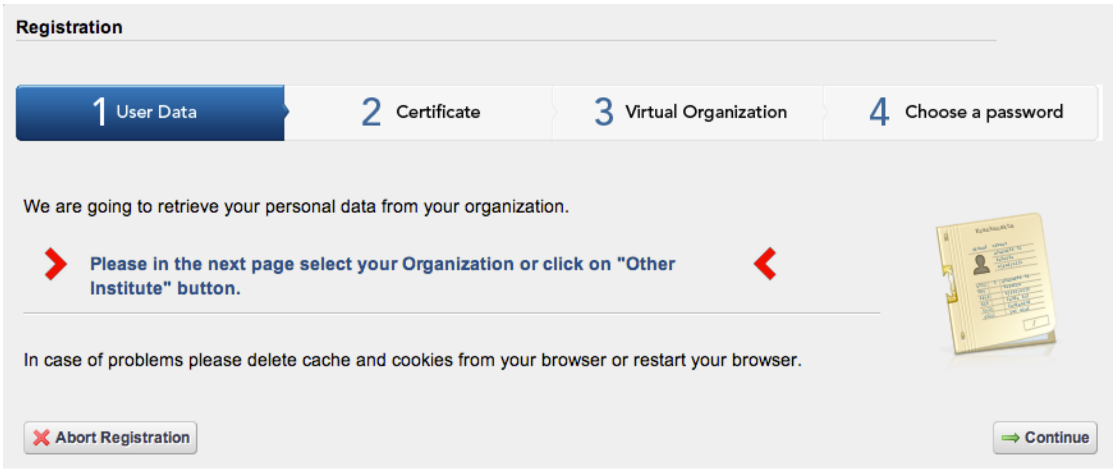
Users who have a valid X.509 personal certificate and a VO membership can use the portal. If they are already registered in a trusted IdP (case 3 in Table 3.1), their personal information are retrieved from the IdP, otherwise (case 1 in Table 3.1) the portal retrieves the user's personal information (such as first and last

Case	**IDP member**	**X.509 certificate**	**VO**
1	x	o	o
2	o	x	x
3	o	o	o

TABLE 3.1: Credentials needed for the registration
Case 2 is possible only as proof of concept

name, email address, institute) from their certificate and registers him/her in the Portal IdP using those information.
The registration phase consists of 4 steps:

- *Step 1: Retrieve personal data from the IdP* By clicking on the "Registration" button, users are automatically redirected to a page where they can select their institute (Fig. 3.1). In case it is in the list, they are automatically redirected to their institute's IdP login page where they have to insert their personal credentials. The portal retrieves the personal information such as first and last name, email address and institute from the IdP. If the institute is not in the list, users have to select "the other institute" option. Then they are automatically redirected to the second step of the registration.



We are going to retrieve your personal data from your organization.

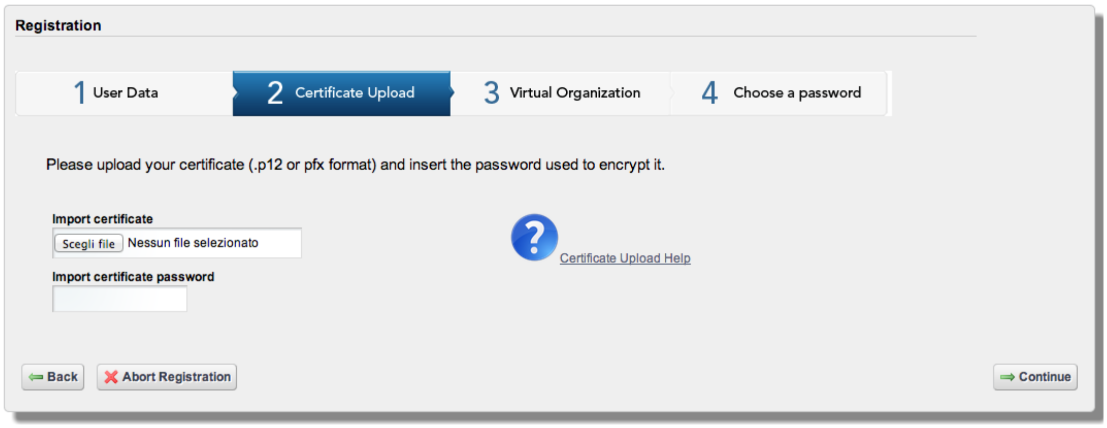
Please in the next page select your Organization or click on "Other Institute" button.

In case of problems please delete cache and cookies from your browser or restart your browser.

FIGURE 3.1: Registration - Step1

- *Step 2: Certificate upload* If users already have a personal certificate they can upload it to the portal through an encrypted channel and type the passphrase necessary to decrypt it. The web interface to do these actions

is shown in Fig. 3.2. The portal at this point starts a set of subsequent actions: saves it in a reserved area of the local file system, does not store the passphrase but uses it to decrypt the certificate, creates a proxy certificates Grid Security Infrastructure (GSI) compliant with the lifetime of the certificate [12], encrypts the proxies with a random password auto-generated, saves the encrypted proxies in a dedicated MyProxy Server. The credentials saved in this server last as the original ones so it is called MyProxy Server (long) to distinguish from another MyProxy used to store shorter credentials called MyProxy Server (short). Once the proxy is saved in MyProxy server the original certificate is deleted. If the whole procedure ends successfully, the Portal registers the user's certificate information (such as DN, expiration time, subject, issuer and proxy username) in the Portal DB. In case the user's organization is not in the list of the trusted IdPs, the information retrieved in this step are used to register them in the portal IdP.
As shown in Table 3.1, users who do not have a valid certificate can not use the portal components. To upload a personal certificate we follow the EUGridPMA guidelines specified in the document "Protection of private key data for end-users in local and remote systems" chapter 2.2 point 3.
We then upload an encrypted personal certificate, in a protected network and we store it only for the few seconds necessary to create the proxy. All the connections are SSL/TSL encrypted. All the web authentications are shibboleth based. The passphrases requested to the user in order to decrypt the user certificate is kept in a non-swappable memory area in clear text and only for the time necessary to complete the procedure and then deleted.

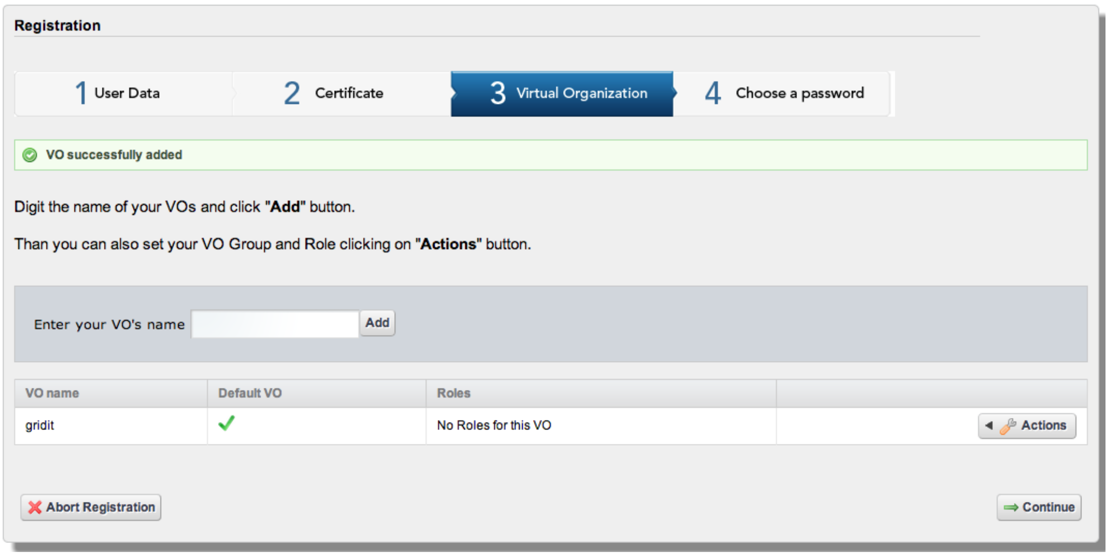


Please upload your certificate (.p12 or pfx format) and insert the password used to encrypt it.

Import certificate
Scegli file Nessun file selezionato
Import certificate password

FIGURE 3.2: Registration - Step2

- *Step 3: Declare the VO membership* Users have to declare their VOs membership. The portal checks the consistency of this declaration querying the VOMS server through the VOMS API. If this operation ends successfully, users can also set their roles and groups for each VO they belong to and set a default VO. In case of more VOs declared, the default VO will be the first VO in the drop-down menu used to choose which VO to use. All these information are stored in the portal DB.
  All these actions are possible through the web interface shown in Fig. 3.3.



VO successfully added

Digit the name of your VOs and click "**Add**" button.

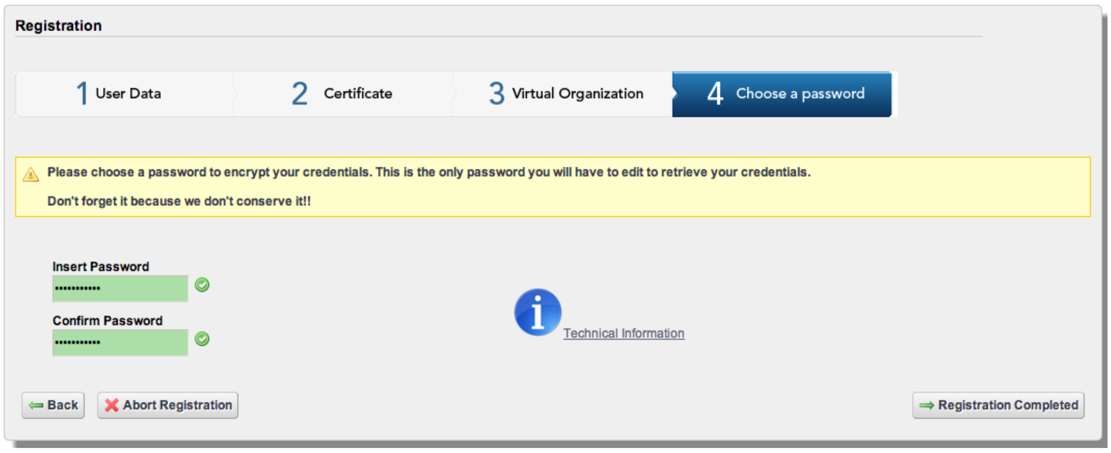
Than you can also set your VO Group and Role clicking on "**Actions**" button.

VO name	Default VO	Roles	
gridit	✓	No Roles for this VO	Actions

FIGURE 3.3: Registration - Step3

- *Step 4: Encrypt credentials* Users must choose a personal passphrase to encrypt their credentials. This passphrase is not saved in the portal DB. The portal replaces the old proxy passphrase, randomly generated during the second step, with this new one. This passphrase is valid until a new certificate is uploaded.
  This passphrases is also kept in a non-swappable memory area in clear text and only for the time necessary to complete the procedure and then deleted. The web interface to do this action is shown in Fig. 3.4.

After successfully completing all the registration steps, users are redirected to a summary registration page (called "MyData") shown in Fig. 3.5 and then can use the portal services.



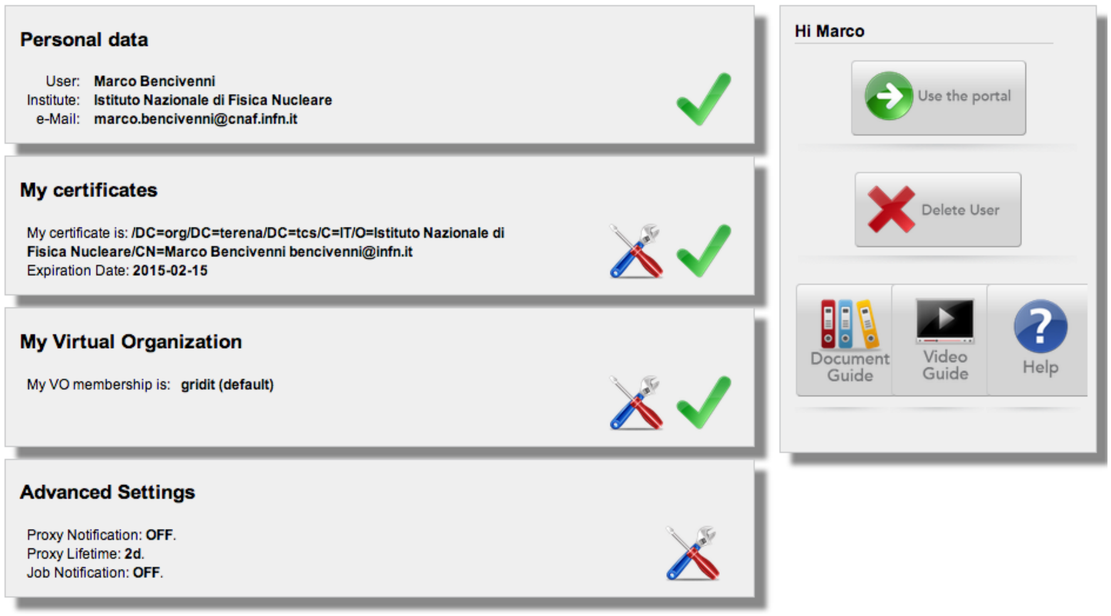
Registration

1 User Data | 2 Certificate | 3 Virtual Organization | 4 Choose a password

Please choose a password to encrypt your credentials. This is the only password you will have to edit to retrieve your credentials.
Don't forget it because we don't conserve it!!

Insert Password: ••••••••••
Confirm Password: ••••••••••

FIGURE 3.4: Registration - Step4



Personal data

User: Marco Bencivenni
Institute: Istituto Nazionale di Fisica Nucleare
e-Mail: marco.bencivenni@cnaf.infn.it

My certificates

My certificate is: /DC=org/DC=terena/DC=tcs/C=IT/O=Istituto Nazionale di Fisica Nucleare/CN=Marco Bencivenni bencivenni@infn.it
Expiration Date: 2015-02-15

My Virtual Organization

My VO membership is: gridit (default)

Advanced Settings

Proxy Notification: OFF.
Proxy Lifetime: 2d.
Job Notification: OFF.

Hi Marco

FIGURE 3.5: Registration - Summary

If the registration procedure is interrupted after the first step, it can be resumed on the following login. The registration procedure can be carried out also setting none VO. In this case users are registered but only when they add a VO Membership in MyData they will be allowed to use all the portal services.
The same portlet used for the registration phase is also used to make some changes during the normal use of the portal, such as adding new VO memberships, adding a new role or group for a VO, or updating the certificate when approaching the expiration date. In this page there is also a section for advanced setting where the users can choose 3 parameters:

- default proxy lifetime, from 12 hours to 7 days (default value is 2 days that is a good compromise between security and usability);
- if they should receive an email notification when the jobs submitted change the status (default value is "yes");
- if they should receive an email notification when the proxy is near expiration only if the user has jobs in running state (default value is "no").

Fig. 3.6 shows the four registration steps in detail.

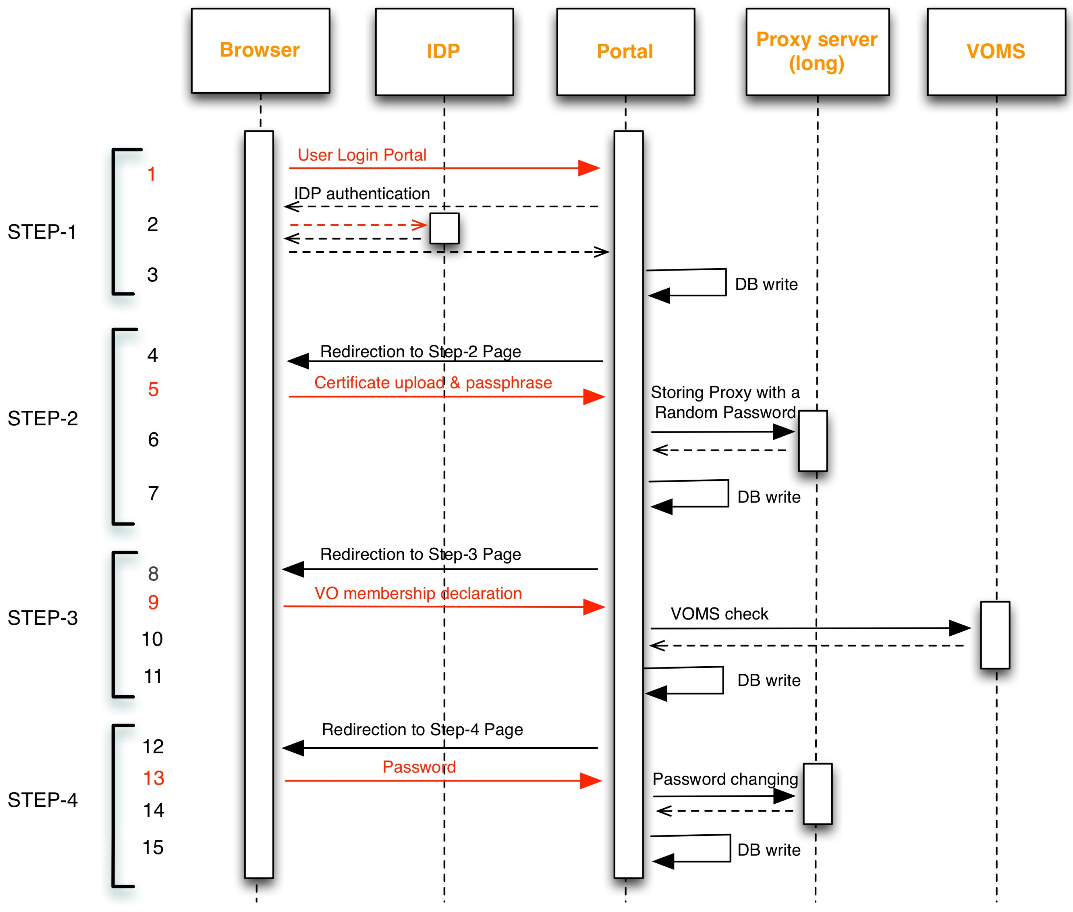


FIGURE 3.6: Registration Flow Diagram
(red arrow indicate a user actions, black arrow indicate a system action)

3.3 Authentication

The portal adopts a federated authentication mechanism based on the SAML protocol. This choice has a double advantage: it allows the users to utilize the credentials stored by their organization's IdP and the portal does not have to store sensible data in its database. Unfortunately Liferay does not support the SAML authentication natively, it supports the Central Authentication Service[7] (CAS) instead. To overcome this issue Casshib[8] has been used. This software enables the CAS server to act as a Shibboleth service provider. We have configured Liferay to use the CAS authentication while, in background, Casshib translates the CAS requests in Shibboleth requests in order to take advantage of the IdP federation. The authentication mechanism in the portal consists of two steps.
Step 1: federated authentication.

1. Users, through the "Sign In" button, are redirected to a web page where they have to choose their institution or "Portal IdP" in case their institute is not in the list (Fig. 3.7 shows this page).

2. Users are then automatically redirected to the IdP login page where they must insert their credentials.

3. The portal checks in its database if the users is already registered.

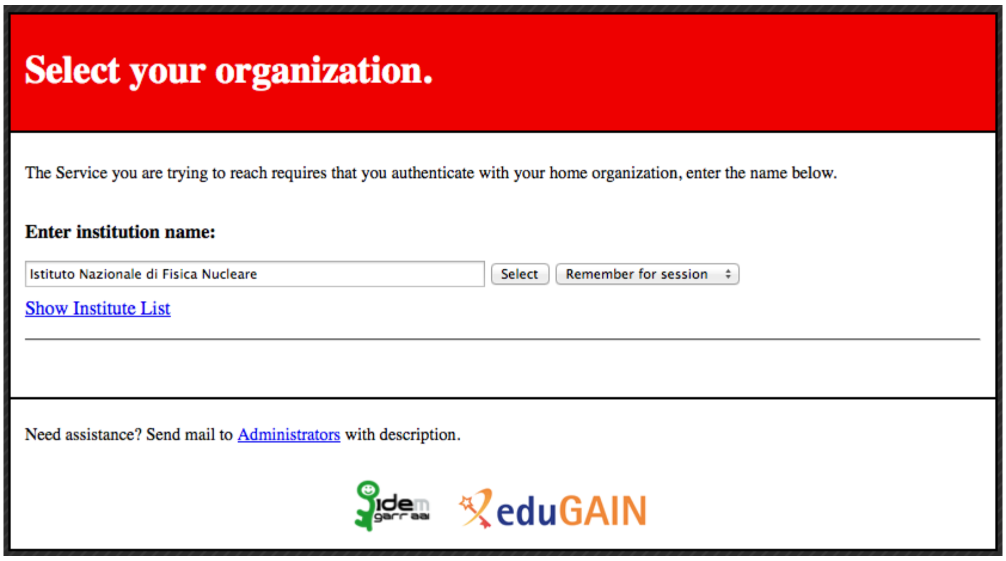
4. If users are already registered on the portal, they are redirected to the main portal page otherwise they are redirected to the registration page.

Step 2: Grid credentials retrieval.

6. Users must select a VO from a menu which lists all the VOs they declared to be part of and insert the passphrase set during the registration phase for the credential encryption as shown in Fig. 3.8.

7. The portal retrieves a proxy by querying the MyProxy Server (long) server. If the proxy is not in the Certificate Revocation List, the portal saves it in an appropriate Portal directory and its lifetime is set to 7 days.

[7]http://www.jasig.org/cas
[8]https://code.google.com/p/casshib



Select your organization.

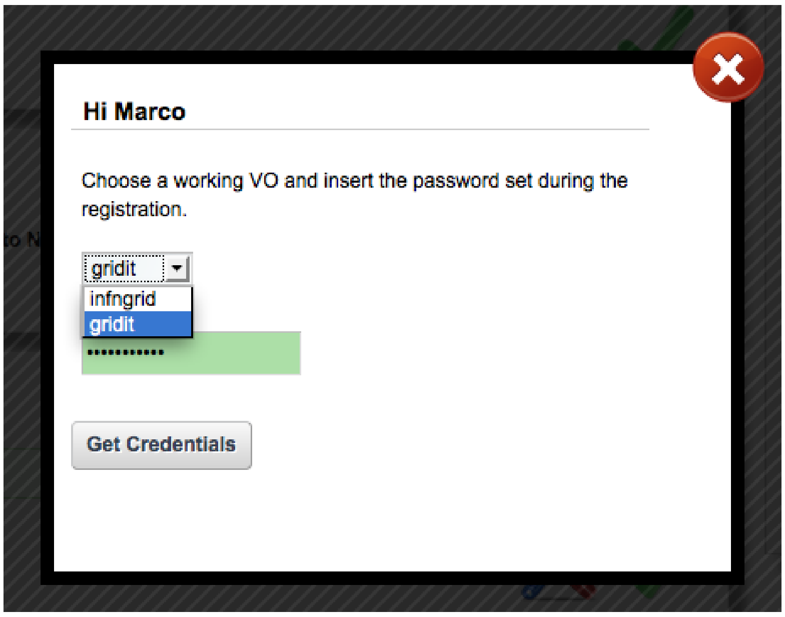
The Service you are trying to reach requires that you authenticate with your home organization, enter the name below.

Enter institution name:

Istituto Nazionale di Fisica Nucleare

Need assistance? Send mail to Administrators with description.

FIGURE 3.7: Authentication - Organization choice



Hi Marco

Choose a working VO and insert the password set during the registration.

FIGURE 3.8: Authentication - VO choice

8. The portal then makes a copy of this proxy to the MyProxy Server (short) that will be used by the WMS to renew the proxy if necessary.

9. According to the selected VO the portal contacts the appropriate VOMS server in order to add VOMS extension to the short-lived proxy. The proxy with VOMS extensions is called voms-proxy and is the only proxy ever transmitted to the Grid. The VOMS extension lifetime is limited by the VO policy. For example, if a user sets N equal to 4, the portal requests a VOMS extension of 4 days. In case the VO policy only allows the extensions of 24 hours, the portal shall renew the VOMS extension every 24 hours.

Fig. 3.9 shows each step of the authentication in detail.

