Università degli Studi di Ferrara

Dottorato di ricerca in Matematica e Informatica

Ciclo XXIII

Coordinatore: Prof. Luisa Zanghirati

# Grid accounting for computing and storage resources towards standardization

Settore Scientifico Disciplinare INF/01

Dottorando:
**Dott. Cristofori Andrea**

Tutore:
**Prof. Luppi Eleonora**

Anni 2008/2010

Università degli Studi di Ferrara

Dottorato di ricerca in Matematica e Informatica

Ciclo XXIII

Coordinatore: Prof. Luisa Zanghirati

# Grid accounting for computing and storage resources towards standardization

Settore Scientifico Disciplinare INF/01

Dottorando:
**Dott. Cristofori Andrea**

Tutore:
**Prof. Luppi Eleonora**

Anni 2008/2010

# Contents

# Introduction

In the last years, we have seen a growing interest of the scientific community first and commercial vendors then, in new technologies like Grid and Cloud computing. The first in particular, was born to meet the enormous computational requests mostly coming from physic experiments, especially Large Hadron Collider's (LHC) experiments at Conseil Européen pour la Recherche Nucléaire (European Laboratory for Particle Physics) (CERN) in Geneva. Other scientific disciplines that are also benefiting from those technologies are biology, astronomy, earth sciences, life sciences, etc. Grid systems allow the sharing of heterogeneous computational and storage resources between different geographically distributed institutes, agencies or universities. For this purpose technologies have been developed to allow communication, authentication, storing and processing of the required software and scientific data. This allows different scientific communities the access to computational resources that a single institute could not host for logistical and cost reasons. Grid systems were not the only answer to this growing need of resources of different communities. At the same time, in the last years, we have seen the affirmation of the so called Cloud Computing. Cloud computing is a model for enabling convenient, 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[1]. The use of both computational paradigms and the utilization of storage resources, leverage on different authentication and authorization tools. The utilization of those technologies requires systems for the accounting of the consumed resources. Those systems are built on the top of the infrastructure and they collect all the needed data related to the users, groups and resources utilized. This information is then collected in central repositories where they can be analyzed and aggregated. Open Grid Forum (OGF) is the international organism that works to develop standards in the Grid environment. Usage Record - Working Group (UR-WG) is a group, born within OGF aiming at standardizing the Usage Record (UR) structure and publication for different kinds of resources. Up to now, the emphasis has been on the accounting for computational resources. With time it came out the need to expand those concepts to other aspects and especially to a definition and implementation of a standard UR for storage accounting. Several extensions to the UR definition are proposed in this thesis and the proposed developments in this field are described. The Distributed Grid Accounting System (DGAS) has been chosen, among other tools available, as the accounting system for the Italian Grid and is also adopted in other countries such as Greece and Germany. Together with HLRmon, it offers a complete accounting system and it is the tool that has been used during the writing of the thesis at INFN-CNAF.

In *Chapter 1*, I will focus the attention on the paradigm of distributed computing and the Grid infrastructure will be introduced with particular emphasis on the gLite middleware and the EGI-InSPIRE project. Following this description, *Chapter 2*, will discuss some Grid accounting systems for computational resources with particular stress for DGAS. In *Chapter 3*, the cross-check monitoring system used to check the correctness of the gathered data at the INFN-CNAF's Tier1 is presented. The BASH scripts and SQL code that are part of this tool are available on *Appendix A*. Another important aspect on accounting is the accounting for storage resources that will be discussed in *Chapter 4*. In the same chapter some extensions to the current UR definition will be proposed. Using the definitions introduced in *Chapter 4*, *Chapter 5* will describe a system developed during the last part of the thesis that implements those UR extensions. *Appendix B* will collect the scripts used for this purpose. Another important aspect is the need for the standardization of the communication protocol of the storage accounting tools. Most of those tools on Grid are adopting ActiveMQ for the exchange of URs. *Chapter 6* describes the implementation of this new messaging system in DGAS and the tests that have been made to evaluate the new functionality. *Appendix C* collects the scripts and configurations used during those tests. The last appendix, *Appendix D*, collects the publications in which a contribution has been given and where some of the developments and work discussed on the thesis are shown.