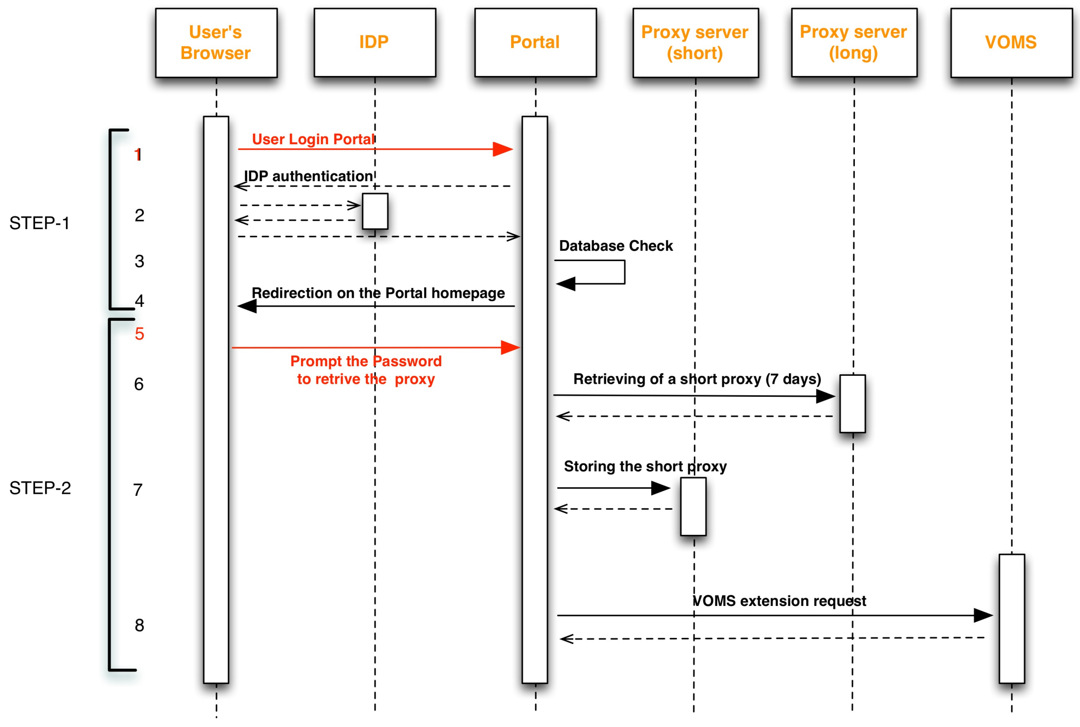


FIGURE 3.9: Authentication Flow Diagram
(red arrow indicate a user actions, black arrow indicate a system action)

3.4 Online CA integration

As highlighted during the introduction, the request of a personal certificate and its management often represents an issue for not ITC expert users and none of the

solutions available up to now satisfies completely their requirements.
As a proof of concept we have designed a solutions to overcome this limitation including in the portal architecture an Online CA which provides X.509 certificates MICS profile on the basis of a federation identity (federation acts as RA) and a service (CA-Bridge) to manage these certificates on behalf of the users.
This solution is useful for users without an X.509 certificate but member of a trusted IdP (case 2 in Table 3.1). Using the infrastructure that we are going to describe, users could get a proxy to operate on the Grid through the IGI portal in a transparent way. During the registration phase at step 2 in the previous section (3.2), the portal could provide the option to ask for a certificate.
Fig. 3.10 shows the role of the Online CA and CA-bridge in the portal architecture. Users can interact directly with the Web Portal for Grid and Cloud tasks as well as with the CA-Bridge for the certificate management services but not with the Online CA because it is protected by a two-levels firewall. The CA-Bridge service is the only one that can interact to the CA to apply for a certificate on behalf of users and then automatically stores the certificate in the MyProxy server. The encrypted long-term proxy is stored into the MyProxy server, while a delegated short-term proxy is available whenever needed for the Grid tasks. The CA-Bridge, the MyProxy server and the Online CA are the fundamental components of a certificate provisioning service integrated in the portal framework. To implement the Online CA it has been used an open source software, EJBCA[9], while the CA-Bridge has been completely developed by us; both these components are Java based.

Certificates are issued according to the following workflow depicted in Fig. 3.13 (each number of this list has a correspondence in the figure):

1. By clicking on the "Registration" button, users are automatically redirected to a page where they can select their institute.

2. They are automatically redirected to their institute's IdP login page where they have to insert their personal credentials. The portal retrieves the personal information such as first and last name, email address and institute from the IdP.

3. The portal checks if the users are already registered or not.

[9]http://www.ejbca.org

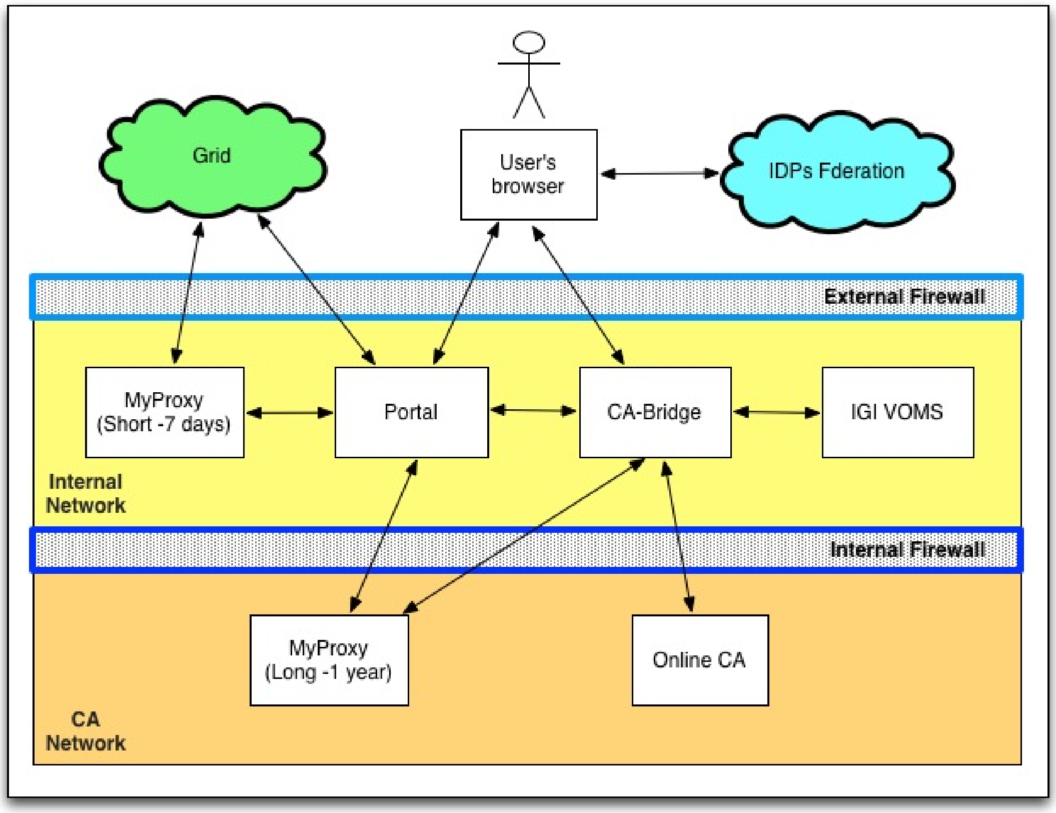


FIGURE 3.10: Portal architecture with the online-CA

4. Users not already registered are redirected to the second step of the registration process.

5. Users ask for a X.509 certificate. Fig. 3.11 shows the step-2 of the registration modified in order to allow the user to choose if they want to upload their certificate (if they have one) or ask for a new one.

6. The Web Portal redirects the users to the CA-Bridge.

7. This web page is provided by the CA-bridge server and not by the Portal server. Even if this aspect is totally transparent for the user, the user's browser provides to CA-bridge service the necessary information to perform a successful authentication against the user's IdP. This is a possible thanks to the SSO. In this way there is a double authentication: the first is needed to access the portal and the second to access the CA-bridge, using the same IdP. The IdP releases the required attributes: Name, Surname, Institute etc.

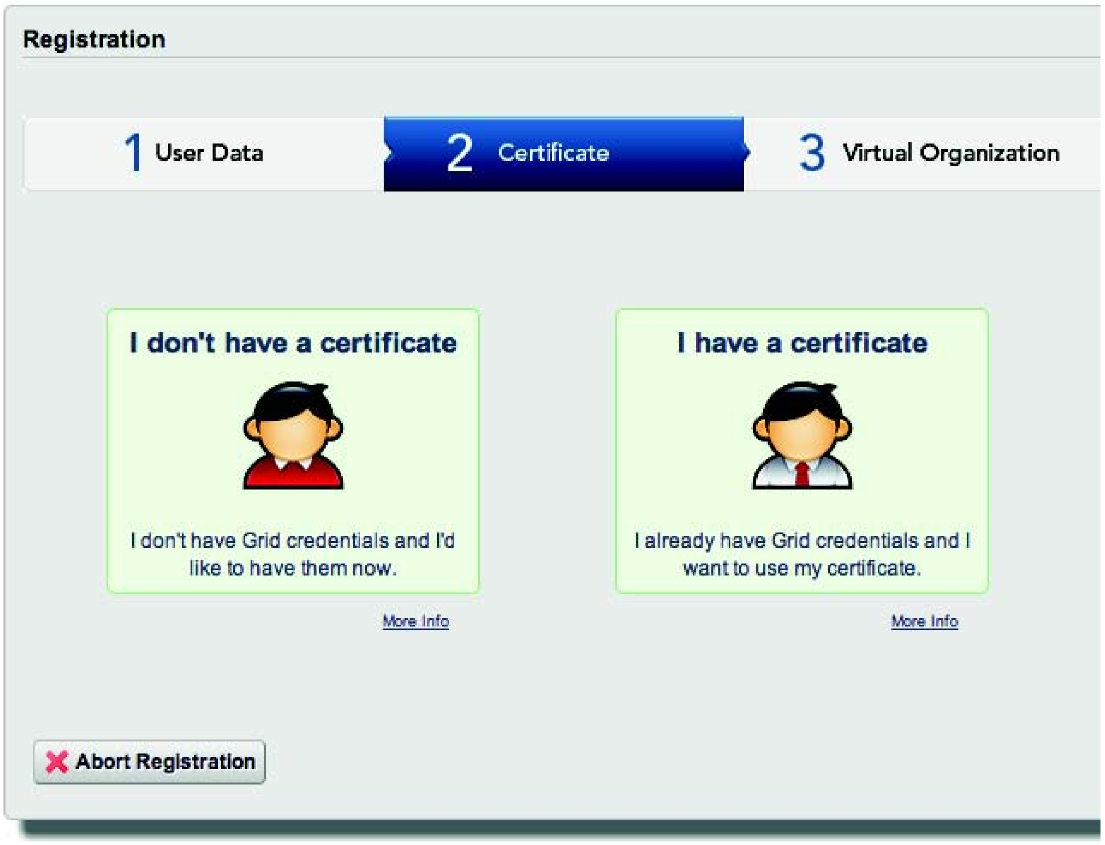

Registration
1 User Data
2 Certificate
3 Virtual Organization
I don't have a certificate
I don't have Grid credentials and I'd like to have them now.
More Info
I have a certificate
I already have Grid credentials and I want to use my certificate.
More Info
Abort Registration

FIGURE 3.11: Registration with the online-CA - step2

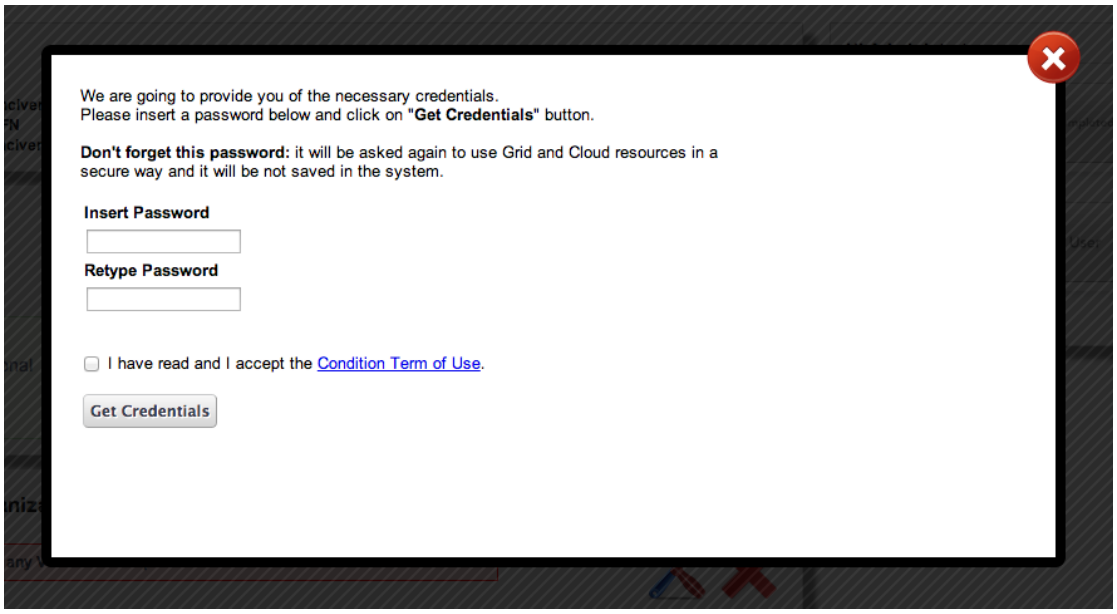
8. The CA-bridge, using a particular mechanism explained in the next pages, checks if the user logged in the portal is the same logged in the CA-bridge. In case of inconsistency the users are automatically logged out.

9. In case of success, the user is asked to provide a passphrase. Fig. 3.12 shows the pop-up window in which the user insert the passphrase.

10. The CA-Bridge generates a private key and a Certificate Signing Request (CSR) on behalf of the user and sends the CSR to the online-CA to be signed

11. When the certificate is received back on the CA-Bridge, it is used to store a long term proxy.

12. The user's certificate is published in the CA repository and the private key is destroyed.



We are going to provide you of the necessary credentials.
Please insert a password below and click on **"Get Credentials"** button.

Don't forget this password: it will be asked again to use Grid and Cloud resources in a secure way and it will be not saved in the system.

Insert Password

Retype Password

☐ I have read and I accept the Condition Term of Use.

FIGURE 3.12: CA-bridge interface

13. The users are automatically added in a catch all VO whose the portal is the administrator. This VO has a limited set of resources, useful as first approach for the new Grid users.

14. The CA Bridge writes in Portal DB the certificate information for the targeted user.

The security mechanism used to protect the communication occurring at points 3 and 4 and to enable the certificate request is based on a string (called token) generated using the Time-based One Time Password-algorithm RFC 6238 (TOTP) [14].
This algorithm generates a token based on two elements: the timestamp, a combination of the attributes retrieved by the IdP, and a secret-key shared between the Web Portal and the CA-Bridge.

- *Timestamp* is used in order to force the user to make a choice, uploading a certificate or requesting a new one, within 2 minutes: the token does not change inside this time interval. In this way the token changes very often also for the same user.

- *IdP attributes* are used in order to verify that the user who has asked for a certificate in the portal page is the same that is finishing the procedure in the CA-bridge page.

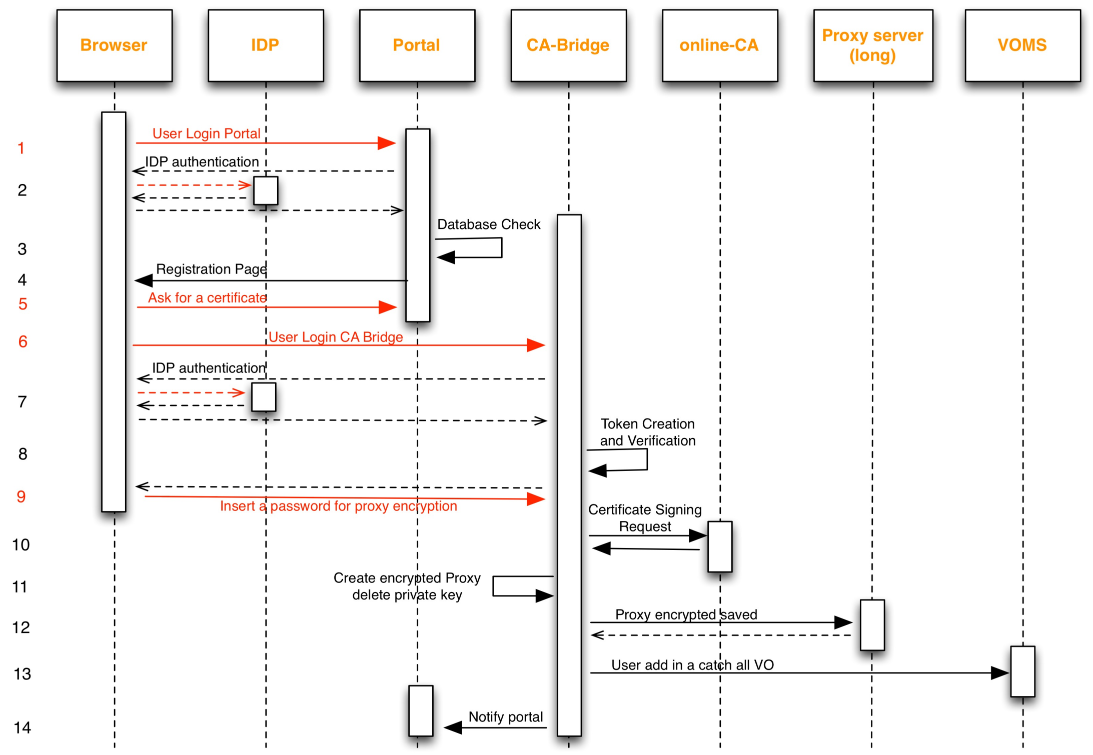


FIGURE 3.13: Online-CA Flow Diagram
(red arrow indicate a user actions, black arrow indicate a system action)

- *SecretKey* is a long string shared between the portal and CA-bridge; this in order to grant that the request of this certificate is coming from the Portal and not directly to CA because in this case the output produced would be different. The secret key is periodically changed.

A first token is generated by the WEB Portal and appended to the redirection URL. A second token is generated by the CA-Bridge.
The CA-Bridge compares the 2 tokens: the one received by the WEB Portal and the one generated by itself. If and only if the two tokens are equal, the point 5 and following are executed. It is worth mentioning that the 2 tokens are equal if and only if the secret-key and IdP's attribute are the same and the time interval between 2 tokens generation is within 120 seconds.
Online-CA and MyProxy Server (long) are in a private network and have a dedicated firewall in front of them. MyProxy refreshes Certificate Revocation Lists every 6 hours. When Portal retrieves a proxy from MyProxy Server (short) and saves it in MyProxy Server (long) checks if the proxy requested is not in any CRLs otherwise the procedure is stopped and the long proxy is deleted. In Grid only

proxies of 12 hours are sent, renewable to a max of N days, with N depending on the value set in MyProxy Server (short). If a proxy is compromised it will be revoked the first time it is refreshed, this means that a compromised proxy has 12 hours as a maximum lifetime. This is the standard behaviour used on all Grid procedures.
Such strong verification can assure both the validation of the user identity on the WEB Portal and the CA-Bridge and that the WEB Portal is the vector of the request. In this way the user is not responsible for the private key, since he will never possess it. Nevertheless he is responsible to protect the passphrase which is used to encrypt the long term proxy. This passphrase is not saved in the system. The users can use the Grid services simply by means of short-lived proxies that are automatically generated and are based on the long-lived proxy certificate stored into the MyProxy Server (long).

Chapter 4

Workload Management

4.1 Existing solutions

Before describing in detail the mechanisms of the workload management implemented in the portal, we introduce two services: DIRAC and WS-Pgrade. These services are widely and successfully used by numerous communities to perform their computation, are developed from third party and actively maintained. For these reasons we have interfaced the portal with them. In the next paragraphs we describe the interaction between the portal and these services and the types of computations possible through the portal: normal job, workflow and specific applications.

4.1.1 gUSE & WS-Pgrade

The "grid and cloud User Support Environment" (gUSE) is an open-source software that enables users to access Grid and Cloud infrastructures. gUSE is developed by the Laboratory of Parallel and Distributed Systems (LPDS) at MTA-SZTAKI, Hungary.
It consists of a set of Web Services which have the functionality to create, interpret and submit a workflow to the Grid in various DCIs. WS-Pgrade is the gUSE front-end: a Web portal based on Liferay implemented using portlets and a database necessary to track and manage the creation and execution of the workflows. These portlets use the gUSE client APIs to submit users' requests to the gUSE back-end

[3]. Although WS-Pgrade and gUSE provide several functionalities (data management, proxy management, etc.) in the portal developed they have been used only for the workflows[18] creation and management. The API, called Application Specific Module (ASM) [19] provides most of the gUSE programmatic functionalities. In order to facilitate the users in workflow construction, WS-Pgrade provides also a Java applet to graphically build the workflow graph which then will be configured using the WS-Pgrade portlet interface.
The last component of this architecture is the DCI Bridge, a web application that creates a transparent layer between the workflow systems and the DCI systems. In this way it is possible to provide a standard access to the DCIs like Grids, desktop Grids, clusters, Clouds etc. All these components are shown in Fig. 4.1.

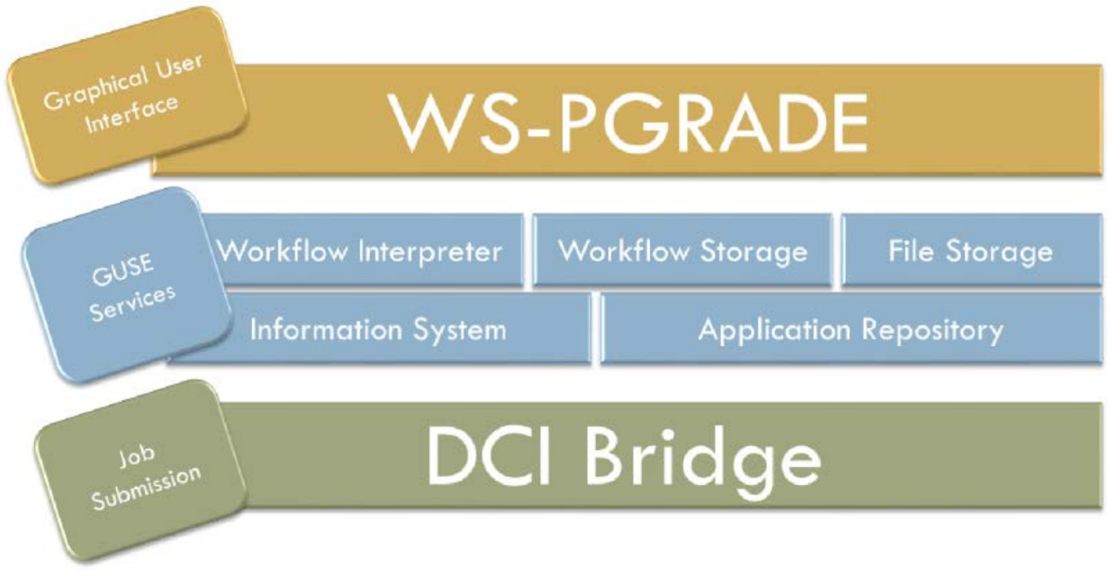


FIGURE 4.1: gUSE/WS-Pgrade architecture

4.1.2 DIRAC

The Distributed Infrastructure with Remote Agent Control (DIRAC) [15] provides a Grid middleware stack that integrates heterogeneous computing resources and provides a solution for both job submission and data management tasks. DIRAC now is a solid, widespread and sustainable solution adopted by the LHCb experiments and by many national Grid infrastructures.
The basic DIRAC components are Databases, Services and Agents that combined together form Systems:

- *Databases*: keep the persistent state of a System. They are accessed by Services and Agents as a kind of shared memory.
- *Services*: are passive components listening to incoming client requests and reacting accordingly by serving requested information from the Database or inserting requests on the Database.
- *Agents*: are the active components which are running continuously invoking periodically their execution methods. They execute actions and send requests to the DIRAC services.
- *System*: is delivering a complex functionality to the rest of DIRAC, providing a solution for a given class of tasks.

DIRAC WMS [16] is the component that provides the scheduling mechanism for jobs. It organizes pending jobs in task queues and each task queue has jobs with similar requirements as shown if Fig.4.2. The fundamental concept in DIRAC is the Pilot job [17], a very useful solution especially for massive job submissions. For each jobs submitted by the users DIRAC may submit four pilot jobs to the resources that most match the job requirements. DIRAC, before to submit new pilot jobs, checks if there are already pilot jobs with the same requirements. The pilot jobs check the environment in which they are running. It is a task of the pilot job to communicate with the task queue, through a dedicated component called Matcher Service, in order to pull down the first job of the queue whose requirements fit the environment where pilot job is running. In this way there is a decoupling between resource allocation and job management. Moreover there is also a notably reduction of the time that each job has to spend in the batch system queue electing this model very useful for massive short job submission. It has been shown that this model increases significantly the number of jobs submitted successfully. Using the centralized task queue is also possible to define a sort of prioritization (for example for VO policies as shown in Fig. 4.2) changing the job position in the queue [16].

In order to allow the communication between the portal and the DIRAC server, a portlet using the DIRAC Command Line Interface has been developed. Although DIRAC was originally developed to support only one VO (LHCb), during the years it evolved towards a more generic and configurable structure. Our DIRAC installation has been configured to support multi-VOs, as shown in Appendix C.

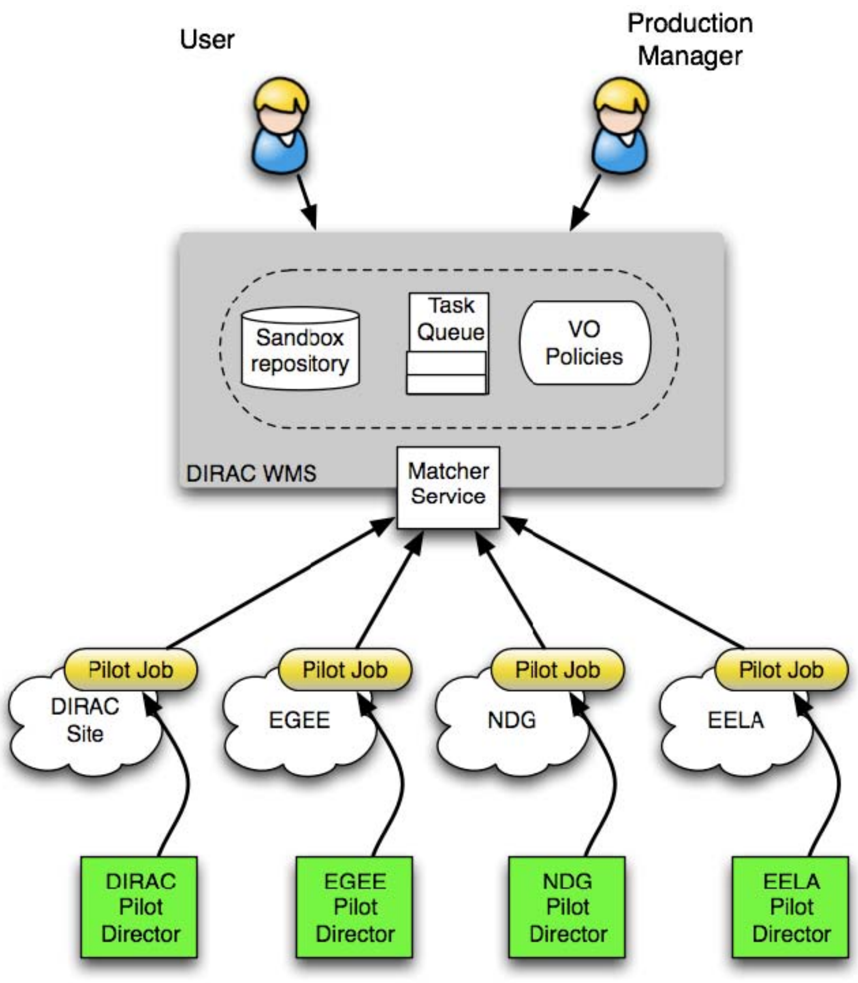
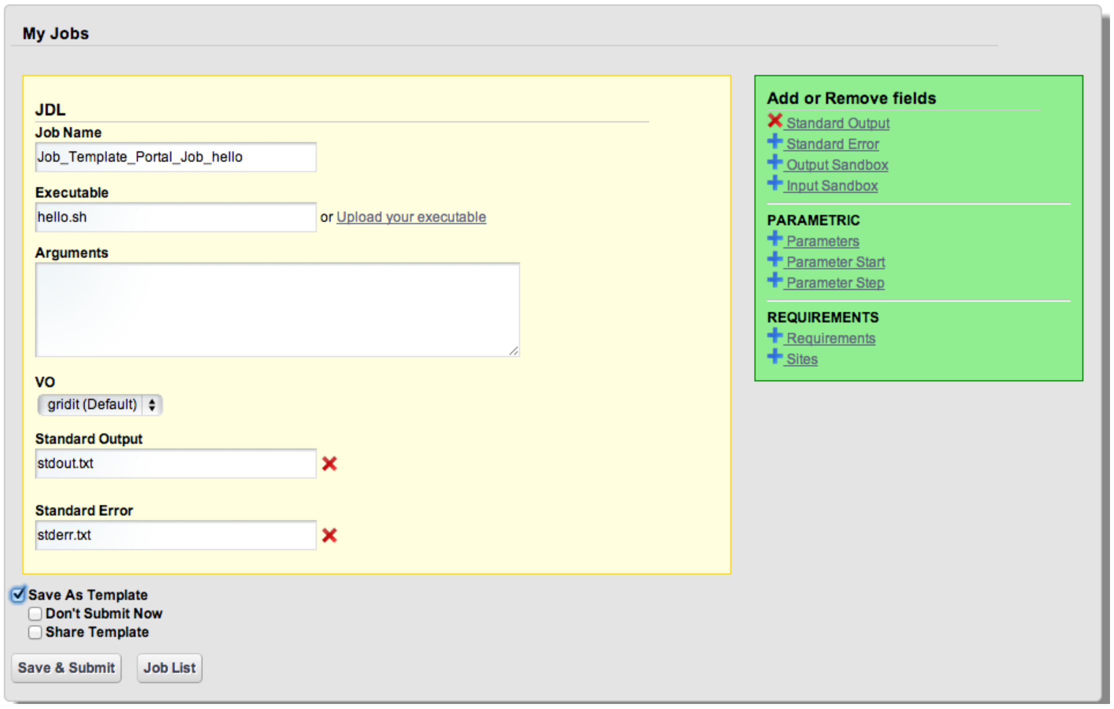


FIGURE 4.2: DIRAC architecture

4.2 Generic jobs

The first type of computation that we describe is the most simple: users want to run jobs by uploading their executables, input files and specifying the executable parameters. For these job types the portal developed provides a very simple interface, through a specific portlet, in which the user can easily build his JDL by setting the appropriate values from a list of JDL attributes. In background the portal uses the DIRAC services inheriting its functionalities but also some of its limitations like the limited choice on the type of job which can be submitted: normal and parametric jobs.
Fig. 4.3 shows this interface, on the left column there are the available fields to build the JDL, in the center there is the JDL that will be submitted.



My Jobs

JDL

Job Name: Job_Template_Portal_Job_hello

Executable: hello.sh or Upload your executable

Arguments:

VO: gridit (Default)

Standard Output: stdout.txt

Standard Error: stderr.txt

Add or Remove fields
- Standard Output
- Standard Error
- Output Sandbox
- Input Sandbox

PARAMETRIC
- Parameters
- Parameter Start
- Parameter Step

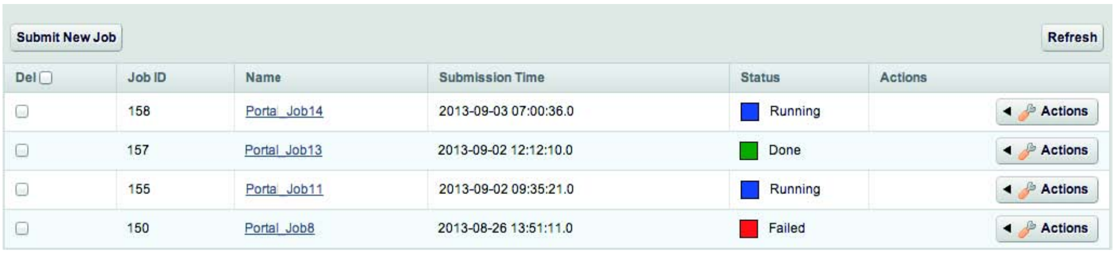
REQUIREMENTS
- Requirements
- Sites

[✓] Save As Template
[ ] Don't Submit Now
[ ] Share Template

Save & Submit | Job List

FIGURE 4.3: The portlet to submit job using the DIRAC service

Once the job has been submitted, the user can monitor the job status during its execution and retrieve the output and log files at its conclusion. It is also possible to re-submit or duplicate a terminated job.
It has also been implemented the possibility to save a JDL as a template in order to reuse all the settings previously made and increase productivity, but the real added value is the possibilities to share the template with other users.
Fig. 4.4 shows this interface, each row represents a job submitted and contains a job univoque identifier (Job ID), a job name, the submission time, the status (blue=running, green=finished successfully, red=finished with error, orange=waiting) and the bottom to perform some actions.



Submit New Job | Refresh

Del	Job ID	Name	Submission Time	Status	Actions
	158	Portal_Job14	2013-09-03 07:00:36.0	🟦 Running	Actions
	157	Portal_Job13	2013-09-02 12:12:10.0	🟩 Done	Actions
	155	Portal_Job11	2013-09-02 09:35:21.0	🟦 Running	Actions
	150	Portal_Job8	2013-08-26 13:51:11.0	🟥 Failed	Actions

FIGURE 4.4: The portlet to monitor the job submitted using the DIRAC service

Once the job is terminated from the same interface is possible to resubmit, delete one or more jobs at a time, and also retrieve the "Standard Output" and "Standard Error" for each job.
The biology use case is a typical example where DIRAC is a good solution because from the computational point of view biology jobs can be considered as a massive computation. Biology jobs can demand for thousands hours of computations and produce TeraByte of data but they can be decomposed in hundreds of independent smaller jobs executed at the same time according to the resources available. In this way the effective computational time necessary to perform the analysis can be notably reduced.

4.3 Workflow

The second type of computation executable through the portal are the Workflows: acyclic sequences of connected nodes in which the execution of one or more nodes depend on the results of the previous ones. Conceptually they are very similar to DAG jobs: both can not be cyclic. The difference is that in workflow a single node can be something different from a job, such as database interaction, VM instantiation etc.
Fig. 4.5 shows an example of workflow structure created through the portal, there are 10 nodes and each of them take as input the output of previous node.

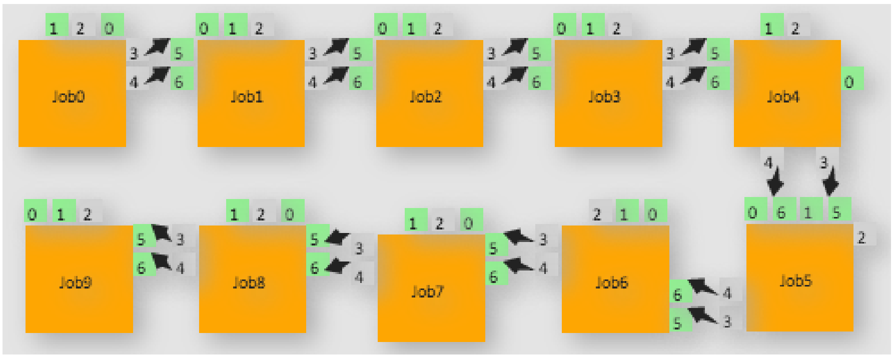
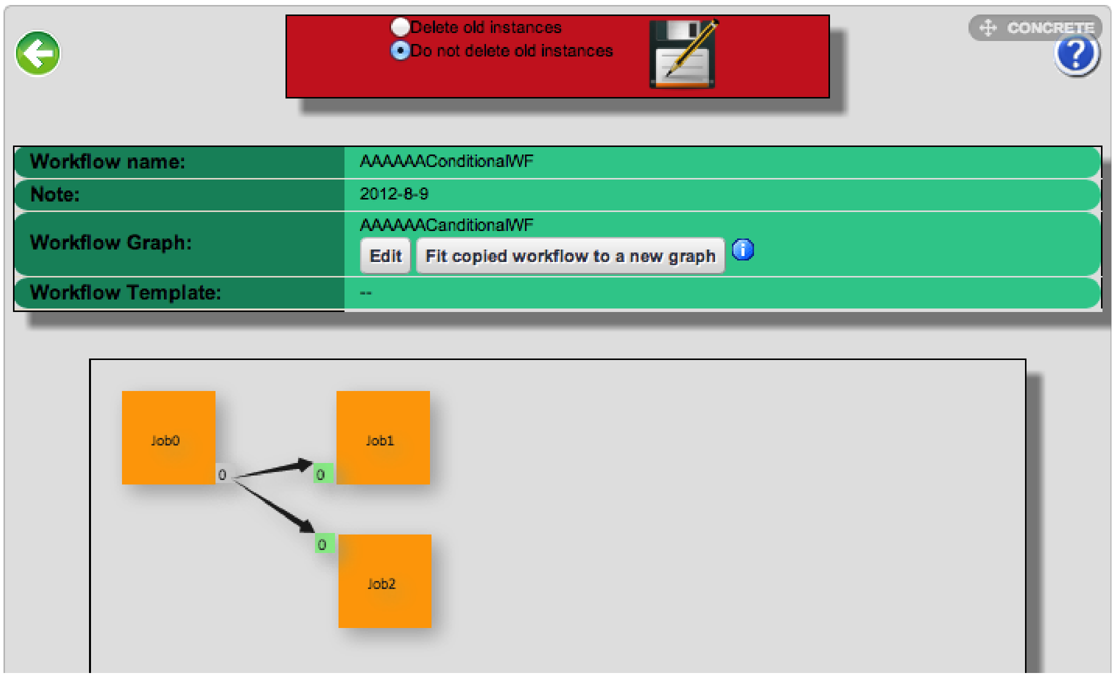


Figure 4.5: Example of Workflow structure

The main advantage of the workflow is the possibility to split the global calculation into smaller and simpler problems: each node of the workflow represents a specific portion of the complex computations. The workflow engine manages every node of the workflows to allow the computation progress independently from the user interaction. From a Grid point of view, each node is independent from the others and are assigned to different resources, optimizing the computational task. It is also possible to define conditional branches which react in different ways according to the evolution of the computation. As for the simple jobs, also for the workflows the user has a total freedom in uploading their executables, input files and specifying the executable parameters for each node. The template mechanism is implemented too. Fig. 4.6 shows this interface which can be used to create the workflow and configure each node.


Delete old instances
Do not delete old instances
CONCRETE
Edit
Fit copied workflow to a new graph
Job0
Job1
Job2
0
0
0

Workflow name:	AAAAAAConditionalWF
Note:	2012-8-9
Workflow Graph:	AAAAAACanditionalWF
Workflow Template:	--

FIGURE 4.6: The portlet to monitor the job submitted using the gUSE/WS-Pgrade service

4.4 Specific Applications

Some communities use ad-hoc applications and consequently may have specific requirements. Typically these communities need to perform complex calculations requiring computing power not provided by their laboratories. At the same time, they do not have ICT experts. An high level interface helps these users to exploit the Grid resources. At the time of writing we have ported several applications. The table 4.1 lists the applications, with a little description, already ported and usable through the portal interface, but also the application which we are working

Applications	**Description**	**Porting status**
Ansys	Engineering simulation software	Completed
Fluka	Physics MonteCarlo simulation package	Completed
Crystal	Alignment search tool for genome sequences	Completed
Quantum Espresso	Electronic structure calculations and materials modelling	Completed
Nemo	Oceanographic modelling	Completed
Blast	Alignment search tool for genome sequences	Completed
Venus	Chemical Dynamics simulations	Completed
Namd	Molecular Dynamics simulations	In development
Geant4	Simulation of the passage of particles through matter	In development
Gaussian	Electronic structure modelling	To do
DMRG	Electronic structure modelling	To do

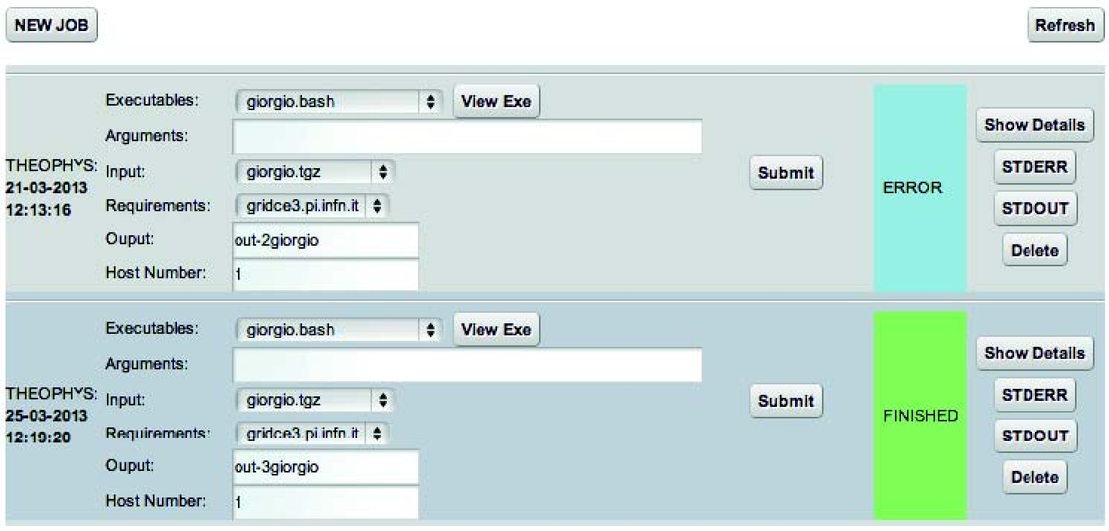
Table 4.1: Applications in the portal and their status

on and those that we will examine in the future.

To do that we followed a well tested procedure: firstly we perform an analysis of the portability of the application in Grid trying to run it on a Worker Node; then we identify the minimum hardware and software requirements (RAM, number of CPU cores, specific libraries, licences, etc.), as last step we investigate, with the application's community experts, what is needed by the users: the input data and output files, application's parameters, required for retrieving log and output files at runtime and any other relevant information.
Once all the previous tasks are completed, we write a script that takes care of the following operations:

- retrieve the user input, parameters and files;
- execute the application;
- collect the output produced during and at the end of the computation.

At this point the porting process can be considered finished and on the base of user's requirements we design a custom Web interface through which users can execute their applications, monitor execution and retrieve partial and final outputs. Technically the implementation of this interface consists in writing an appropriate portlet that performs all the requested actions by means of the ASM API provided by WS-Pgrade which takes a workflow as input. Therefore the last step is the creation of a custom workflow (see section 4.3), based on the user and application requirements. Each node of the workflow executes the script produced in the porting process. In Fig. 4.7 there is an example of a custom interface for an application of Theoretical Physics.



THEOPHYS: 21-03-2013 12:13:16
Executables: giorgio.bash
Arguments:
Input: giorgio.tgz
Requirements: gridce3.pi.infn.it
Ouput: out-2giorgio
Host Number: 1
ERROR

THEOPHYS: 25-03-2013 12:19:20
Executables: giorgio.bash
Arguments:
Input: giorgio.tgz
Requirements: gridce3.pi.infn.it
Ouput: out-3giorgio
Host Number: 1
FINISHED

FIGURE 4.7: Example of portlet for specific application

The fact of having build a simplified interface and of knowing exactly the application behaviour, the submission of an application, compared with the submission of either generic jobs or workflows, has several advantages:

- The job submission only requires filling a Web form.
- It is possibke to interrupt the application execution and restart it in another node. This a fundamental feature in case of long running Jobs because the maximum wallclock time of a site could be not sufficient for the application to finish its computation. Except the first one, every node of the workflow retrieves the data produced from the previous step from a predefined Storage Element so that execution can continue. This procedure goes on until the execution is completed. A file indicating that the computation has ended is

then created on the same SE. This feature is possible only if the application supports this type of interruption and restart.

- In case of failure it is possible to distinguish the error type: a Grid error or an application error. On the base of the error type the portlet can trigger different actions: standard Grid recovery mechanisms or specific application mechanism recovery.

The biggest advantage is probably for long running applications in which this allow for the inspection of logs and output files at run time. The Grid middleware provides the WMS functionality called perusal[1], that has to be specified in JDL files. This functionality saves the job output at regular intervals or before it is removed when it reaches the walltime limit. In the JDL the user must specify the file(s) to be sent to WMS every X seconds. The limit of this feature is given by the number and size of the file(s) to retrieve in order to avoid overload of the WMS. To overcome these limitations it has been developed a new mechanism inside the portal called application progress monitoring. Specific SE are configured in order to be used as a temporary storage for the intermediate outputs. The script used by the portlets for a specific application (as described in the previous section) copies and retrieves the files to/from these SE. To achieve the desired behaviour it has been used the SRM technology. SRM allows to copy selected files from a WN where the job is running to a SE, where the files are stored. The files are copied, every N minutes, in specific SE storage area preconfigured to be available for inspection by the portal. The value of N is a degree of freedom, it is settable but it depends on the file size. Typically N is 30 minutes in order to avoid rewriting a file before the previous writing is finished. In this way the users from the portlet interface can access the files via web. To implement these features, a set of bash scripts has been developed. The bash functions are executed directly on the WN and can interact directly with the selected SE without involving the WMS and avoiding the potential limitations aforementioned. In this way users can check the output at runtime and evaluate possible strategies aimed at saving time, terminating in advance the jobs that produce unuseful output, and computing resources as well as at avoiding the waste of license usage.

[1]http://wiki.egee-see.org/index.php/Job_output_monitoring_using_job_perusal

Chapter 5

Data Management

5.1 EMI Data Management utilities

In this chapter we describe in detail the mechanisms of data management implemented in the portal, explaining the different phases of data management: upload, download and other actions on the data in Grid.
As described in section 1.2.4, thanks to the SRM interface, Grid users can carry out the common operations (such as upload, copy, download, etc) transparently from the specific implementations of the involved SEs. These operations are possible by means of some sets of command line utilities distributed with the UI: the *LCG utils* and the *lcf-* commands*. Both of these utilities interacts with the LCG File Catalog LFC and are used by the portal to execute operations on files and directories.

5.1.1 LCG utils

LCG utils is a suite of client tools for data movement based on the Grid File Access Library (GFAL), which is also included. The tools allow users to copy files between SEs and to register entries in the LFC and replicate files between SEs.. Table 5.1 lists the complete list of commands.

Command	**Description**
lcg-cp	Copies a Grid file to a local destination
lcg-cr	Copies a file to a SE and register the file in the catalog
lcg-del	Delete one file
lcg-rep	Replication between SEs and registration of the replica
lcg-gt	Gets the TURL for a given SURL and transfer protocol

Table 5.1: LCG Utils

Comman	**Description**
lfc-chmod	Change access mode of the LFC file/directory
lfc-chown	Change owner and group of the LFC file/directory
lfc-delcomment	Delete the comment associated with the file/directory
lfc-getacl	Get file/directory access control list
lfc-ln	Make a symbolic link to a file/directory
lfc-ls	List file/directory entries in a directory
lfc-mkdir	Create a directory
lfc-rename	Rename a file/directory
lfc-rm	Remove a file/directory
lfc-setacl	Set file/directory access control lists
lfc-setcomment	Add/replace a comment

Table 5.2: lfc commands

5.1.2 lfc-* commands

The lfc-* commands only operate on the LFC and do not manipulate data. lfc-* commands equivalents exist for many UNIX file commands and as far as possible use the same name prefixed by ”lfc-”. Table 5.2 shows in detail the available commands.

5.2 File manager

As described in the introduction, to perform data management tasks in a standard Grid environment, users can use a set of command line utilities (the LCG utils and the lfc-* commands). These utilities require the knowledge of a command syntax that is not so easy and intuitive to learn. We have worked in order to allow these operations by means of typically web actions as dragging and dropping files in the Web page.
In order to simplify the Grid data management, it has been designed and implemented a sophisticated architecture (shown in Fig. 5.1 that includes several elements intercommunicating in a secure way. The purpose of this solution is to simplify data movement in Grid and at the same time to avoid the portal from becoming the bottleneck.

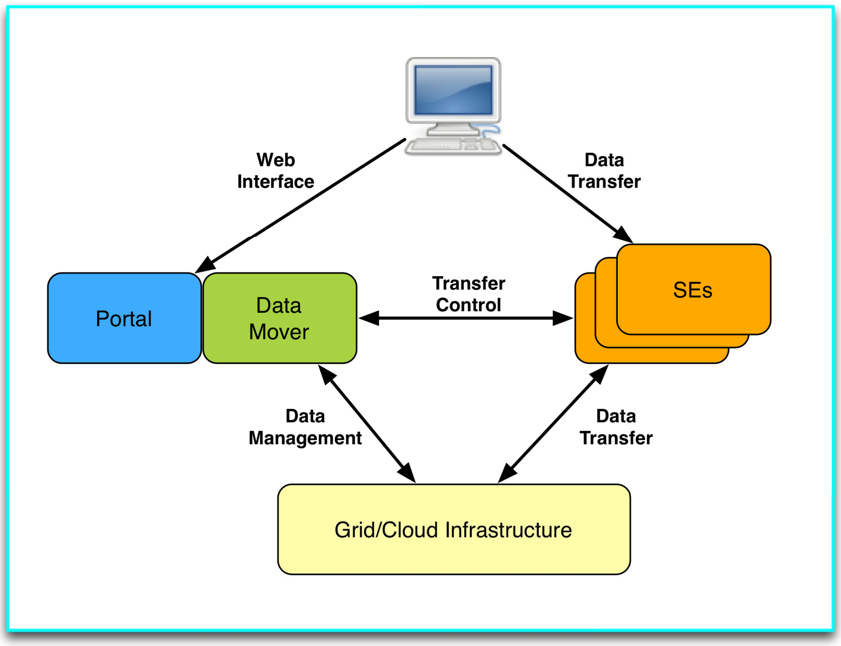
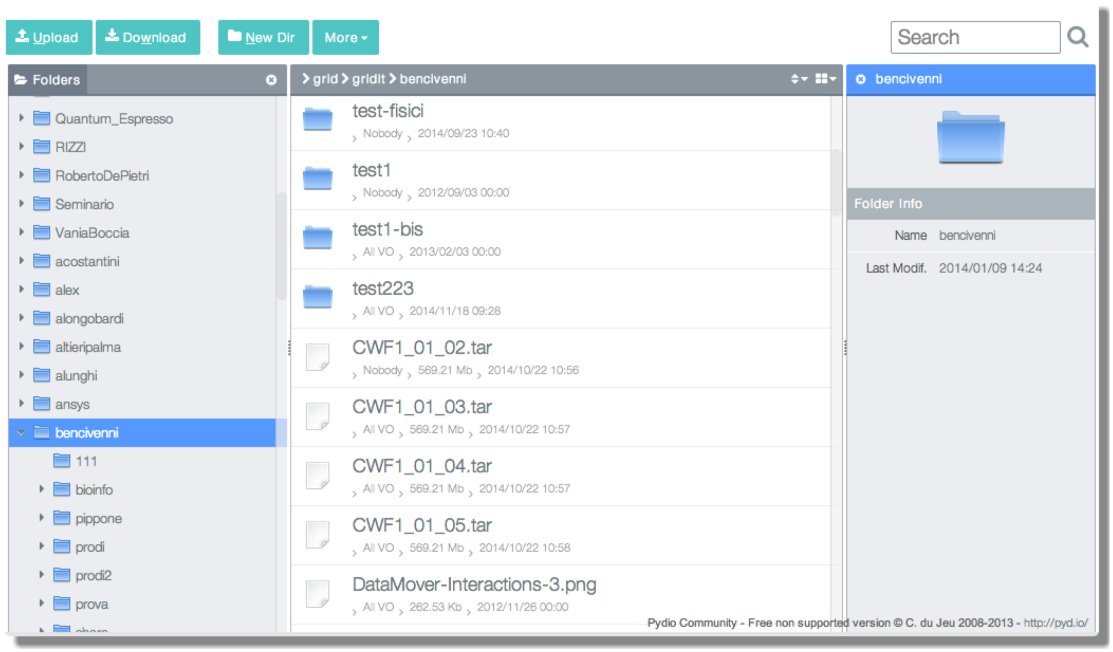


FIGURE 5.1: Data Management Architecture

In the portal architecture, the Data Mover is the external component which allows users to easily transfer and manage data among Grid resources. It controls and manages every step of the file transfer actions (upload and download) [22].
Data are transferred to and from the Grid SEs through an external storage service composed by a set of SEs (Portal SEs), based on StoRM, which keeps the files

until they are transferred to Grid SE (upload phase) or downloaded (download phase). In this way it is possible to achieve a complete decoupling of the interface view (Portal), the control service (Data Mover) and the physical storage (SEs Portal). This architecture also allows a scalable solution because the SEs which act as cache can grow, with geographically distributed SEs, and ensures the final user a correct and fast data transfer without bottlenecks.
The Data Mover component is a Web-based data management service that offers the possibility to benefit from the capabilities of the Grid data management command line utilities by means of simple web operations. This service is based on the Pydio[1] framework, a PHP-based file management system which has been chosen for the intuitive interface and the easy integration of new extensions. Each action is implemented by a specific plug-in that can be easily developed and integrated in Pydio alongside the ones already available. The Portal Data Management plug-in has been developed to extend the basic functionalities of Pydio for local file systems and integrate the LCG utils and the lfc-* commands. In this way, through the web interface (shown in Fig. 5.2), the user can easily browse the content of the file catalogues (those pertaining to his/her VO) and perform the data management operations both on the logical data (which affect the catalogue) and on the physical files (which affect the SE).



Quantum_Espresso
RIZZI
RobertoDePietri
Seminario
VaniaBoccia
acostantini
alex
alongobardi
altieripalma
alunghi
ansys
bencivenni
111
bioinfo
pippone
prodi
prodi2
prova

test-fisici
Nobody , 2014/09/23 10:40
test1
Nobody , 2012/09/03 00:00
test1-bis
All VO , 2013/02/03 00:00
test223
All VO , 2014/11/18 09:28
CWF1_01_02.tar
Nobody , 569.21 Mb , 2014/10/22 10:56
CWF1_01_03.tar
All VO , 569.21 Mb , 2014/10/22 10:57
CWF1_01_04.tar
All VO , 569.21 Mb , 2014/10/22 10:57
CWF1_01_05.tar
All VO , 569.21 Mb , 2014/10/22 10:58
DataMover-Interactions-3.png
All VO , 262.53 Kb , 2012/11/26 00:00

Name bencivenni
Last Modif. 2014/01/09 14:24

Pydio Community - Free non supported version © C. du Jeu 2008-2013 - http://pyd.io/

FIGURE 5.2: Data Management Interface

[1] http://pyd.io/

The interface is divided in four main sections:

- Command Section (on the top) where the user can perform all the possible actions on selected files or directories and chose the VO to use.
- List Section (in the center) where all the files and directory are listed. If the number of files is too big, the output is automatically split in pages in order to not overload the user's web browser.
- Directory Section (on the left) where only the directories are listed: it is useful to quickly browse the tree.
- Detail Section (on the right) where some details on selected folder or file are shown.

By clicking on items in the List Section or in the Directory Section, it is possible to list the content of the LFC. The allowed actions can be grouped in three main categories: operations on the logical data (which affects the catalogues), operations on the physical files (which affect the SE) or both. The basic operations permitted on catalog content with effect only on the logical data are: creation of a new folder, deletion of an empty folder, renaming or moving a folder or file (changing the LFN), getting detailed information about a file (LFN, GUID, list of replicas, owner, level of access) and sharing the file with other portal's users. The basic operation permitted with effect only on the physical data are: the replicas of files on other storage elements and the download of files (see section 5.4). The basic operation permitted on catalog content with effect on both physical and logical data are: the removal and upload of files (see paragraph 5.3).

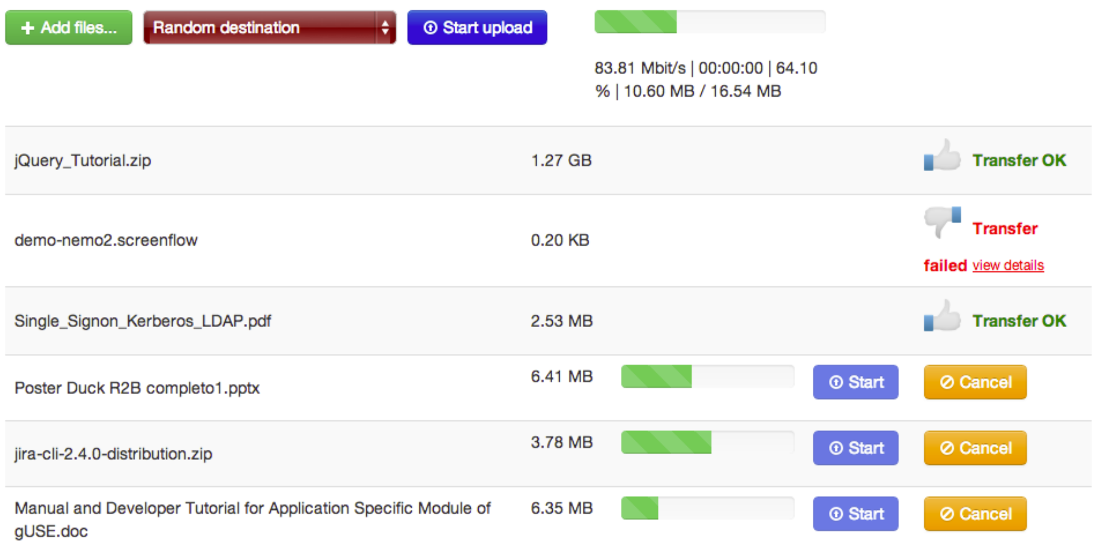
5.3 Upload

In order to upload files to the Grid, the Data Mover implements an external tool: jQuery File Upload[2]. It has been chosen because it allows the upload of large files, of the order of tens of gigabytes, using chunking mechanism and, on browser that support the HTML5[3] function *xmlhttprequest*, it operates also the Resumable Upload, that permits to resume a previous failed upload restarting from the last

[2]https://github.com/blueimp/jQuery-File-Upload
[3]http://www.w3schools.com/html/html5_intro.asp

bit successfully transferred.
Through the upload operation the file is copied on a SE and registered into the LFC. In particular, the file is first uploaded on the SEs Portal and then copied on Grid SE and registered in the catalog.
According to the file size and the VO, the system produces a list of suitable SEs. If the user chose a specific SE the system tries to upload the file to that destination; in case of failure, the system tries to copy the file on some other SEs of the list following a random order. During the upload operation the user can check the transfer rate and the percentage of completeness as shown in Fig. 5.3.
At the end of the transfer the user is notified on the result. There are three possible outcomes:

- *Transfer ok*: the file has been correctly transferred on the selected SE
- *Transfer ok but not on the selected SE*: the file has been correctly transferred in Grid but not on the selected SE
- *Transfer failed*: it has not been possible to transfer the file in any SEs, in this case it is also possible to have more details about the failure.



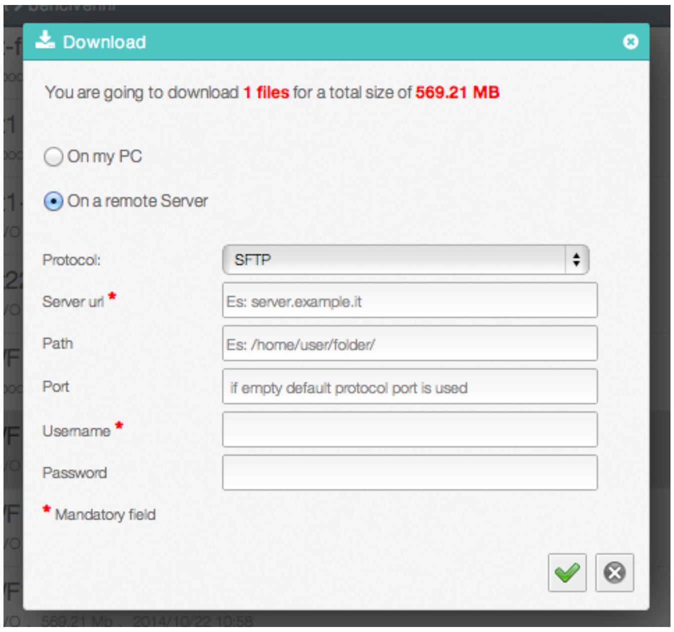
83.81 Mbit/s | 00:00:00 | 64.10 % | 10.60 MB / 16.54 MB

jQuery_Tutorial.zip	1.27 GB			Transfer OK
demo-nemo2.screenflow	0.20 KB			Transfer failed view details
Single_Signon_Kerberos_LDAP.pdf	2.53 MB			Transfer OK
Poster Duck R2B completo1.pptx	6.41 MB		Start	Cancel
jira-cli-2.4.0-distribution.zip	3.78 MB		Start	Cancel
Manual and Developer Tutorial for Application Specific Module of gUSE.doc	6.35 MB		Start	Cancel

FIGURE 5.3: Data Management - Upload interface

5.4 Download

Another feature is the possibility to download files from Grid to a local destination. The file transfer from Grid to user's local space is completely transparent to the user and, like the upload phase, it consists of two steps. The file is first retrieved from Grid, temporarily stored on the Portal SEs and then transferred to the user's local space. The file will be removed at the end of the transfer.It is also possible to retrieve more than one file in a single operation; in this case the system waits until all the files are retrieved from the Grid and then generates a tar file and transfers it to the user local space.
Users can choose to copy the files either to a local destination or to an external server. The transfers to remote servers run in background and the users is notified by mail at the end of the transfer. The transfers to user's PC are syncronous: the users has to wait until the transfer is finished before to reuse the portal. For these reason a five Gigabytes quota limit is currently set as the maximum size to be downloaded on the users' personal computer. In case of transfer to remote server, the Data Mover automatically retries to transfer the files that at the first attempt were not copied. The Supported protocols for the data transfer to external servers are FTP and HTTP plain or over SSL.
Before starting the file transfer to a remote server, a set of checks are performed: username and password verification, if the folder destination exists, if the target server is reachable and if the proxy lifetime is reasonably long to complete the transfer. In order to be sure that the transfers do not fail due to the proxy expiration, a proxy must have a lifetime of 1 hour to transfer up to 2GB of data, of 2 hours for 4 GB and at least 4 hour to move more than 4 GB of data. These values are experimental. If the proxy lifetime does not comply with these roles a pop window appears asking for the proxy renewal. Only if these checks end successfully the data transfer starts. Fig.5.4 shows the window to retrieve files from Grid. The Appendix B describes in detail the script used to move files form Grid to a Remote Server.



You are going to download **1 files** for a total size of **569.21 MB**

( ) On my PC

[✓ On a remote Server]

Protocol:	SFTP
Server url *	Es: server.example.it
Path	Es: /home/user/folder/
Port	if empty default protocol port is used
Username *	
Password	

* Mandatory field

FIGURE 5.4: Data Management - Download interface

Chapter 6

Cloud services

Grid services offered in the portal are to be considered production level while Cloud services are still in development. Through the portal we want to provide two different types of utilization of the Virtual Machines (VMs) managed by several Cloud implementations. They can be used in interactive way: the users login and use the VM for his purpose or they can be used as computing resources for job execution: the users submit a job through the same interface described in 4.2 and the portal chooses if the job has to run in Grid or Cloud environment, depending on the job requirements, in a transparent way for the user.

6.1 Interactive Cloud Service

The IGI Web portal lets users access resources offered by one of more IaaS Cloud providers through a Cloud interface. This interface is based on the WNoDeS Cloud CLI, available since the EMI-3 Montebianco distribution. This CLI, developed in the context of the EGI Federated Cloud Task Force[1], allows the access to resources served by several resource providers, using different Cloud platforms such as WNoDeS, OpenStack or OpenNebula. The users can get a VM independently by the provider, using the same procedure.
The CLI has been designed to be as modular as possible to easily add further Cloud needs and satisfy site peculiarities. Table 6.1 provides a short description of the commands supported by the CLI.

[1]https://wiki.egi.eu/wiki/Fedcloud-tf:FederatedCloudsTaskForce

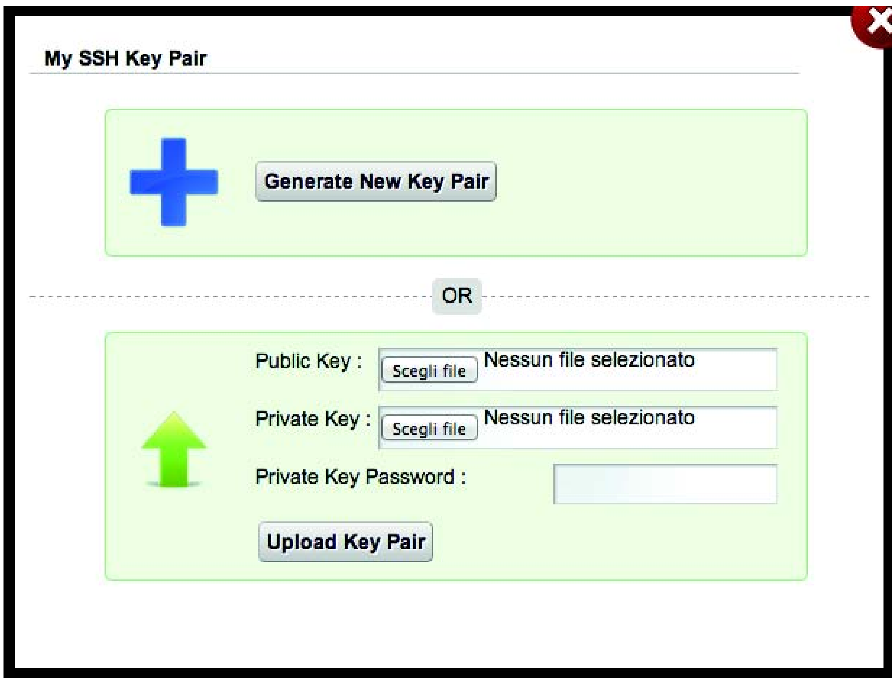
Command	**Description**
resource_providers_info	returns a list of providers registered in the BDII service
size_images_info	for each site returns the images size of the adopted Cloud platform
metadata_images_info	returns a list of the images metadata registered in the MarketPlace service
create_instance	returns the instance location that satisfies the user's request in terms of the image size
show_instance	shows information for the specified instance location such as the hostname, and the instance state
list_instances	returns either the whole instance locations associated to the user who is running the command or the whole Cloud platform resources
delete_instance	destroys the created instance

Table 6.1: Cloud CLI

To use the Cloud CLI services within the portal, a portlet has been developed; it invokes the CLI methods and exposes a Web interface simplifying the task to create and manage new instances.

In order to login into a virtual machine with root privileges and without password, users have to upload their own SSH public key, or require the portal to generate a SSH private/public key pair. Once generated by the portal, the keys can be retrieved by users using a secure channel. This step is necessary otherwise the users are not allowed to do any action in this interface. Fig. 6.1 shows the window to manage the SSH key pair.

Users can instantiate new VMs choosing from a list of images preloaded in a repository, also called MarketPlace (the Stratuslab MarketPlace [26] in the EGI use case). For each image is possible to select a flavour (usually offered with different number of cores, memory and disk size) and how many instances of this type need to be instantiated; each image has a range size that depends on the Cloud platform it belongs to.
Fig. 6.2 shows the page in which is possible to select the image from the repository and the virtual machine features.



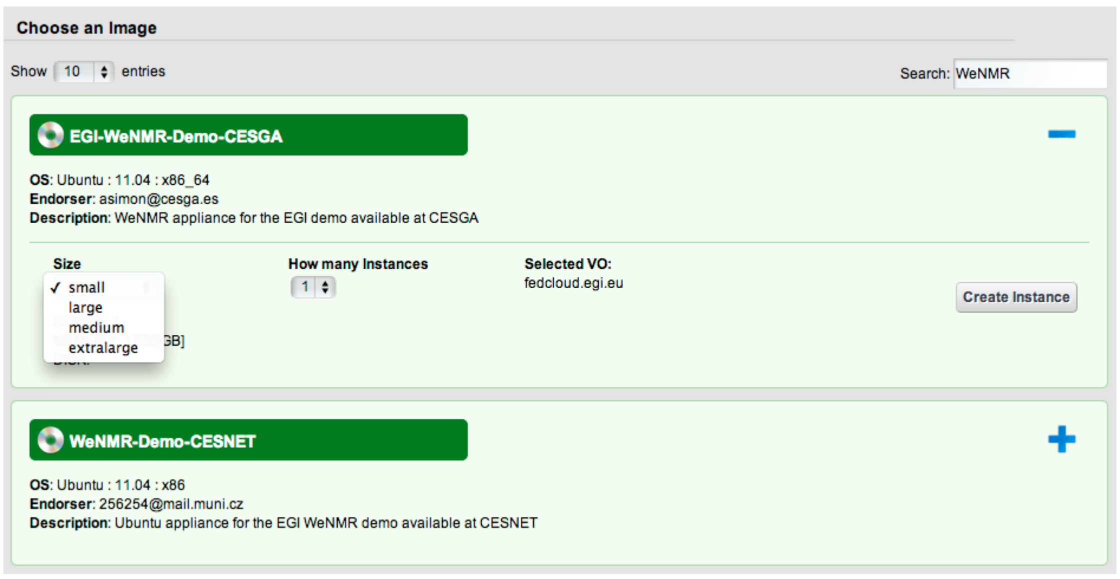
My SSH Key Pair

Public Key : Nessun file selezionato

Private Key : Nessun file selezionato

Private Key Password :

Figure 6.1: Portlet Cloud - SSH Keys management



Choose an Image

Show 10 entries

Search: WeNMR

EGI-WeNMR-Demo-CESGA

OS: Ubuntu : 11.04 : x86_64
Endorser: asimon@cesga.es
Description: WeNMR appliance for the EGI demo available at CESGA

Size: ✓ small

How many Instances: 1

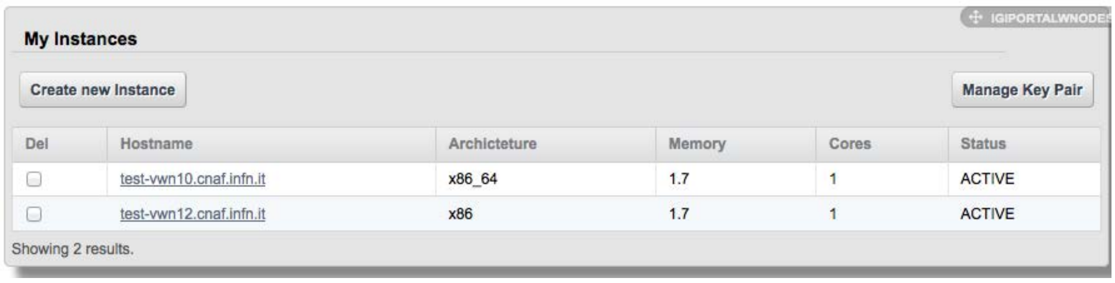
Selected VO: fedcloud.egi.eu

WeNMR-Demo-CESNET

OS: Ubuntu : 11.04 : x86
Endorser: 256254@mail.muni.cz
Description: Ubuntu appliance for the EGI WeNMR demo available at CESNET

Figure 6.2: Portlet Cloud - Image repository

Once new instances are created, the list of user's VMs is displayed with information such as architecture, size and status as shown in Fig. 6.3.

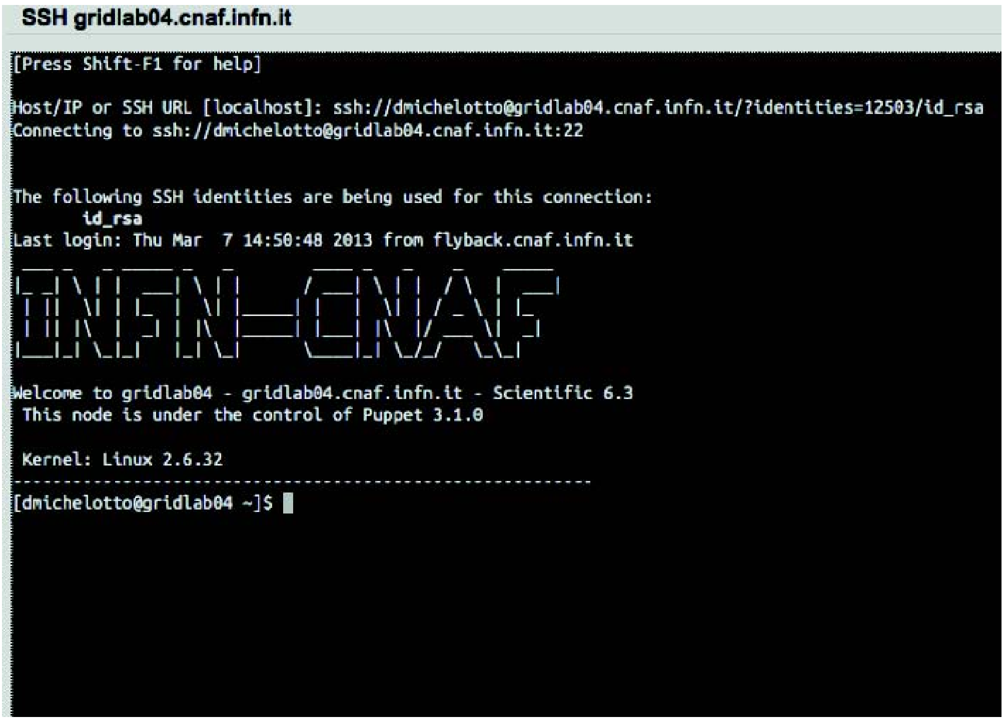


Del	Hostname	Archicteture	Memory	Cores	Status
☐	test-vwn10.cnaf.infn.it	x86_64	1.7	1	ACTIVE
☐	test-vwn12.cnaf.infn.it	x86	1.7	1	ACTIVE

Showing 2 results.

FIGURE 6.3: Portlet Cloud - VMs Lists

By selecting the instance name, users are automatically logged into the virtual machine with root privileges through a Web terminal[2] and it is part of the portal. Fig. 6.4 shows an example of web terminal for a VM instantiated through the portal.



```
[Press Shift-F1 for help]

Host/IP or SSH URL [localhost]: ssh://dmichelotto@gridlab04.cnaf.infn.it/?identities=12503/id_rsa
Connecting to ssh://dmichelotto@gridlab04.cnaf.infn.it:22


The following SSH identities are being used for this connection:
      id_rsa
Last login: Thu Mar  7 14:50:48 2013 from flyback.cnaf.infn.it
 ___ _   _ _____ _   _        ____ _   _    _    _____
|_ _| \ | |  ___| \ | |      / ___| \ | |  / \  |  ___|
 | ||  \| | |_  |  \| |_____| |   |  \| | / _ \ | |_
 | || |\  |  _| | |\  |_____| |___| |\  |/ ___ \|  _|
|___|_| \_|_|   |_| \_|      \____|_| \_/_/   \_\_|

Welcome to gridlab04 - gridlab04.cnaf.infn.it - Scientific 6.3
 This node is under the control of Puppet 3.1.0

 Kernel: Linux 2.6.32
-----------------------------------------------------------
[dmichelotto@gridlab04 ~]$ 
```

FIGURE 6.4: Portlet Cloud - Web terminal

While the portal lets users choose VM images offered by different Cloud providers, the definition of policies dictating how the resulting VMs are connected to the network (e.g. with public IP addresses or with private IP addresses using NAT)

[2]the Web terminal used is GateOne: http://liftoffsoftware.com/Products/GateOne

is left to each resource provider.
A demo of the Cloud interface has been shown at the EGI Community Forum 2013[3][4] and succesfully adopted by communities such as WeNMR[5] [27].

6.2 Cloud resources for the Jobs execution

In order to use the Cloud resources to execute the jobs submitted through the appropriate portal interface, we have used a specific DIRAC Component: Virtual Machine Scheduler [35]. This component has to be installed in DIRAC server separately from the other used for job submission and it allows to "submit" a VM to a generic Cloud Manager compliant to Open Cloud Computing Interface[6] (OCCI) compliant or Amazon EC2[7]. The model for instantiating VMs uses the same philosophy of the pilot jobs in the Grid infrastructure. Fig. 48 shows the DIRAC architecture relative at this component.
Since the DIRAC configuration is static, the administrator has to uploaded the images to the Cloud Manager and set the images features in the DIRAC configuration file. When a new job is added to the task queue this component gets the list of tasks to be executed from the central Task Queues, by matching the pending tasks to the features of the defined images in the configuration file. If not enough VMs are available and the maximum VMs threshold is not reached, then it submits a new VM using the specific VM Director.
The Virtual Machine Scheduler starts a VM with pre-installed some DIRAC agents: Job Agent and Monitor Agent. Job Agent is responsible to communicate with the central Task Queue to check if there are pending tasks to be executed which match the VM features and supervises the correct execution of the job running on the VM.
The Monitor Agent starts immediately after the operating system of the new VM created is running. Its first action is to declare the VM in running state, it periodically monitors the CPU load, the number of executed tasks and the amount of output data. In case of problems it reports the malfunctioning to DIRAC WMS which halts the VM.

[3]EGI Community Forum 2013: http://cf2013.egi.eu
[4] EGI FedCloud Task Force Demo
[5] A worldwide e-Infrastructure for NMR and structural biology: http://www.wenmr.eu
[6]http://occi-wg.org/
[7]http://aws.amazon.com/ec2/

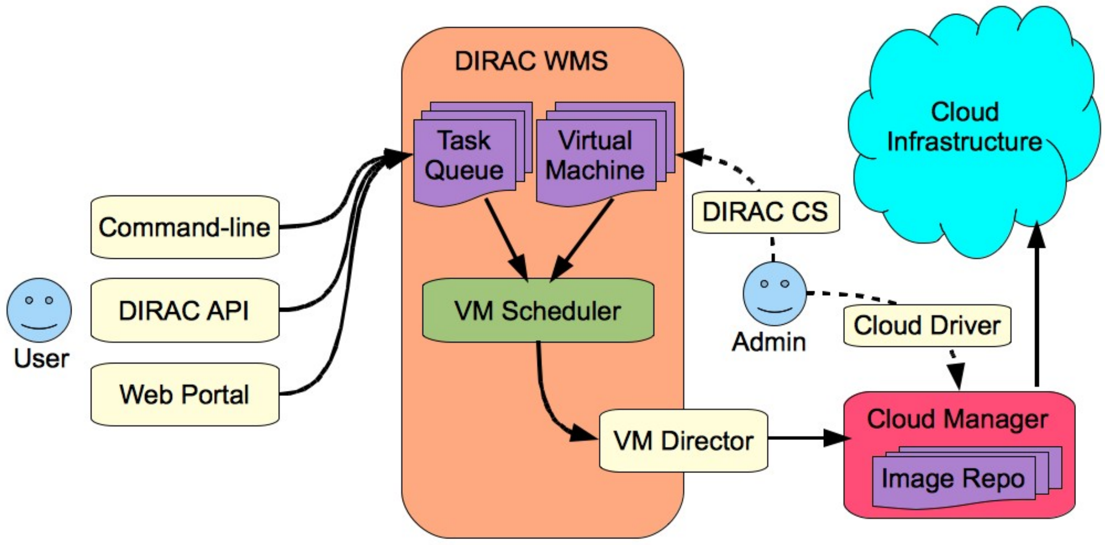


FIGURE 6.5: Dirac - Cloud components

Using DIRAC for both Grid job submission and Clod job submission gives the advantage of using a single portlet for the job submission independently from the infrastructure used. It is DIRAC that chooses which resources fit most the job requirements based on the information declared by the administrator in the DIRAC configuration file. This behaviour provides an high level of abstraction that helps the users to exploit the different resource types.

Chapter 7

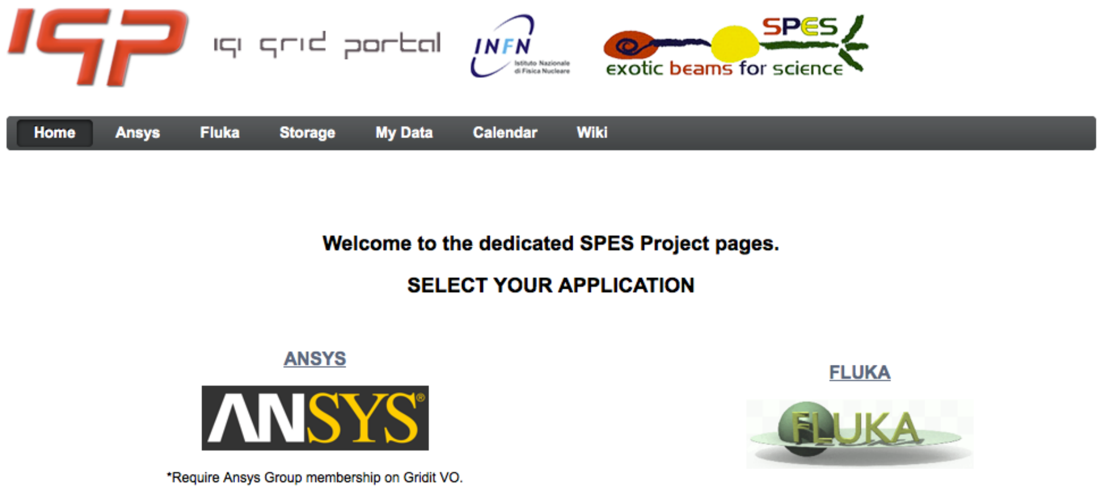
Use cases

The Grid infrastructures, initially designed to satisfy the computational and storage requirements of the HEP communities, are now increasingly adopted by new user communities belonging to various scientific domains: computational chemistry, bioinformatics, astronomy and astrophysics, earth science, mathematics, engineering etc. These communities have different computational and storage requirements, composition, experience and size. Often it is necessary to provide dedicated support developing specific high level Web interfaces for their computing models to hide the Grid complexity as much as possible, such as proxy credential handling, job submission, data management, error recovery, etc.
In the typical use case the users has to provide the initial input files, configuration parameters and wait for results at the end of the calculation. For these reasons we have developed interfaces that enable the user to do that in 3 easy actions:

- uploading the needed input files and setting the necessary parameters (e.g. the number of CPUs);
- monitoring the status of the submitted jobs;
- retrieving the output files directly from the Web Portal.

Accordingly to the application requirements we can decide to use a workflow or normal jobs. Some specialized workflows developed are particularly complex, they are able to run in an automated way by evaluating dependences occurring at runtime.
For example we are collaborating with a working group at INFN-Legnaro involved

in the Selective Production of Exotic Species[1] (SPES) project, providing them 2 interfaces ad hoc for 2 applications: ANSYS[2] and FLUKA[3]. We have created a mini-site, shown in Fig. 7.1 inside the general web portal for these users who have the privileges to see only these private pages. In this way they are able to use only the required services for their needs and choose which applications to use.
The two applications mentioned are interesting because they cover several typical user requirements: ANSYS provides the possibility to check the logs and output files at runtime and to have a start and stop mechanism. FLUKA offers the possibility to submit hundreds of job at the same times which run simultaneously reducing drastically the computation time.



Welcome to the dedicated SPES Project pages.

SELECT YOUR APPLICATION

*Require Ansys Group membership on Gridit VO.

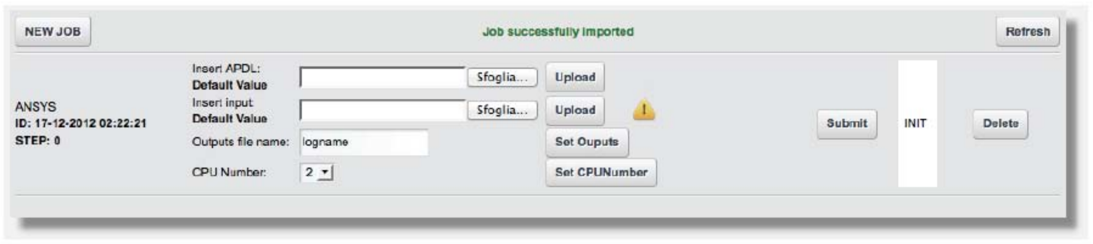
FIGURE 7.1: The dedicated SPES section in the portal

7.1 ANSYS

The ANSYS suite is a Finite Element Method (FEM) commercial program for simulations of models belonging to various physical environments to simulate problems concerning mechanical, thermal, electrical, magnetic and fluid dynamics matters. Depending on the model to be simulated, various packages of the ANSYS suite exist: the Mechanical, Fluid Dynamics, Electromagnetic and Multiphysics. Very often observable properties are the results of averaging (or integrating) over energies, time, etc. which means that ANSYS runs have to be repeated a large

[1]https://web.infn.it/spes/
[2]website: http://www.ansys.com/
[3]http://www.fluka.org/fluka.php

number of times making the exploitation of the distributed resources available in Grids highly effective for this kind of analyses [29]. The ANSYS suite has been installed and configured in some Grid sites and a dedicated interface to run the simulation exploiting the production Grid services is provided through a dedicated portlet of the IGI Portal shown in Fig. 7.2.


Job successfully imported

ANSYS
ID: 17-12-2012 02:22:21
STEP: 0

Insert APDL: **Default Value**
Insert input **Default Value**
Outputs file name: logname
CPU Number: 2

INIT

FIGURE 7.2: Portlet developed for the ANSYS suite

To be compliant with the terms of license imposed by the seller, a license handling mechanism has been implemented. In order to verify if the threshold of simultaneous usage has not been reached, the portal has to contact the Flex servers that manages the licences for every job submission. Moreover only the users registered in VO “gridit” and in ANSYS group can run this application. The main problem to deal with is the amount of time necessary to run this application: from several hours to some days. Since the amount of CPU time is limited on Grid sites (from 12 hours to few days), we have implemented a mechanism in order to run the application and monitoring the computation time allowing a safely interruption of the application.

To do this we have developed a workflow, using the WS-Pgrade workflow engine, able to monitor a set of continuous runs in an automated way.

The workflow is composed of 10 nodes which means 10 consecutive job submissions that is equivalent to an average of 10 days of continuative calculation for a single run. Each step of workflow is conditioned by event-related dependencies occurring at runtime which determine the execution of the successive node or not.

If the simulation is not finished after the last step, the user can submit a new workflow that automatically takes as input the outcomes of the previous run. This approach increases the feeling of the users with the submission procedures with a consequent reduction of support requests.

On the base of this workflow a suitable graphical interface has been developed to set the needed parameters that makes the whole Grid execution process completely transparent to the final user. Another crucial aspect of such long time

simulations is the possibility to check the calculations at runtime. In order to do that we took advantage of the SRM client-server functionalities which allow to copy selected files from the WN where the job is physically running to a SE. Using the same SRM functionalities, the files in the SE are made available for inspection at runtime and can be accessed by the user directly via the web GUI. The main bash script which interact with the GUI are described in appendix A.
The solution implemented has a twofold advantage for the users: easiness and rapidity. The users through this interface are able to perform long time simulations on the Grid infrastructure in a completely transparent way and to retrieve the outcomes of the calculation with a Web browser. In this way the users can check the consistency of the output at runtime, evaluating possible strategies aimed at saving time, computing resources and at avoiding waste of license usage.
The average execution time for a single instance of ANSYS on the Grid is almost equivalent to the one on a dedicated local machine.The added value of Grid runs is the possibility to submit in parallel different sets of concurrent simulations. Accordingly, as soon as there are at least 4 ANSYS jobs running simultaneously, the Grid based execution is advantageous because 4 jobs running simultaneously on 4 IGI Grid resources will take on average 8 days. Meanwhile, the same simulation would take about 1 month on a local machine. The more parameter study jobs are executed, the higher speedup can be achieved on the Grid. Fig. 7.3 shows an example of the output produced using the ANSYS application.

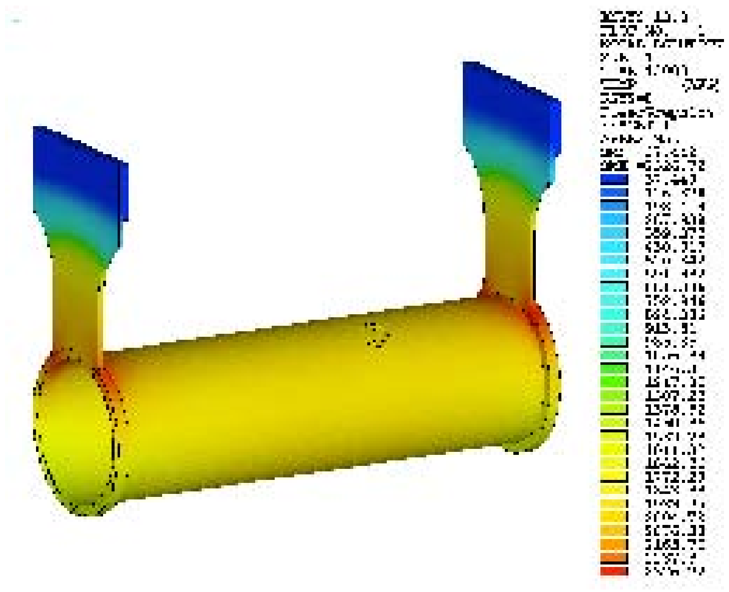


FIGURE 7.3: Example of ANSYS Output
(Temperature map carried out from the adoption of the FEM model at 1300A)

7.2 FLUKA

FLUKA is a fully integrated particle physics MonteCarlo simulation package. It has many applications in high energy experimental physics and engineering, shielding, detector and telescope design, cosmic ray studies, dosimetry, medical physics and radio-biology. FLUKA can simulate with high accuracy the interaction and propagation in matter of about 60 different particles.
The FLUKA applicative has the features to be linear and parameterizable on the base of the input files and for these reasons we decided to use DIRAC as a service for the submission. In this case we have reused the DIRAC portlet as base, omitting all the fields not useful for this simulation and integrating some specific checks. Because the FLUKA applicative needs input files to run, the first time SPES users access the FLUKA portal interface they have to select their working directory that represents a folder in LFC or create a new one. Once this action is completed the users have to upload the files necessary, in archive format (.tar or .tgz) or for the simulation in this directory using the portal data management service.
The input file must be generated from the user and adhere to the following format:

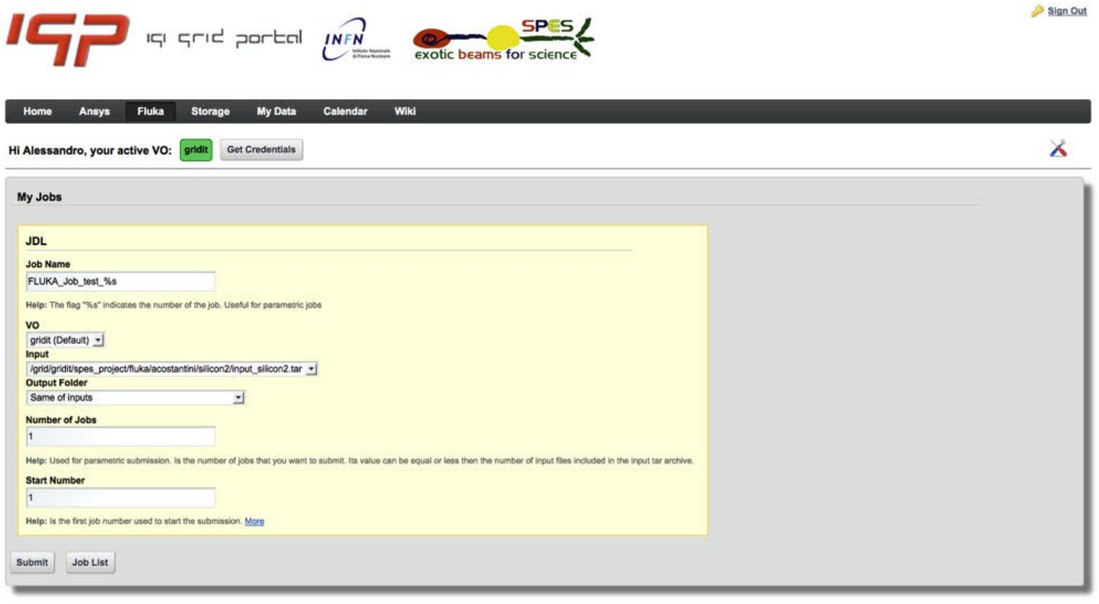
```
input_<archive_name>.tar
```

where archive_name is a string of alphanumeric chars chosen by the users. This archive contains the input flies in accordance with this format:

```
<nome_archivio>_<N>.inp
```

where ¡N¿ is an integer number and the number must be consecutive inside the same archive. The users are now able to submit the application from the dedicated interface, shown in Fig. 7.4, setting the necessary fields like:

- *Job Name*: an identifier to distinguish the jobs submitted,
- *Input*: from a dropdown menu the users have to choose one of the file uploaded in their directory,
- *Output folder*: the LFC directory where the output will be saved. As default it will be used the same directory of the input file.



Hi Alessandro, your active VO: gridit

My Jobs

JDL

Job Name
FLUKA_Job_test_%s

Help: The flag "%s" indicates the number of the job. Useful for parametric jobs

VO
gridit (Default)

Input
/grid/gridit/spes_project/fluka/acostantini/silicon2/input_silicon2.tar

Output Folder
Same of inputs

Number of Jobs
1

Help: Used for parametric submission. Is the number of jobs that you want to submit. Its value can be equal or less then the number of input files included in the input tar archive.

Start Number
1

Help: Is the first job number used to start the submission. More

FIGURE 7.4: Portlet developed for the FLUKA application

- Number of Jobs and Start Number.

After filling the form the job is ready to be submitted and at this point it is possible to monitor its states that can be:

- *Waiting*: the job has been taken in charge from the portal.
- *Running*: the job is running in the Grid site specified,
- *Failed*: the job is failed
- *Done*: the job has finished correctly: there have not been failures due to the portal or Grid. The portal does not consider the failures of the applicative.

Once the simulation is finished the jobs are in state "Done". The users in the "Data Management" section can download the output produced. In order to avoid to have file with the same name, the output file has the following format:

```
<archive_name>_<N>_out_<MMDDYY-HHMMSS>-<2to10_random_digits>.tgz
```

In this use case we can take benefit from the linearity of the application. The entire job has been splitted in smaller job, each one takes as input a portion of the entire input data and in this way the jobs submitted lasts only some hours.

The advantage consists of the possibility to run these jobs simultaneously having a consistent reduction on computational time. The more resources are available at the same time, the bigger is the speed up. Fig. 7.5 shows an example of the output produced using the FLUKA application.

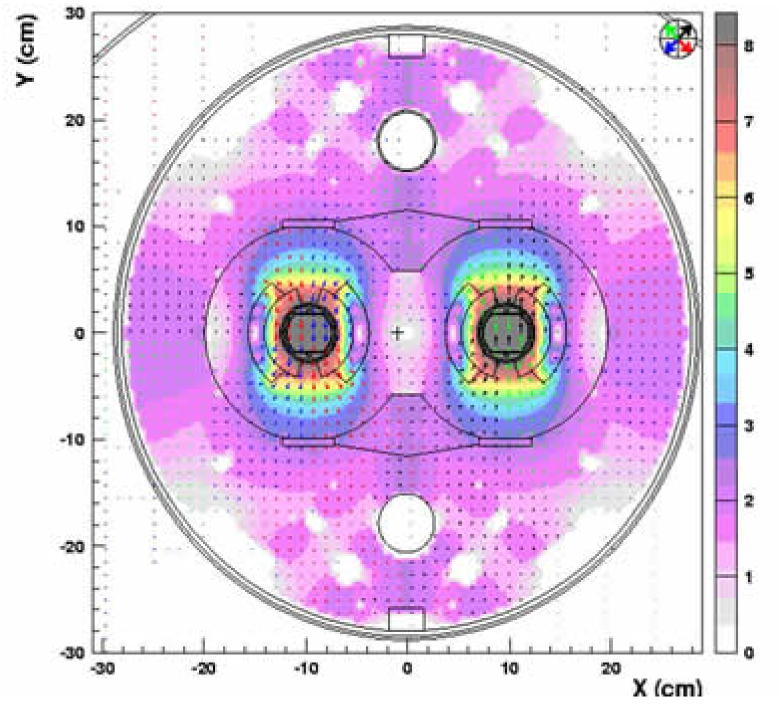


FIGURE 7.5: Example of FLUKA Output
(Output of Fluka simulations by EN/STI/EET team)

Chapter 8

Conclusions and future developments

8.1 Conclusions

This document describes a web portal mainly targeted at scientific communities making use of distributed resources and its benefits for its users. Through the portal, users can access computational resources made available by Grid communities or by Cloud resource providers, submit either simple or complex jobs as well as workflows, move data to and from the Grid resources and finally access to specific applications through custom interfaces. This document also describes some examples of practical usage of the portal by several scientific communities.
Moreover it has been developed a prototype portal with an integrated online CA in order to reduce the complexity of obtaining and managing certificates which is an important feature to make this tool suitable for a much greater audience and increase the number of users.
Being the portal a service accredited in international federations (EDUgain) it can provide the described services to different user classes outside the national borders, e. g. international collaborations or different NGIs.
The use cases described shows that the nature of Grid heterogeneity can be exploited when porting scientific applications on distributed platforms. These include the possibility of segmenting large calculations into smaller ones and the selective distribution of the segments on different machines in order to gain efficiency as well as feasibility. This makes the work performed for one application

more general and reusable for other applications whose features are similar.
The present work of integration of applications into the portal is going to be extended in order to port the current and new user applications to different OS flavours, as well as to different hardware platforms.

8.2 Future developments

The portal was initially developed to overcome the biggest Grid problems which are the access to resources belonging to the Grid environment. The current and future works will be oriented to Cloud resources integration which means:

- extending the Cloud high-level interface for the direct provisioning of SaaS type-of services;
- implementing a tighter integration with Cloud frameworks such as OpenStack or OpenNebula. This will allow a more fine-grained specification of the resource requirements supported by such frameworks, such as those related to network connectivity;
- improving support for scalable and distributed Grid and Cloud storage systems;
- extending the Cloud interface to support non-Linux systems.

Some improvements concern also the data management section:

- supporting data transfer across Grid and Cloud resources;
- improving support for scalable and distributed Grid and Cloud storage systems;
- enhancing the reporting for error status linked to storage provisioning;
- providing for each user or group of users a catalog with metadata since LFC will be dismissed in the next months;
- allowing the upload of directories/files from a remote server to a Grid SE

Thanks to the modularity of the portal architecture and to the use of widely adopted frameworks, the portal can easily be extended to integrate or being interfaced to new services by developing appropriate portlets.

Appendix A

Script for running ANSYS in Grid

In this appendix, are shown the bash scripts developed to run ANSYS suite in Grid. The bash functions are executed directly on the computing node and can interact directly with the selected SE without involving the WMS and avoiding the potential limitations described in chapter 4. The main bash functions interacting with the SE are:

- SETENV Checks the environment parameters (both provided by the GUI and set in the WN) needed to start the calculation
- USERFOLDER Checks if user-folder exists using the SRM-client functionalities and create it, if needed;
- PREPARETOPUT Uses the PreparetoPut function implemented in the SRM-client to copy a rewritable file(s) in the SE and store it for a limited amount of time (lifetime has been set by default to 24 hours);
- PREPAREINPUT Retrieves the input file needed for the calculation depending on the value provided by the GUI (first run or resubmission after hang-up);
- RUNNINGAPP Runs the executable monitoring its activity for a fixed amount of time, that depends from the CPU time assigned by the batch system to the queue where the job is running, and gracefully kill it allowing a safe

interruption of the application. This component implements a check procedure that, at fixed intervals, control the status of the running application and uploads the needed file(s) to the SE.

- CHECKLOGS Uploads the file(s) created by PREPARETOPUT function to the SE making use of specific commands available on the WNs.
- PREPAREOUTPUT The function manages the output files carried out from the calculations acting on both sides: from the WN selects the output files making a univocally named archive; from the SE (i) removes the oldest output file, if it exists, (ii) rename the output file carried out from the previous run assigning a proper name, (iii) copies the actual output file from the WN to the SE (in this case the file lifetime has been set by default to 7 days).

```
#!/bin/bash
echo ""
echo "Script ANSYS start"
echo "Script 2"
echo "Set variables"
echo ""
#assign arguments to variable name
output=$1
userfolder=$2
url=$3
# variables
#output_last
outputl=$1"_last"
# APDL
APDL="Restart Job; APDL not neened"

#get the ncpus from PBS
ncpus=$(wc -l $PBS_NODEFILE | awk '{print $1}')

#get walltime from PBS
maxtimequeue=$(qmgr -c "list queue "$PBS_QUEUE |grep default.walltime |
awk '{print $3}'| awk -F\: '{print $1 *3600}')
#set it if not specified
if [ "x$maxtimequeue" == "x" ]
then
maxtimequeue=43200
fi
################
runtime=7200
ctrltime=1800
################
#killing time
killtime=$runtime
#ansys bynary
progID="/opt/exp_soft/gridit/ansys_inc/v130/ansys/bin/ansys130 "
bynID="ansys.e130"
#closese=$SE_HOST
closese="darkstorm.cnaf.infn.it"
#clientSRM
clientSRM="/usr/bin/clientSRM"
#print variables
echo "APLD: " $APDL
echo "output: " $output
echo "user folder: " $userfolder
echo "URL: "$url
echo "cpu unumber: " $ncpus
echo "max runtime queue: " $maxtimequeue
echo "max runtime imposed: " $maxtime
echo "effective runtime: " $runtime
echo "kill time: " $killtime
echo "control time: "$ctrltime
echo "ansys path: "$progID
echo "ansys binary: "$bynID
echo "used SE: "$closese
# Set automatic resubmission to FALSE by default
touch rstauto
echo "FALSE" > rstauto
```

```
# Set LD_LIBRARY_PATH in the WN
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/exp_soft/gridit/ansys_inc/
libs/64:/opt/exp_soft/gridit/ansys_inc/libs/32
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/exp_soft/gridit/ansys_inc/
v130/ansys/syslib/linx64:/opt/exp_soft/gridit/ansys_inc/v130/
commonfiles/Tcl/lib/linx64
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/exp_soft/gridit/ansys_inc/
v130/ansys/lib/linx64:/opt/exp_soft/gridit/ansys_inc/v130/Framework/bin/Linux64
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/exp_soft/gridit/ansys_inc/
v130/Framework/bin/Linux64/Mesa
export LD_LIBRARY_PATH

#FUNCTIONS

##USERFOLDER
#check if userfolder exists via clientSRM
function user_folder()
{
echo ""
echo "Check if folder "$userfolder" exists"
#
ufexist=$($clientSRM Ls -e httpg://$closese:8444/ -s
srm://$closese:8444/ansys/$userfolder | grep "status: statusCode=" |
 head -n 1 | awk -F\( '{print $NF}' | awk -F\) '{print $1}')
if [ "$ufexist" -ne "0" ]; then
 echo "Dir "$userfolder" does not exist... Potential incongruence found"
 echo "This is a true resubmission, the Dir"$userfolder" has to be present in the SE."
 echo "Contact the Admin."
 echo "Exit script Err. 113"
 echo ""
 exit 113
else
echo $userfolder" already present in the SE:"$closese
echo ""
fi
}

##PREPARETOPUT
#PreparetoPut function
function ptp_log()
{
#remove file, if exists in the SE
echo ""
echo "Remove "$output".log on the SE "$closese
$clientSRM rm -e httpg://$closese:8444/ -s
srm://$closese:8444/ansys/$userfolder/$output".log"
echo "Done"
#log file -- 24h lifetime
echo ""
echo "Create file "$output".log on the SE "$closese
touch $output".log"
echo "File ready to be copyed" > $output".log"
$clientSRM PtP -e httpg://$closese:8444/ -w 1 -c 86400 -p -s
srm://$closese:8444/ansys/$userfolder/$output".log" > ptp_log.txt
#extract turl
turllog=$(cat ptp_log.txt |grep "TURL" | awk -F\" '{print $2}')
#"#extract token
tokenlog=$(cat ptp_log.txt | grep "requestToken" | awk -F\" '{print $2}')
#"#
try=1
while [ "$try" -lt "4" ]; do
 globus-url-copy file:`pwd`/$output".log" $turllog
 if [ "$?" -ne "0" ]; then
  let try=$try+1
  if [ "$try" -lt "4" ]; then
   echo "Problem copying file "$output".log ... retry in few seconds"
  else
   echo "Problem copying file "$output".log ... no more tentatives"
   echo "Exit script Err. 114"
   $clientSRM pd -e httpg://darkstorm.cnaf.infn.it:8444 -s
    srm://darkstorm.cnaf.infn.it:8444/ansys/$userfolder/$output".log" -t $tokenlog
   echo ""
   exit 114
  fi
 else
  echo "File "$output".log successfully copied to the SE "$closese
  echo ""
  try=5
 fi
done
$clientSRM pd -e httpg://darkstorm.cnaf.infn.it:8444 -s
srm://darkstorm.cnaf.infn.it:8444/ansys/$userfolder/$output".log" -t $tokenlog
}

##PREPAREINPUT
#prepare input
function pre_input()
{
echo ""
echo "Trasnfering file "$url" from the SE "$closese
try=1
while [ "$try" -lt "4" ]; do
curl --cert $X509_USER_PROXY --capath /etc/grid-security/certificates
-o `pwd`/input.tar.gz $url
 if [ "$?" -ne "0" ]; then
  let try=$try+1
  if [ "$try" -lt "4" ]; then
   echo "Problem transfering file "$url" ... retry in few seconds"
  else
   echo "Problem transfering file "$url" ... no more tentatives"
   echo "Exit script Err. 114"
   echo ""
   exit 114
  fi
 else
  echo "File "$url" successfully transferred from the SE "$closese
  echo ""
  try=5
 fi
done
tar -zxf input.tar.gz
echo "List content"
ls -la
}

##RUNNINGAPP
#sample script to start a program, permit it to run
#for a predefined amount of CPU time, then kill it.
function ansys_run()
{
# generation of input commands file used to tun ansys
echo ""
echo "Preparing command file"
# generation of command file used to tun ansys
echo "y" >> commands
echo "FINISH" >> commands
echo "RESUME,file,db" >> commands
echo "/CONFIG,NRES,1000000" >> commands
echo "/POST1" >> commands
echo "SET,LAST" >> commands
echo "*GET,nsub,ACTIVE,0,SET,SBST" >> commands
echo "*GET,nloa,ACTIVE,0,SET,LSTP" >> commands
echo "/SOLU" >> commands
echo "ANTYPE,TRANS,REST,nloa,nsub-1,CONTINUE" >> commands
echo "OUTRES,NSOL,ALL" >> commands
echo "VFOPT,READ" >> commands
echo "SOLVE" >> commands
echo "y" >> commands
echo "SAVE" >> commands
echo "Command file ready"
echo""
ansys=" -np "$ncpus" < commands > "$output".log"
#run it!
echo "Running program: ANSYS"
echo $progID $ansys
$progID $ansys & echo $! > ansyspid
#application pid
mypid=$(cat ansyspid)
echo "PID ansys130 is "$mypid
sleep 60
mypidchild=$(ps --ppid $mypid | grep -iv "PID" | awk '{print $1}')
echo "PID ansys.e130 is "$mypidchild
#first chack log
echo "Starting check at runtime..."
log_check

#start time
secs=0
#control loop
while [ $secs -lt $killtime ]; do
 sleep $ctrltime
#check if the application is running
 mypidcontrol=$(ps --ppid $mypid |grep -iv "PID" | awk '{print $1}')
 echo "PID ansys.e130 is "$mypidcontrol
 if [ "x$mypidcontrol" == "x" ]; then
  echo ""
  echo "Simulation ended before the assigned time"
  echo ""
  let secs=$killtime+1
 else
  rmin=$(eval ps --ppid $mypid |grep -iv "PID" |
   awk '{print $3}' |awk -F\: '{print $2}')
  curmin=$(eval ps $mypidcontrol |grep -iv "PID"|
   awk '{print $4}'|awk -F\: '{print $1}')
  let secs2=$curmin*60
  cursec=$(eval ps $mypidcontrol |grep -iv "PID"|
   awk '{print $4}'|awk -F\: '{print $2}')
  let secs3=$cursec
  let secs=$secs2+$secs3

  #output_log check
  #
  echo ""
  echo "Starting check at runtime; "$secs " seconds elapsed time"
  #
  log_check
 fi
#
#end
done
#last check if the application is running
mypidcontrol=$(eval ps --ppid $mypid |grep -iv "PID" | awk '{print $1}')
if [ -n "$mypidcontrol" ]; then
# kill the application
 echo ""
 echo "The job reached the CPU time assigned to the simulation "
 echo "The job will be killed"
 echo "The above errors are armless..."
 echo "Kill ANSYS. PID: "$mypidchild
 kill -s 2 "$mypidchild"
 sleep 10
 echo ""
fi
# wait until the application is running
echo "Check il the application is really terminated"
check=1
while [ "$check" -eq "1" ]; do
 mypidcheck=$(eval ps --pid $mypid |grep -iv "PID" | awk '{print $1}')
 if [ -n "$mypidcheck" ]; then
  echo "Application still up... retry in 10 min."
  sleep 600
 else
  echo "Application terminated."
  check=4
 fi
done
}

##CHECKLOGS
# check logs
function log_check()
{
echo "Copy file "$output".log to the SE "$closese
try=1
while [ "$try" -lt "4" ]; do
 globus-url-copy file:`pwd`/$output".log" $turllog
 if [ "$?" -ne "0" ]; then
  let try=$try+1
  if [ "$try" -lt "4" ]; then
   echo "Problem copying file "$output".log ... retry in few seconds"
  else
   echo "Problem copying file "$output".log ... no more tentatives"
  fi
 else
  echo "File "$output".log successfully copied to the SE "$closese
  echo ""
  echo ""
  try=5
 fi
done
}

##PREPAREOUTPUT
#prepare output
function ansys_out()
{
echo "Simulation ended, preparing output file..."
echo "Update file "$output".log to the SE "$closese
log_check
rm input.tar.gz
#remove file output_last, if exists in the SE
echo ""
echo "Remove "$outputl".tar.gz on the SE "$closese
$clientSRM rm -e httpg://$closese:8444/ -s
srm://$closese:8444/ansys/$userfolder/$outputl".tar.gz"
#insert control...to do
echo "Done"
echo ""
#move file output to output_last in the SE
echo ""
echo "Move "$output".tar.gz on the SE "$closese" to "$outputl".tar.gz"
$clientSRM mv -e httpg://$closese:8444/ -s
srm://$closese:8444/ansys/$userfolder/$output".tar.gz" -t
srm://$closese:8444/ansys/$userfolder/$outputl".tar.gz"
#insert control...to do
echo "Done"
echo ""
```

```
#remove file output, if exists in the SE ?
echo ""
echo "Remove "$output".tar.gz on the SE "$closese
$clientSRM rm -e httpg://$closese:8444/ -s srm://$closese:8444/ansys/$userfolder/$output".tar.gz"
echo "Done"
echo ""
# create new output file
tar -zcf $output.tar.gz --exclude='stdout.log'
--exclude='stderr.log' --exclude='gridnfo.log'  --exclude='execute.bin'
--exclude='wrapper.sh' --exclude='commands'
--exclude='ansys.out' --exclude='ansys.err'
--exclude='ansys2srm_restart5.sh' --exclude='ansyspid' --exclude='rstauto' *
echo "Output file "$output".tar.gz ready"
echo "Prepare to move "$output".tar.gz on the SE "$closese
#PtP and output transfer -- 7days lifetime
$clientSRM PtP -e httpg://$closese:8444/ -w 1 -c 604800 -p -s
srm://$closese:8444/ansys/$userfolder/$output".tar.gz" > ptp_out.txt
#TURL
turlout=$(cat ptp_out.txt |grep "TURL" | awk -F\" '{print $2}')
#"#Token
tokenout=$(cat ptp_out.txt | grep "requestToken" | awk -F\" '{print $2}')
#"#
try=1
while [ "$try" -lt "4" ]; do
 globus-url-copy file:`pwd`/$output".tar.gz" $turlout
 if [ "$?" -ne "0" ]; then
  let try=$try+1
  if [ "$try" -lt "4" ]; then
   echo "Problem copying file "$output".tar.gz ... retry in few minutes"
   sleep 300
  else
   echo "Problem copying file "$output".tar.gz ... no more tentatives"
   echo "Exit script Err. 114"
   echo ""
   exit 114
  fi
 else
  echo "File "$output".tar.gz successfully copied to the SE "$closese
  try=5
 fi
done
$clientSRM pd -e httpg://darkstorm.cnaf.infn.it:8444 -s
srm://darkstorm.cnaf.infn.it:8444/ansys/$userfolder/$output".tar.gz" -t $tokenout
# Control for resubmission: endtime
endtime=$(cat end_TIME)
# Stop automatic resubmission if endtime is not set
if [ "x$endtime" == "x" ]; then
 echo ""
 echo "******"
 echo "TIME variable has not been correctly set from the previous run."
 echo "Automatic resubmission aborted. Set rstauto to FALSE"
 echo "******"
 echo ""
 echo "FALSE" > rstauto
else
# Control for resubmission: realtime
 realtime=$(cat $output".log" |grep "TIME =" |tail -n1 |
  awk '{print $4}' |awk -F. '{print $1}')
# Stop automatic resubmission if realtime is not set
 if [ "x$realtime" == "x" ]; then
  echo ""
  echo "******"
  echo "It has not been possible extract the actual TIME value from the "$output".log file."
  echo "Automatic resubmission aborted. Set rstauto to FALSE"
  echo "Please, have a look to the "$output".log for more details"
  echo "******"
  echo ""
  echo "FALSE" > rstauto
 else
# realtime is set, evaluating...
  echo ""
  echo "******"
  echo "Calculation ended at TIME = "$realtime" of "$endtime
  if [ "$realtime" -lt "$endtime" ]; then
   echo "Resubmission in progress. Set rstauto to TRUE"
   echo "******"
   echo ""
   echo "TRUE" > rstauto
   echo $realtime >> rstauto
   echo $endtime >> rstauto
  else
   echo "Resubmission not needed. Set rstauto to FALSE"
   echo "******"
   echo ""
   echo "FALSE" > rstauto
   echo $realtime >> rstauto
   echo $endtime >> rstauto
  fi
 fi
```

```
fi
}
#
# REAL RUN
#
#invoke user_folder function
user_folder
#invoke ptp_log function
ptp_log
#invoke pre_input function
pre_input
#invoke running function
ansys_run
#invoke output function
ansys_out
echo ""
echo "Job "$output" terminated with success!"
echo ""
exit 0
```

Appendix B

Script for moving files from Grid to Remote Servers

In this appendix it is shown the code used to transfer files from Grid to Remote Servers using FTP(s) or http(s) protocol. This script is invoked by a function of the Pydio plugin developed to browse the LFC and runs in background in order to allow the users to access other portal's functionalities. The files are retrieved from Grid and stored temporarily in portal SE then they are transferred to the server specified from the users. At the end of the transfer the script sends an email with the transfer report: which files have been transferred correctly, which not and the failure reason. In case the failures are under the 2% of the total files transferred number, the script automatically retries to transfer the files failed. There is also a mechanism to make the transfer dynamics: on the base of the file size we used different values of timeout. In this way we try to set the time necessary to retrieve the file but avoiding waste of time in case of failures.

```
<?php

function download_from_grid($user_id, $x509_user_proxy, $lfc_server,
$selectedFile_array, $selectedFile_size, $lcg_gfal_infosys, $vo_active,
$size1, $lfn_file, $file_name, $downloaded_file_name, $filename) {
$error='true';
$random2=rand(5, 15);
$filename_rand ="file:/opt/dm/ajp/data/personal/".$user_id."/".$random2."_".$file_name;
$downloaded_file_name_rand=substr($filename_rand, 5);
exec("sudo -u tomcat X509_USER_PROXY=$x509_user_proxy LFC_HOST=$lfc_server
/usr/bin/lcg-lr $lfn_file", $replica_values2);
shuffle($replica_values2);
foreach ($replica_values2 as $element2){
        if($error=="true"){
            exec("sudo -u tomcat X509_USER_PROXY=$x509_user_proxy LFC_HOST=$lfc_server
            LCG_GFAL_INFOSYS=$lcg_gfal_infosys  lcg-cp --sendreceive-timeout 4
            --vo $vo_active $element2 $filename_rand > /dev/null &");
            usleep(3800000);
            if (file_exists($downloaded_file_name_rand)){
            if ($size1<524288000) {
                 $srtimeout=300;
```

```
             } else if($size1>=524288000 && $size1<1610612736){
                  $srtimeout=600;
             } else if($size1>=1610612736 && $size1<16106127360){
                  $srtimeout=1200;
             } else {
                   $srtimeout=3600;
             }
             exec("sudo -u tomcat X509_USER_PROXY=$x509_user_proxy LFC_HOST=$lfc_server
             LCG_GFAL_INFOSYS=$lcg_gfal_infosys  /usr/bin/lcg-cp --sendreceive-timeout
             $srtimeout --connect-timeout 300 --bdii-timeout 300 --srm-timeout 300
             --vo $vo_active $element2  $filename ",
             $download_values);
             $size2=filesize($downloaded_file_name);
             if($size2==$size1){
                 $error='false';
             }
             unlink($downloaded_file_name_rand);
             } else {
                $error='true';
             }
        }
    }
return $error;
}

#VARIABLES DEFINITION
proc_nice(15);
$selectedFile_list=$argv[1];
$x509_user_proxy=$argv[2];
$lfc_server=$argv[3];
$home=$argv[4];
$protocol=$argv[5];
$server_comp=$argv[6];
$username=$argv[7];
$password=$argv[8];
$random=$argv[9];
$lcg_gfal_infosys=$argv[10];
$user_home_dir_big_transfer=$argv[11];
$numElements=$argv[12];
$mail=$argv[13];
$firstName=$argv[14];
$lastName=$argv[15];
$user_id=$argv[16];
$vo_active = $argv[17];
$pref = $argv[18];
if (substr($pref, 0, 1)!="/" && $pref!='') {
$pref = "/".$pref;
}
$port = $argv[19];
$mycloud = $argv[20];
$iesimo = $argv[21];
$dir = $argv[22];
$total_files_size_list = $argv[23];
$x=0;
$dir_selected= end(explode('/', substr($dir, 0, -1)));
if ($dir_selected=="") {
$dir_selected = "/";
} else {
$dir_selected = "/".$dir_selected."/";
}

    require_once('/opt/dm/ajp/plugins/access.lfc/phpmailer/class.phpmailer.php');
    require_once('/opt/dm/ajp/plugins/access.lfc/fpdf/fpdf.php');

#RETRIEVE FILES FROM GRID
$selectedFile_array=explode('***', $selectedFile_list);
$selectedFile_size=explode('***', $total_files_size_list);
foreach ($selectedFile_array as $selectedFile){
$user_home_dir_big_transfer_grid_error=$user_home_dir_big_transfer.'/grid_error';
$lfn_file="lfn:/".$home."/".$selectedFile;
$lfn_file2=$home."/".$selectedFile;
$selectedFile_pieces=split("/", $selectedFile);
$file_name=end($selectedFile_pieces);
$file_name = clean_file_name($file_name);
$filename="file:/opt/dm/ajp/data/personal/".$user_id."/".$file_name;
$replica_values2=array();
exec("sudo -u tomcat X509_USER_PROXY=$x509_user_proxy LFC_HOST=$lfc_server
/usr/bin/lcg-lr $lfn_file", $replica_values2);
shuffle($replica_values2);
$path=$user_home_dir_big_transfer;
$server=$server_comp;
$downloaded_file_name=substr($filename, 5);
$firstname=$firstName;
$lastname=$lastName;
$path_user = str_replace($random, '', $path);
$pieces_1 = explode("/", $downloaded_file_name);
$file=end($pieces_1);
$server_dest =$protocol."://".$server.":".$port;
$path_completo=$server_dest.$pref.$dir_selected.$file;
```

```
$path_mail=$server_dest.$pref;
$user_dir="/opt/dm/ajp/data/personal/".$user_id."/";
     $file_lock=$user_home_dir_big_transfer.'/lock';
$size1=$selectedFile_size[$x];
$error = download_from_grid($user_id, $x509_user_proxy, $lfc_server,
$selectedFile_array, $selectedFile_size, $lcg_gfal_infosys, $vo_active,
$size1, $lfn_file, $file_name, $downloaded_file_name, $filename);

#COPY FILES ON REMOTE SERVER
if(!file_exists($downloaded_file_name)||$error!='false'){
$fd = fopen($user_home_dir_big_transfer_grid_error, 'a');
fwrite($fd, $selectedFile_size[$x]."***".$lfn_file."***".$dir_selected.$file_name.PHP_EOL);
fclose($fd);
} else {
if($protocol=='ftp'|| $protocol=='http'){
$ch = curl_init();
$fp = fopen($downloaded_file_name, 'r');
curl_setopt($ch, CURLOPT_URL, $path_completo);
curl_setopt($ch, CURLOPT_USERPWD, "$username:$password");
curl_setopt($ch, CURLOPT_UPLOAD, 1);
curl_setopt($ch, CURLOPT_INFILE, $fp);
curl_setopt($ch, CURLOPT_INFILESIZE, filesize($downloaded_file_name));
curl_setopt($ch, CURLOPT_FAILONERROR, true);
curl_setopt($ch, CURLOPT_FTP_CREATE_MISSING_DIRS, true);
curl_exec ($ch);
$error_no = curl_errno($ch);
$error_string = curl_error($ch);
curl_close ($ch);
if ($error_no == 0) {
$fd = fopen($path."/ok", 'a');
fwrite($fd, $dir_selected.$file.PHP_EOL);
fclose($fd);
} else {
$fd = fopen($path."/error", 'a');
fwrite($fd, $dir_selected.$file.PHP_EOL);
fclose($fd);
}
} else {
$filepath=$pref.$dir_selected.$file;
$connection = ssh2_connect("$server", "$port");
ssh2_auth_password($connection, "$username", "$password");
$error_no = ssh2_scp_send($connection, "$downloaded_file_name", "$filepath", 0644);
if ($error_no) {
$fd = fopen($path."/ok", 'a');
fwrite($fd, $dir_selected.$file.PHP_EOL);
fclose($fd);
} else {
$fd = fopen($path."/error", 'a');
fwrite($fd, $dir_selected.$file.PHP_EOL);
fclose($fd);
}
}
}
unlink($downloaded_file_name);
if ($handle = opendir($user_dir)) {
    while (false !== ($entry = readdir($handle))) {
        if ($entry != "." && $entry != "..") {
        }
    }
    closedir($handle);
}
$user_home_dir_big_transfer_iesimo=$user_home_dir_big_transfer.'/file_'.$iesimo.'.zzz';
touch("$user_home_dir_big_transfer_iesimo");
$occ = count(glob($user_home_dir_big_transfer.'/*.zzz'));
$iesimo++;
$x++;
}
if($occ==($numElements)){
if(unlink($file_lock)){
$filePath = "$path/myPdf.pdf";
$pdf = new FPDF();
$pdf->AddPage();
$lines_ok = file($path."/ok", FILE_IGNORE_NEW_LINES);
$lines_grid = file($path."/grid_error", FILE_IGNORE_NEW_LINES);

// RETRY CODE START //
$num_ok=sizeof($lines_ok);
$num_grid_errors=sizeof($lines_grid);
if ($num_grid_errors>0 && round($num_ok/$num_grid_errors)>1) {
        $lines_grid_after_retry=array();
        foreach($lines_grid as $row){
$file_info=explode("***", $row);
$lfn_file=$file_info[1];
$size1=$file_info[0];
$error = download_from_grid($user_id, $x509_user_proxy, $lfc_server,
$selectedFile_array, $selectedFile_size, $lcg_gfal_infosys, $vo_active,
$size1, $lfn_file, $file_name, $downloaded_file_name, $filename);

if ($error=="true") {
        $lines_grid_after_retry[]=$file_info[2];
} else {
$lines_ok[]=$file_info[2];
}
        }
$lines_grid = $lines_grid_after_retry;

}
// RETRY CODE END //

// PREPARE MAIL //
sort($lines_ok);
$str_error_ok = "";
$ok_i=0;
foreach($lines_ok as $line_ok){
if ($ok_i==0) {
$pdf->SetTextColor(0,170,0);
$pdf->SetFont('Arial','B',14);
$pdf->Cell(40,10,'Files correctly copied!');
$pdf->Ln();
}
$pdf->SetTextColor(0,0,0);
$pdf->SetFont('Arial','B',11);
$pdf->Cell(40,10,$line_ok);
$pdf->Ln(6);
$str_error_ok.=$line_ok."<br />";
$ok_i++;
}
if ($str_error_ok=="") {
$display_ok="none";
} else {
$display_ok="block";
}
sort($lines_grid);
$str_error_grid="";
$grid_err_i=0;
foreach($lines_grid as $line_grid){
if (!isset($retry)) {
$line_grid=end(explode("***", $line_grid));
}
if ($grid_err_i==0) {
$pdf->Ln(16);
$pdf->SetTextColor(255,0,0);
$pdf->SetFont('Arial','B',14);
$pdf->Cell(40,10,'Files not copied for Grid error reasons!');
$pdf->Ln();
}
$pdf->SetTextColor(0,0,0);
$pdf->SetFont('Arial','B',11);
$pdf->Cell(40,10,$line_grid);
$pdf->Ln(6);
$str_error_grid.=$line_grid."<br />";
$grid_err_i++;
}
if ($str_error_grid=="") {
$display_grid="none";
} else {
$display_grid="block";
}
$lines_error = file($path."/error", FILE_IGNORE_NEW_LINES);
sort($lines_error);
$str_error_other="";
$other_err_i=0;
foreach($lines_error as $line_error){
if ($other_err_i==0) {
$pdf->Ln(16);
$pdf->SetTextColor(255,128,0);
$pdf->SetFont('Arial','B',14);
$pdf->Cell(40,10,'Files not copied for other error reasons!');
$pdf->Ln();
}
$pdf->SetTextColor(0,0,0);
$pdf->SetFont('Arial','B',11);
$pdf->Cell(40,10,$line_error);
$pdf->Ln(6);
$str_error_other.=$line_error."<br />";
$other_err_i++;
}
if ($str_error_other=="") {
$display_other="none";
} else {
$display_other="block";
}
$pdf->Output($filePath,'F');
$to = $mail;
$subject = 'Grid file transfer completed';
$from ='igi-portal@italiangrid.it';
$ok_i_perc=round(($ok_i/($numElements))*100, 1);
$grid_err_i_perc=round(($grid_err_i/($numElements))*100, 1);
```

```
$other_err_i_perc=round(($other_err_i/($numElements))*100, 1);
if($numElements<20){
$body =" ...  ";
} else {
$body =" ...  ";
}
$email = new PHPMailer();
$email->IsSMTP(); // telling the class to use SMTP
$email->Host       = "postino.cnaf.infn.it"; // SMTP server
$email->SMTPDebug  = 2;
$email->From      = $from;
$email->FromName = 'IGI Portal';
$email->Subject   = $subject;
$email->Body      = $body;
$email->IsHTML(true);
if($to=="marco.bencivenni@cnaf.infn.it"){
$to="marco.bencivenni@gmail.com";
}
$email->AddAddress($to);
if($numElements>=20){
$email->AddAttachment( $filePath , 'report.pdf' );
}
    if($email->Send()){
 deleteDir($path);
}  else {
$error_mail = $mail->ErrorInfo;
}
}
}

// FUNCTIONS //
function clean_file_name($filename) {
        $bad = array(
                    "<!--",  "-->", "'", "<", ">", '"', '&', '$', '=', ';',
                    '?', ' ', ':', "\n", "\r", "%20", "%22", "%3c", "%253c",
                    "%3e",  "%0e", "%28", "%29",  "%2528", "%26", "%24", "%3f",
                    "%3b", "%3d", "(", ")", );
    $filename = str_replace($bad, '', $filename);
    return stripslashes($filename);
    }

function deleteDir($dirPath) {
    if (! is_dir($dirPath)) {
        throw new InvalidArgumentException("$dirPath must be a directory");
    }
    if (substr($dirPath, strlen($dirPath) - 1, 1) != '/') {
        $dirPath .= '/';
    }
    $files = glob($dirPath . '*', GLOB_MARK);
    foreach ($files as $file) {
        if (is_dir($file)) {
            self::deleteDir($file);
        } else {
            unlink($file);
        }
    }
    rmdir($dirPath);
}

?>
```

Appendix C

DIRAC multi VO configuration

In this appendix, it is shown the DIRAC configuration to support multi VOs. The DIRAC Configuration is organized in a tree structure. It is divided in sections, which can also be seen as directories. Each section can contain other sections and options. The options are the leafs in the configuration tree, which contain the actual configuration data. At the top level of the Configuration tree there are the following sections:

- *DIRAC*: this section contains the most general information about the DIRAC installation.
- *Systems*: this section provides configuration data for all the DIRAC Systems, their instances and components - services, agents and databases.
- *Registry*: this section contains information about DIRAC users, groups and communities (VOs).
- *Resources*: this section provides description of all the DIRAC computing resources. This includes computing and storage elements as well as descriptions of several DIRAC and third party services.
- *Operations*: this section collects various operational parameters needed to run the system.

```
DIRAC
{
  Configuration
  {
    Name = MyConfiguration
    Version = 2014-01-24 08:45:17.754002
    Servers = dips://dirac.cnaf.infn.it:9135/Configuration/Server
    MasterServer = dips://dirac.cnaf.infn.it:9135/Configuration/Server
  }
  Setups
  {
    MyDIRAC-Production
    {
      Configuration = Production
      Framework = Production
      WorkloadManagement = Production
      Accounting = Production
      RequestManagement = Production
    }
  }
}
Registry
{
  Users
  {
    portal-igi
    {
      DN = /C=IT/O=INFN/OU=Host/L=CNAF/CN=portal.italiangrid.it
      Email = igi-portal-admin@lists.italiangrid.it
    }
    marco_bencivenni
    {
      DN = /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Marco Bencivenni
      Email = marco.bencivenni@cnaf.infn.it
    }
  }
  Groups
  {
    user
    {
      Users = apaolini
      Properties = NormalUser
    }
    dirac_admin
    {
      Users = flyback
      Users += portal-igi
      Users += marco_bencivenni
      Properties = AlarmsManagement
      Properties += ServiceAdministrator
      Properties += CSAdministrator
      Properties += JobAdministrator
      Properties += FullDelegation
      Properties += ProxyManagement
      Properties += Operator
    }
    gridit_user
    {
      Users += marco_bencivenni
      Properties = NormalUser
      VOMSRole = /gridit
      VOMSVO = gridit
      VO = gridit
      SubmitPool = Pool_gridit
      AutoAddVOMS = True
      AutoUploadProxy = True
      AutoUploadPilotProxy = True
    }
    infngrid_user
    {

      Properties = NormalUser
      VOMSRole = /infngrid
      VOMSVO = infngrid
      VO = infngrid
      SubmitPool = Pool_infngrid
      AutoAddVOMS = True
      AutoUploadProxy = True
      AutoUploadPilotProxy = True
    }
  }
  Hosts
  {
    dirac.cnaf.infn.it
    {
      DN = /C=IT/O=INFN/OU=Host/L=CNAF/CN=dirac.cnaf.infn.it
      Properties = TrustedHost
      Properties += CSAdministrator
      Properties += JobAdministrator
      Properties += FullDelegation
      Properties += ProxyManagement
      Properties += Operator
    }
  }
  VOMS
  {
    Mapping
    {
      user = /gridit
      gridit_user = /gridit
      infngrid_user = /infngrid
     }
    Servers
    {
      gridit
      {
        voms.cnaf.infn.it
        {
          DN = /C=IT/O=INFN/OU=Host/L=CNAF/CN=voms.cnaf.infn.it
          Port = 15008
          CA = /C=IT/O=INFN/CN=INFN CA
        }
        voms-01.pd.infn.it
        {
          DN = /C=IT/O=INFN/OU=Host/L=Padova/CN=voms-01.pd.infn.it
          CA = /C=IT/O=INFN/CN=INFN CA
          Port = 15008
        }
      }
      infngrid
      {
        voms.cnaf.infn.it
        {
          DN = /C=IT/O=INFN/OU=Host/L=CNAF/CN=voms.cnaf.infn.it
          Port = 15000
          CA = /C=IT/O=INFN/CN=INFN CA
        }
        voms-01.pd.infn.it
        {
          DN = /C=IT/O=INFN/OU=Host/L=Padova/CN=voms-01.pd.infn.it
          CA = /C=IT/O=INFN/CN=INFN CA
          Port = 15000
        }
      }
    }
  }
  VO
  {
    gridit
    {
      SubmitPools = Pool_gridit
      VOAdmin = dmichelotto
      VOMSName = gridit
      VOMSServers
      {
        voms.cnaf.infn.it
        {
          DN = /C=IT/O=INFN/OU=Host/L=CNAF/CN=voms.cnaf.infn.it
          CA = /C=IT/O=INFN/CN=INFN CA
          Port = 15008
        }
        voms-01.pd.infn.it
        {
          DN = /C=IT/O=INFN/OU=Host/L=Padova/CN=voms-01.pd.infn.it
          CA = /C=IT/O=INFN/CN=INFN CA
          Port = 15008
        }
      }
    }
    infngrid
    {
      SubmitPools = Pool_infngrid
      VOAdmin = dmichelotto
      VOMSName = infngrid
      VOMSServers
      {
        voms.cnaf.infn.it
        {
          DN = /C=IT/O=INFN/OU=Host/L=CNAF/CN=voms.cnaf.infn.it
          CA = /C=IT/O=INFN/CN=INFN CA
          Port = 15000
        }
        voms-01.pd.infn.it
        {
          DN = /C=IT/O=INFN/OU=Host/L=Padova/CN=voms-01.pd.infn.it
          CA = /C=IT/O=INFN/CN=INFN CA
          Port = 15000
        }
      }
    }
  }
  DefaultGroup = gridit_user
  DefaultProxyTime = 604800
}
```

```
Operations
{
  MyDIRAC-Production
  {
    Pilot
    {
      Version = v6r7p20
      Project = DIRAC
      CheckVersion = True
    }
  }
  gridit
  {
    Defaults
    {
      EMail
      {
        Production = diego.michelotto@cnaf.infn.it
        Logging = diego.michelotto@cnaf.infn.it
      }
      Shifter
      {
        ProductionManager
        {
          User = dmichelotto
          Group = user
        }
        SAMManager
        {
          User = dmichelotto
          Group = user
        }
      }
    }
    Production
    {
      Pilot
      {
        Version = v6r7p20
        Project = DIRAC
        CheckVersion = True
      }
    }
  }
  EMail
  {
    Production = diego.michelotto@cnaf.infn.it
    Logging = diego.michelotto@cnaf.infn.it
  }
  Gridit-Production
  {
    Shifter
    {
      SAMManager
      {
        User = dmichelotto
        Group = gridit_user
      }
      ProductionManager
      {
        User = dmichelotto
        Group = gridit_user
      }
      DataManager
      {
        User = dmichelotto
        Group = gridit_user
      }
    }
  }
  infngrid
  {
    Infngrid-Production
    {
      Shifter
      {
        ProductionManager
        {
          User = dmichelotto
          Group = user
        }
        SAMManager
        {
          User = dmichelotto
          Group = user
        }
        DataManager
        {
          User = dmichelotto
          Group = user
        }
      }
    }
  }
  JobDescription
  {
    AllowedJobTypes = MPI
    AllowedJobTypes += User
    AllowedJobTypes += Test
    SubmitPools = Pool_gridit
  }
}
Website
{
  DefaultGroups = visitor
  DefaultGroups += user
  DefaultGroups += dirac_admin
  DefaultSetup = MyDIRAC-Production
  Authorization
  {
    systems
    {
      configuration
      {
        Default = all
        showHistory = CSAdministrator
        commitConfiguration = CSAdministrator
        showCurrentDiff = CSAdministrator
        showDiff = CSAdministrator
        rollbackToVersion = CSAdministrator
        manageRemoteConfig = CSAdministrator
        manageRemoteConfig += ServiceAdministrator
      }
    }
  }
}
Systems
{
  Framework
  {
    Production
    {
      Services
      {
        SystemAdministrator
        {
          Port = 9162
          Authorization
          {
            Default = ServiceAdministrator
          }
        }
        SystemLoggingReport
        {
          Port = 9144
          Authorization
          {
            Default = authenticated
          }
        }
        Monitoring
        {
          Port = 9142
          Authorization
          {
            Default = authenticated
            queryField = ServiceAdministrator
            tryView = ServiceAdministrator
            saveView = ServiceAdministrator
            deleteView = ServiceAdministrator
            deleteActivity = ServiceAdministrator
            deleteActivities = ServiceAdministrator
            deleteViews = ServiceAdministrator
            FileTransfer
            {
              Default = authenticated
            }
          }
        }
        Notification
        {
          Port = 9154
          SMSSwitch = sms.switch.ch
          Authorization
          {
            Default = AlarmsManagement
            sendMail = authenticated
            sendSMS = authenticated
            removeNotificationsForUser = authenticated
            markNotificationsAsRead = authenticated
            getNotifications = authenticated
            ping = authenticated
          }
        }
        SecurityLogging
        {
          Port = 9153
          Authorization
          {
            Default = authenticated
          }
        }
        UserProfileManager
        {
          Port = 9155
          Authorization
          {
            Default = authenticated
          }
        }
        ProxyManager
        {
          Port = 9152
          MaxThreads = 100
          getVOMSProxyWithTokenMaxThreads = 2
          Authorization
          {
            Default = authenticated
            getProxy = FullDelegation
            getProxy += LimitedDelegation
            getProxy += PrivateLimitedDelegation
            getVOMSProxy = FullDelegation
            getVOMSProxy += LimitedDelegation
            getVOMSProxy += PrivateLimitedDelegation
            getProxyWithToken = FullDelegation
            getProxyWithToken += LimitedDelegation
            getProxyWithToken += PrivateLimitedDelegation
            getVOMSProxyWithToken = FullDelegation
            getVOMSProxyWithToken += LimitedDelegation
            getVOMSProxyWithToken += PrivateLimitedDelegation
            getLogContents = ProxyManagement
            setPersistency = ProxyManagement
          }
          UseMyProxy = True
        }
        SystemLogging
        {
          Port = 9141
          Authorization
          {
            Default = authenticated
          }
        }
        Plotting
        {
          Port = 9157
          PlotsLocation = data/plots
          Authorization
          {
            Default = authenticated
            FileTransfer
            {
              Default = authenticated
            }
          }
        }
        BundleDelivery
        {
          Port = 9158
          Authorization
          {
            Default = authenticated
            FileTransfer
            {
              Default = authenticated
            }
          }
        }
      }
      URLs
      {
        SystemAdministrator = dips://dirac.cnaf.infn.it:9162/Framework/SystemAdministrator
        SystemLoggingReport = dips://dirac.cnaf.infn.it:9144/Framework/SystemLoggingReport
        Monitoring = dips://dirac.cnaf.infn.it:9142/Framework/Monitoring
        Notification = dips://dirac.cnaf.infn.it:9154/Framework/Notification
        SecurityLogging = dips://dirac.cnaf.infn.it:9153/Framework/SecurityLogging
        UserProfileManager = dips://dirac.cnaf.infn.it:9155/Framework/UserProfileManager
```

```
      ProxyManager = dips://dirac.cnaf.infn.it:9152/Framework/ProxyManager
      SystemLogging = dips://dirac.cnaf.infn.it:9141/Framework/SystemLogging
      Plotting = dips://dirac.cnaf.infn.it:9157/Framework/Plotting
      BundleDelivery = dips://dirac.cnaf.infn.it:9158/Framework/BundleDelivery
    }
    Agents
    {
      SystemLoggingDBCleaner
      {
        RemoveDate = 30
      }
      CAUpdateAgent
      {
        PollingTime = 21600
      }
      TopErrorMessagesReporter
      {
        MailList =
        Reviewer =
        Threshold = 10
        QueryPeriod = 7
        NumberOfErrors = 10
      }
    }
    Databases
    {
      UserProfileDB
      {
        DBName = UserProfileDB
        Host = localhost
      }
      NotificationDB
      {
        DBName = NotificationDB
        Host = localhost
      }
      ComponentMonitoringDB
      {
        DBName = ComponentMonitoringDB
        Host = localhost
      }
      ProxyDB
      {
        DBName = ProxyDB
        Host = localhost
      }
      SystemLoggingDB
      {
        DBName = SystemLoggingDB
        Host = localhost
      }
    }
  }
}
WorkloadManagement
{
  Production
  {
    Services
    {
      SandboxStore
      {
        Port = 9196
        LocalSE = ProductionSandboxSE
        MaxThreads = 200
        toClientMaxThreads = 100
        Backend = local
        MaxSandboxSizeMiB = 10
        SandboxPrefix = Sandbox
        BasePath = /opt/dirac/storage/sandboxes
        DelayedExternalDeletion = True
        Authorization
        {
          Default = authenticated
          FileTransfer
          {
            Default = authenticated
          }
        }
      }
      Matcher
      {
        Port = 9170
        MaxThreads = 20
        #Flag for checking the DIRAC version of the pilot is the current production one as defined
        #in /Operations/<vo>/<setup>/Versions/PilotVersion option
        CheckPilotVersion = Yes
        #Flag to check the site job limits
        SiteJobLimits = False
```

```
          Authorization
          {
            Default = authenticated
            getActiveTaskQueues = JobAdministrator
          }
        }
        JobMonitoring
        {
          Port = 9130
          Authorization
          {
            Default = authenticated
          }
        }
        JobManager
        {
          Port = 9132
          MaxParametricJobs = 100
          Authorization
          {
            Default = authenticated
          }
        }
        JobStateUpdate
        {
          Port = 9136
          SSLSessionTime = 86400
          MaxThreads = 100
          Authorization
          {
            Default = authenticated
          }
        }
        WMSAdministrator
        {
          Port = 9145
          Authorization
          {
            Default = Operator
            getJobPilotOutput = authenticated
            setJobForPilot = authenticated
            setPilotBenchmark = authenticated
            setPilotStatus = authenticated
            getSiteMask = authenticated
            ping = authenticated
            getPilots = authenticated
            allowSite = authenticated
            banSite = authenticated
            getPilotSummary = authenticated
            getSiteMaskLogging = authenticated
            getPilotSummaryWeb = authenticated
            getPilotMonitorWeb = authenticated
            getSiteSummaryWeb = authenticated
            getSiteSummarySelectors = authenticated
            getPilotLoggingInfo = authenticated
            getPilotMonitorSelectors = authenticated
          }
        }
      }
      URLs
      {
        SandboxStore = dips://dirac.cnaf.infn.it:9196/WorkloadManagement/SandboxStore
        Matcher = dips://dirac.cnaf.infn.it:9170/WorkloadManagement/Matcher
        JobMonitoring = dips://dirac.cnaf.infn.it:9130/WorkloadManagement/JobMonitoring
        JobManager = dips://dirac.cnaf.infn.it:9132/WorkloadManagement/JobManager
        JobStateUpdate = dips://dirac.cnaf.infn.it:9136/WorkloadManagement/JobStateUpdate
        WMSAdministrator = dips://dirac.cnaf.infn.it:9145/WorkloadManagement/WMSAdministrator
      }
      Databases
      {
        SandboxMetadataDB
        {
          DBName = SandboxMetadataDB
          Host = localhost
        }
        JobDB
        {
          DBName = JobDB
          Host = localhost
        }
        JobLoggingDB
        {
          DBName = JobLoggingDB
          Host = localhost
        }
        TaskQueueDB
        {
          DBName = TaskQueueDB
          Host = localhost
        }
        PilotAgentsDB
        {
          DBName = PilotAgentsDB
          Host = localhost
        }
      }
      Agents
      {
        MightyOptimizer
        {
          FilteredOptimizers = InputData
          FilteredOptimizers += AncestorFiles
        }
        PilotStatusAgent
        {
          PollingTime = 300
          #Minimal Validity of the proxy stored in the Proxy Repository. If the validity
          #time is less that this value, the proxy will be renewed. The value is in seconds
          MinValidity = 1800
          #The period for which the proxy will be extended. The value is in hours
          ValidityPeriod = 15
          GridEnv = /etc/profile.d/grid-env
          #Flag enabling sending of the Pilot accounting info to the Accounting Service
          PilotAccountingEnabled = yes
        }
        JobHistoryAgent
        {
          PollingTime = 30
          UpdatePeriod = 300
        }
        SiteDirector
        {
          PollingTime = 120
          CETypes = CREAM
          Site =
          MaxJobsInFillMode = 5
          PilotDebugMode = True
          ExtraPilotOptions =
          GetPilotOutput = True
          UpdatePilotStatus = True
          SendPilotAccounting = True
          GenericPilotDN = /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Marco Bencivenni
          GenericPilotGroup = user
        }
        InputDataAgent
        {
          PollingTime = 120
        }
        TaskQueueDirector
        {
          SubmitPools = gLite
          SubmitPools += Pool_gridit
          SubmitPools += Pool_infngrid
          Status = Active
          DefaultSubmitPools = gLite
          AllowedSubmitPools = gLite
          AllowedSubmitPools += DIRAC
          AllowedSubmitPools += Pool_gridit
          AllowedSubmitPools += Pool_infngrid
          DIRACVersion = v6r7p20
          ListMatchDelay = 10
          extraPilotFraction = 1.0
          extraPilots = 2
          pilotsPerIteration = 100
          maxThreadsInPool = 8
          PollingTime = 30
          MaxCycles = 5000
          gLite
          {
            GridMiddleware = gLite
            GridEnv = /etc/profile.d/grid-env
            ResourceBrokers = wms-multi.grid.cnaf.infn.it
            Failing =
            PrivatePilotFraction = 1.0
            MaxJobsInFillMode = 5
            Rank = ( other.GlueCEStateWaitingJobs == 0 ? ( other.GlueCEStateFreeCPUs * 10
            / other.GlueCEInfoTotalCPUs + other.GlueCEInfoTotalCPUs / 500 ) :
            -other.GlueCEStateWaitingJobs * 4 / (other.GlueCEStateRunningJobs + 1 ) - 1 )
            GenericPilotDN = /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Diego Michelotto
            GenericPilotGroup = dirac_pilot
          }
          Pool_gridit
          {
            GridMiddleware = gLite
            ResourceBrokers = wms-multi.grid.cnaf.infn.it
            Failing =
            PrivatePilotFraction = 1.0
            MaxJobsInFillMode = 5
            Rank = ( other.GlueCEStateWaitingJobs == 0 ? ( other.GlueCEStateFreeCPUs * 10
            / other.GlueCEInfoTotalCPUs + other.GlueCEInfoTotalCPUs / 500 ) :
            -other.GlueCEStateWaitingJobs * 4 / (other.GlueCEStateRunningJobs + 1 ) - 1 )
            GenericPilotDN = /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Diego Michelotto
            GenericPilotGroup = gridit_user
            GridEnv = /etc/profile.d/grid-env
            VirtualOrganization = gridit
            MyProxyServer = myproxy.cnaf.infn.it
          }
          Pool_infngrid
          {
            GridMiddleware = gLite
            ResourceBrokers = wms-multi.grid.cnaf.infn.it
            Failing =
            PrivatePilotFraction = 1.0
            MaxJobsInFillMode = 5
            Rank = ( other.GlueCEStateWaitingJobs == 0 ? ( other.GlueCEStateFreeCPUs * 10
            / other.GlueCEInfoTotalCPUs + other.GlueCEInfoTotalCPUs / 500 ) :
            -other.GlueCEStateWaitingJobs * 4 / (other.GlueCEStateRunningJobs + 1 ) - 1 )
            GenericPilotDN = /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Diego Michelotto
            GenericPilotGroup = infngrid_user
            GridEnv = /etc/profile.d/grid-env
            VirtualOrganization = infngrid
          }
          DIRAC
          {
            GridMiddleware = DIRAC
          }
        }
        JobCleaningAgent
        {
          PollingTime = 120
        }
        StalledJobAgent
        {
          StalledTimeHours = 2
          FailedTimeHours = 6
          PollingTime = 120
        }
      }
    }
  }
  RequestManagement
  {
    Production
    {
      Services
      {
        RequestManager
        {
          Port = 9143
          Backend = file
          Path = requestDB
          Authorization
          {
            Default = authenticated
          }
        }
      }
      URLs
      {
        RequestManager = dips://dirac.cnaf.infn.it:9143/RequestManagement/RequestManager
      }
    }
  }
  Accounting
  {
    Production
    {
      Services
      {
        DataStore
        {
          Port = 9133
          Authorization
          {
            Default = authenticated
            compactDB = ServiceAdministrator
            deleteType = ServiceAdministrator
            registerType = ServiceAdministrator
            setBucketsLength = ServiceAdministrator
            regenerateBuckets = ServiceAdministrator
          }
        }
        ReportGenerator
        {
          Port = 9134
          Authorization
          {
            Default = authenticated
            FileTransfer
            {
              Default = authenticated
            }
          }
        }
      }
      URLs
      {
        DataStore = dips://dirac.cnaf.infn.it:9133/Accounting/DataStore
        ReportGenerator = dips://dirac.cnaf.infn.it:9134/Accounting/ReportGenerator
      }
      Databases
      {
        AccountingDB
        {
          DBName = AccountingDB
          Host = localhost
        }
      }
    }
  }
  Configuration
  {
    Production
    {
      Agents
      {
        CE2CSAgent
        {
          BannedCSs =
          MailTo =
          MailFrom =
          VirtualOrganization = gridit
        }
        CE2CSAgent_infngrid
        {
          Module = CE2CSAgent
          VirtualOrganization = infngrid
        }
      }
    }
  }
  Databases
  {
    User = Dirac
    Password = 8TuX3862Na
  }
}
Resources
{
  Sites
  {
    LCG
    {
      LCG.IGI-BOLOGNA-SL6.it
      {
        Name = IGI-BOLOGNA
        CE = cream-02.cnaf.infn.it
        CEs
        {
          cream-02.cnaf.infn.it
          {
            wnTmpDir = /tmp
            architecture = x86_64
            CEType = CREAM
            Queues
            {
              cream-pbs-test-sl6
              {
                maxCPUTime = 2880
                SI00 = 1039
              }
              cream-pbs-wnodes-sl6
              {
                maxCPUTime = 2880
                SI00 = 1039
              }
            }
            OS = ScientificSL_Carbon_6.2
            SI00 = 1039
          }
        }
        Coordinates = 11.3417:44.4948
        Mail = grid-operations@lists.cnaf.infn.it
      }
    }
  }
```

```
StorageElements
{
  ProductionSandboxSE
  {
    BackendType = DISET
    AccessProtocol.1
    {
      Host = dirac.cnaf.infn.it
      Port = 9196
      Protocol = dips
      ProtocolName = DIP
      Path = /WorkloadManagement/SandboxStore
      Access = remote
    }
```

Appendix D

Portal logical architecture

In this appendix it is described the detailed portal logical architecture. The Fig. D.1 shows the logical layers and the components, exhaustively described in the thesis, involved:

- *External AuthZ/AuthN Services*: in this layer there rae all the exyernal services that the portal uses for the user authentication and authorization purposes.
- *Portal AuthN*: here we can find the portlel developed for the portal authentication.
- *Portal AuthN*: here we can find the portlel developed for the portal authorization.
- *Portal Services*: in this layers there are all the portlets developed to provide the fucntionalities of workload computing, data management and cloud resources provisioning.
- *External Data/Computing Services*: in this layer we can find the external services used to interface the portal with the middleware and physiscal resources.
- *Middleware/Resources*: this layer rapresents the services provided by the middleware and the physiscal resources.

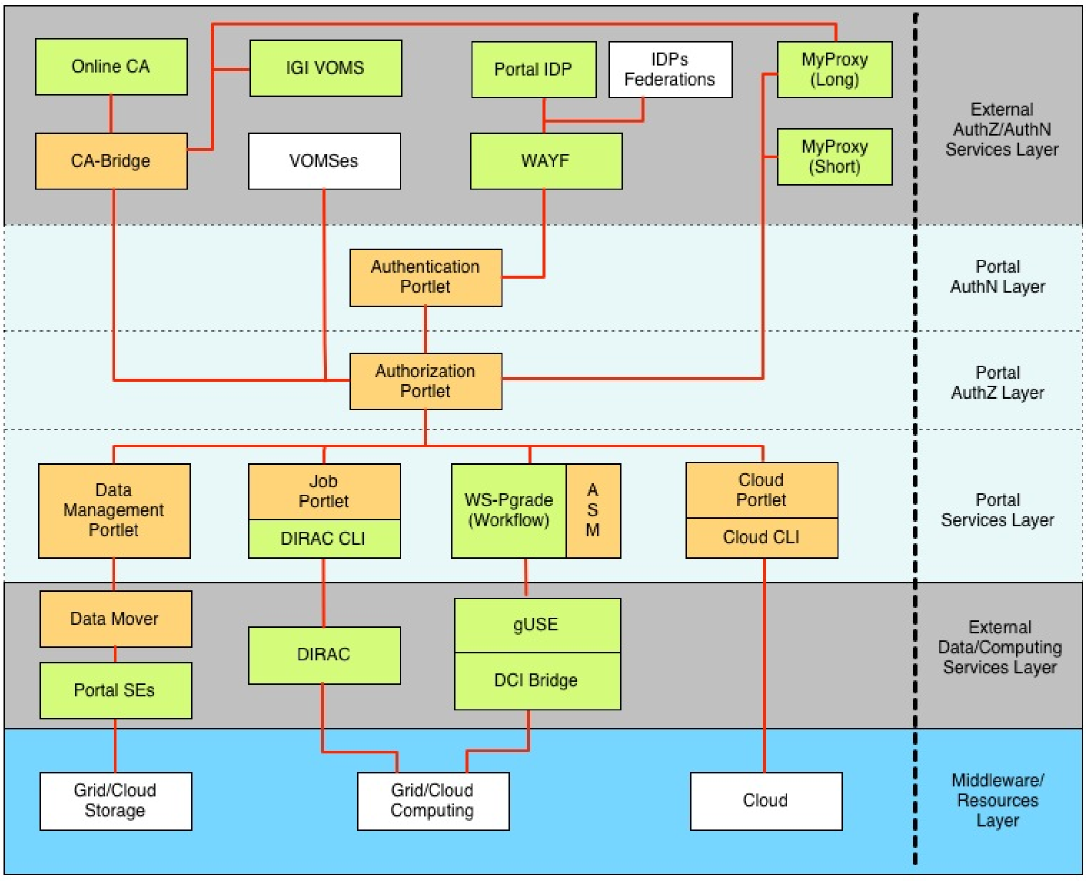


FIGURE D.1: Portal logical architecture

Abbreviations

ASM	**A**pplication **S**pecific **M**odule
AC	**A**ttribute **C**ertificate
AAI	**A**uthentication and **A**uthorization **I**nfrastructure
CAS	**C**entral **A**uthentication **S**ervice
CSR	**C**ertificate **S**igning **R**equest
CA	**C**ertification **A**uthority
CLI	**C**ommand **L**ine **I**nterface
CE	**C**omputing **E**lement
CREAM	**C**omputing **R**esource **E**xecution **a**nd **M**anagement
DAG	**D**irected **A**cyclic **G**raphs
DPM	**D**isk **P**ool **M**anager
DN	**D**istinguished **N**ame
DCI	**D**istributed **C**omputing **I**nfrastructures
DIRAC	**D**istributed **I**nfrastructure with **R**emote **A**gent **C**ontrol
EGEE	**E**nabling **G**rids for **E**-scienc**E**
EMI	**E**uropean **M**iddleware **I**nitiative
FTS	**F**ile **T**ransfer **S**ervice
FEM	**F**inite **E**lement **M**ethod
GPFS	**G**eneral **P**arallel **F**ile **S**ystem
GUID	**G**lobal **U**nique **I**dentifier
gUSE	**g**rid and cloud **U**ser **S**upport **E**nvironment
GSI	**G**rid **S**ecurity **I**nfrastructure
IdP	**Id**entity **P**rovider
IS	**I**nformation **S**ystem

IaaS	**I**nfrastructure **as a** **S**ervice
IGTF	**I**nternational **G**rid **T**rust **F**orum
IGI	**I**talian **G**rid **I**nitiative
JDL	**J**ob **D**efinition **L**anguage
LPDS	**L**aboratory of **P**arallel and **D**istributed **S**ystems
LCG	**L**HC **C**omputing **G**rid
LHC	**L**arge **H**adron **C**ollider
LFC	**L**CG **F**ile **C**atalogue
LGPL	**L**esser **G**eneral **P**ublic **L**icense
LoA	**L**evel **of** **A**ssurance
LSF	**L**oad **S**haring **F**acility
LRMS	**L**ocal **R**esource **M**anagement **S**ystem
LB	**L**ogging and **B**ookkeeping
LFN	**L**ogical **F**ile **N**ame
MICS	**M**ember **I**ntegrated **C**redential **S**ervices
OCCI	**O**pen **C**loud **C**omputing **I**nterface
PFN	**P**hysical **F**ile **N**ame
PaaS	**P**latform **as a** **S**ervice
PMAs	**P**olicy **M**anagement **A**uthorities
PBS	**P**ortable **B**atch **S**ystem
RA	**R**egistration **A**uthority
SAML	**S**ecurity **A**ssertion **M**arkup **L**anguage
SPES	**S**elective **P**roduction of **E**xotic **S**pecies
SP	**S**ervice **P**rovider
SLCS	**S**hort **L**iving **C**redential **S**ervice
SSO	**S**ingle **S**ign **O**n
SURL	**S**ite **URL**
SaaS	**S**oftware **as a** **S**ervice
SAN	**S**torage **A**rea **N**etwork
SE	**S**torage **E**lement
SRM	**S**torage **R**esource **M**anager

StoRM	**S**torage **R**esource **M**anager
SGE	**S**un **G**rid **E**ngine
TOTP	**T**ime-based **O**ne **T**ime **P**assword
TURL	**T**ransport **URL**
UI	**U**ser **I**nterface
VM	**V**irtual **M**achine
VOMS	**V**irtual **O**rganization **M**anagement **S**ervice
VO	**V**irtual **O**rganization
WN	**W**orker **N**ode
WNoDeS	**W**orker **N**odes **o**n **De**mand **S**ervice
WMS	**W**orkload **M**anagement **S**ystem

Bibliography

[1] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk: "RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile". May 2008, (http://tools.ietf.org/html/rfc5280)

[2] "EUGridPMA Guideline on Approved Robots" (http://www.euGridpma.org/guidelines/robot/approved-robots-20120912.pdf)

[3] Peter Kacsuk, Zoltan Farkas, Miklos Kozlovszky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai and Istvan Marton: "WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities". Journal of Grid Computing, Vol. 9, No. 4, pp 479-499, 2012

[4] "EGI-InSPIRE VO Portal Policy document" (https://documents.egi.eu/public/ShowDocument?docid=80)

[5] "EGI-InSPIRE VO Operations Policy" (https://documents.egi.eu/public/ShowDocument?docid=77)

[6] "EGI-InSPIRE, Grid Security Traceability and Logging Policy document" (https://documents.egi.eu/document/81)

[7] Riccardo Murri, Peter Z. Kunszt, Sergio Maffioletti, Valery Tschopp: "GridCertLib: A Single Sign-on Solution for Grid Web Applications and Portals". Journal of Grid Computing, December 2011, Volume 9, Issue 4, pp 441-453

[8] "JSR 168. Portlet Specification". Java Community Process, 2005 (http://www.jcp.org/en/jsr/detail?id=168)

[9] "JSR 286: Portlet Specification 2.0". Java Community Process, 2008 (http://www.jcp.org/en/jsr/detail?id=286)

[10] J. Basney, M. Humphrey, and V. Welch.: "The MyProxy Online Credential Repository". Software: Practice and Experience, Volume 35, Issue 9, July 2005, pages 801-816

[11] Alfieri, R., Cecchini, R., Ciaschini, V., dell'Agnello, L., Frohner, A., Lorentey, K., Spataro F.: "From Gridmap-file to VOMS: managing authorization in a Grid environment". Future Generation Computer Systems Vol. 21, Issue 4, 549558 (2005).

[12] Steven Tuecke, Von Welch, Doug Engert, Laura Pearlman, Mary Thompson: "RFC 3820: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile,June 2004". (http://tools.ietf.org/html/rfc3820)

[13] "TAGPMA, Profile for Member Integrated X.509 Credential Services with Secured Infrastructure". (http://www.euGridpma.org/guidelines/MICS/IGTF-AP-MICS-1.2-clean.pdf)

[14] David M'Raihi, Salah Machani, Mingliang Pei, Johan Rydell: "RFC 6238: TOTP: Time-Based One-Time Password Algorithm". May 2011, (http://tools.ietf.org/html/rfc6238)

[15] Tsaregorodtsev, A; Bargiotti, M; Brook, N; Ramo, A C; Castellani, G; Charpentier, P; Cioffi, C; Closier, J; Graciani, R; Kuznetsov, G; Li, Y Y; Nandakumar, R; Paterson, S; Santinelli, R; Smith, A C; Miguelez, M S; Gomez, S: "DIRAC: a community Grid solution". Journal of Physics: Conference Series 119, 062048 (2008)

[16] Stuart K. Paterson, Andrei Tsaregorodtsev, "DIRAC Optimized Workload Management", Journal of Physics: Conference Series 119 (2008) 062040

[17] I. Sfiligoi: "glideinWMS a generic pilot-based workload management system". Journal of Physics: Conference Series Volume 119 Part 6 2008 J. Phys.: Conf. Ser. 119 062044 doi:10.1088/1742-6596/119/6/062044

[18] Alfieri, R.; Bencivenni, M.; Boccia, V.; Buzzi, A.; Cesini, D.; Costantini, A.; De Pietri, R.; Gaido, L.; Giorgio, E.; La Rocca, G.; Malguzzi, P.; Mastrangelo, D.; Ottani, S.; Malguzzi, P.; Mastrangelo, D.; Venturini, D.: "Porting workflows based on small and medium parallelism applications to the Italian Grid Infrastructure". ISGC2013, in press on PoS

[19] Peter Kacsuk, Zoltan Farkas, Miklos Kozlovszky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai, Istvan Marton: "WS-PGRADE/gUSE Generic DCI Gateway Framework for a Large Variety of User Communities". Journal of Grid Computing Volume 10 Issue 4, December 2012 Pages 601-630

[20] P. Andreetto, S. Andreozzi, G. Avellino, S. Beco, A. Cavallini, M. Cecchi, V. Ciaschini, A. Dorise, F. Giacomini, A. Gianelle, U. Grandinetti, A. Guarise, A. Krop, R. Lops, A. Maraschini, V. Martelli, M. Marzolla, M. Mezzadri, E. Molinari, S. Monforte, F. Pacini, M. Pappalardo, A. Parrini, G. Patania, L. Petronzio, R. Piro, M. Porciani, F. Prelz, D. Rebatto, E. Ronchieri, M. Sgaravatto, V. Venturi and L. Zangrando: "The gLite workload management system". J. Phys.: Conf. Ser. Volume 119, issue 6 (2007)

[21] Lana Abadie, Paolo Badino, Jean-Philippe Baud, Ezio Corso, Matt Crawford, Shaun De Witt, Flavia Donno, Alberto Forti, Akos Frohner, Patrick Fuhrmann, Gilbert Grosdidier, Junmin Gu, Jens Jensen, Birger Koblitz, Sophie Lemaitre, Maarten Litmaath, Dmitry Litvinsev, Giuseppe Lo Presti, Luca Magnoni, Tigran Mkrtchan, Alexander Moibenko, Remi Mollon, Vijaya Natarajan, Gene Oleynik, Timur Perelmutov, Don Petravick, Arie Shoshani, Alex Sim, David Smith, Massimo Sponza, Paolo Tedesco, Riccardo Zappi: "Storage Resource Managers: Recent International Experience on Requirements and Multiple Co-Operating Implementations". Mass Storage Systems and Technologies, IEEE/NASA Goddard Conference on, pp. 47-59, 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007), 2007

[22] Bencivenni, M.; Brunetti, R.; Caltroni, A.; Ceccanti A.; Cesini, D.; Di Benedetto, M.; Fattibene, E.; Gaido, L.; Michelotto, D.; Misurelli, G.; Venturi,

V.; Veronesi, P.; Zappi, R.: ”A Web-based utility for Grid data management”. ISGC2013, in press on PoS.

[23] Magnoni, L.; Zappi, R.; Ghiselli, A. ”StoRM: a Flexible Solution for Storage Resource Manager in Grid”. Proceedings of the IEEE 2008 Nuclear Science Symposium (NSS-MIC 2008), 19 - 25 October 2008, Dresden, Germany. IEEE Computer Society (2008)

[24] Elisabetta Ronchieri, Marco Verlato, Davide Salomoni, Gianni Dalla Torre, Alessandro Italiano, Vincenzo Ciaschini, Daniele Andreotti, Stefano Dal Pra, Wouter Geert Touw, Gert Vriend, Geerten W. Vuister: ”Accessing Scientific Applications through the WNoDeS Cloud Virtualization Framework”. Proceedings of ISCG 2013, 17-22 March, Taipei, 2013

[25] Cristina Aiftimiei, Andrea Ceccanti, Danilo Dongiovanni, Alberto Di Meglio, Francesco Giacomini: ”Improving the quality of EMI Releases by leveraging the EMI Testing Infrastructure”. 2012 Journal of Physics: Conference Series Volume 396 Part 5.

[26] Mohammed Airaj, Christophe Blanchet, Stuart Kenny, Charles Loomis, StratusLab Collaboration(s): ”Appliance Management for Federated Cloud Environments”. CloudCom 2013 - 5th IEEE International Conference on Cloud Computing Technology and Science, Bristol, United Kingdom (2013)

[27] Wassenaar et al., ”WeNMR: Structural Biology on the Grid”. Journal of Grid Computing, 10:743-767 (2012)

[28] Ferrari, T.; Gaido, L.; ”Resources and Services of the EGEE Production Infrastructure”. Journal of Grid Computing, Springer Netherlands, pag. 119-133, vol. 9, Issue 2, June 2011, ISSN: 1570-7873, Doi: 10.1007/s10723-011-9184-1

[29] Costantini A., Michelotto D., Bencivenni M., Cesini D., Veronesi P., Giorgio E., Gaido L., Laganà A., Monetti A., Manzolaro M., Andrighetto A.: ”Implementation of the ANSYS Commercial Suite on the EGI Grid Platform”. Lecture Notes in Computer Science, Volume 7971, 84-95 (2013)

[30] Alfieri, R., Arezzini, S., Ciampa, A., De Pietri, R., Mazzoni, E.: "HPC on the Grid: The Theophys experience". Journal of Grid Computing, 11:265-260, DOI: 10.1007/s10723-012-9223-6 (2013)

[31] Löffler, F., Faber, J. Bentivegna, E., Bode, T., Diener, P., Haas, R., Hinder, I., Mundim, B., Ott, C., Schnetter, E., Allen, G., Campanelli, M., and Laguna, P.: "The Einstein Toolkit: A Community Computational Infrastructure for Relativistic Astrophysics". Classical and Quantum Gravity, 29(11):115001 (2012).

[32] Di Renzo, F, L. Scorzato, L., and C. Torrero, C.: "High loop renormalization constants by NSPT: A Status report". PoS, LAT2007:240, 2007

[33] Kunszt, Peter, et al. "The gLite File Transfer Service." 1st EGEE User Forum (March 2006)

[34] V. Hazlewood and M. Woitaszek, "Securing Science Gateways," TeraGrid Conference, July 2011, Salt Lake City, Utah, USA.

[35] Víctor Méndez Muñoz, Víctor Fernández Albor, Ricardo Graciani Diaz, Adriàn Casajús Ramo, Tomás Fernández Pena4, Gonzalo Merino Arévalo and Juan José Saborido Silva, "The Integration of CloudStack and OCCI/OpenNebula with DIRAC", Journal of Physics: Conference Series Volume 396 Part 3 2012 J. Phys.: Conf. Ser. 396 032075