# Chapter 1

# The Grid infrastructure

*"If you think your users are idiots, only idiots will use it."*
Linus Torvalds

## 1.1 Challenges of modern researches and the Grid

Since the beginning of science and the scientific method, researchers needed tools to make predictions and to verify the results of their experiments. Before the invention of electronic computers, most calculation, where made manually with the help of tables or, more recently, with analog calculators. The importance of those tools is evident as they allow an increasing number of calculations. Researches in physics, and the related experiments, require huge amount of computational power. For this reason, since the invention of computers, they have been massively used to process the simulations and the data coming from the experiments. Some examples are those underway at CERN. A single institute, agency or university usually do not have the money, the manpower and structures to host the necessary resources needed. The idea was then to use the available and heterogeneous resources spread all around the world to share the computational power, the storage and the network interconnection through common interfaces and standards creating a Grid that could optimize their utilization and achieve results that could not have been reached using classical methods. A common example is the comparison with the electrical grid where users and providers might be located very far apart from each other. The user just requires to connect an electronic apparatus to the plug and the electricity, provided that the bill has been paid, will flow and the apparatus will work. In a similar way, in the Grid, the user requires certain resources and, if he meets the required conditions, his software will run in some facility connected to the Grid or e.g. his file will be stored somewhere. In the Grid, the access is not direct to other sites but it requires some sort of authentication and authorization so that the site can decide which users, belonging to certain groups, can gain access to their resources. To solve all those problems, and give the researchers the tools to exploit the distributed resources, several projects were born. Some of the main aims is to provide the middleware requested to create an infrastructure suitable for the operations described and at the same time provide the necessary support for the communities. Probably, the biggest experience of this kind in Europe has been the Enabling Grids for E-sciencE (EGEE) project which officially ended his third phase on April 30, 2010. It started back in 2004 taking on from European DataGrid (EDG) (2002-2004), and helped the adoption and the spread of knowledge on the Grid through different field of research. At the end of the project, it 'represented a world-wide infrastructure of approximately 250.000 CPU cores, collaboratively hosted by more than 300 centers around the world with around 13 million jobs executed on the EGEE Grid each month'[2]. At the end of this project, most of the experience and the knowledge learned during its life span will be carried on by the European Grid Initiative - Integrated Sustainable Pan-European Infrastructure for Researchers in Europe (EGI-InSPIRE) project which will coordinate the European National Grid Initiatives (NGIs) and allow a sustainable Grid infrastructure in Europe. Thanks to those projects, the communities that are utilizing the Grid have grown constantly. The initial users have grown and are now accounting, other than physics, groups in biology, earth sciences, climatology and many more; making it a rather flexible tool that can be used to tackle many challenges of today research. The next step is to allow the use of those technology to the industry and not only for the research. To reach this fundamental goal, the Grid should became even more reliable and provide all the tools needed for resources accounting.

## 1.2 Grid Computing

The Grid, is composed of several components: hardware, software and humans. Before going in deeper detail to explain some of those components and the interaction between them, we will give some definition given by different authors:

- "the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and a vast array of other computing resources over the Internet. A grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across 'multiple' administrative domains based on their (resources) availability, capacity, performance, cost and users' quality-of-service requirements"[3].
- "Computing resources are not administered centrally, open standards are used and non-trivial quality of service is achieved"[4]. According to Ian Foster's lists of primary attributes for a Grid.
- "grid computing is a service for sharing computer power and data storage capacity over the Internet"[5]

As seen from those definition we can say that a Grid infrastructure is a mixture of the three element mentioned. They can inter operate adopting common standards and agreements between the different groups and researchers which are in different places and countries. One of those project is EGEE with all the three phases if went trough and now EGI-InSPIRE. Similar projects have been started in Europe and in the rest of the world. Following are some examples:

- Enabling Desktop Grids for e-Science (EDGeS): is a European project with the aim of creating an integrated Grid infrastructure that seamlessly integrates a variety of Desktop Grids with EGEE type of service Grids[6].
- Uniform Interface to Computing Resources (UNICORE): offers a ready-to-run Grid system including client and server software. UNICORE makes distributed computing and data resources available in a seamless and secure way in intranets and the internet[7].

## 1.3 Grid architecture

Grid components, hardware and software, can be categorized in four layers. Starting from the lower, where we can find the hardware necessary for the infrastructure, up to the top layer that provides basic services and user applications. Figure 1.1 shows this classification which components are:

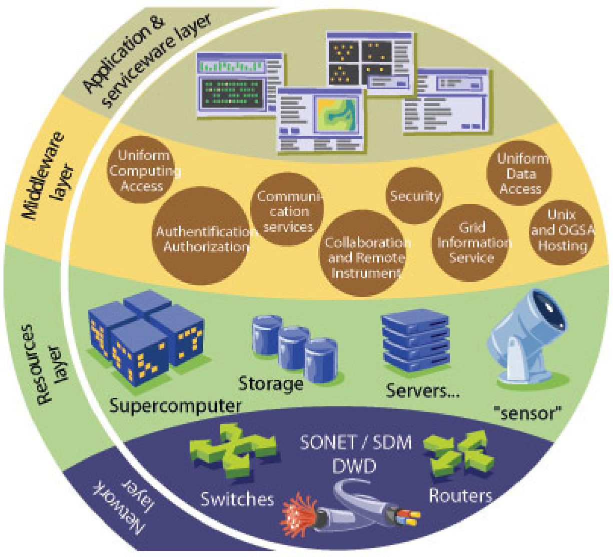


Figure 1.1: Layers classification on Grid

- **Network layer**: this layer ensure the connectivity among all the Grid components. It includes all the components that are useful to this purpose like cables, routers, switches, etc.
- **Resource layer**: it includes all the hardware and software, like computers, storage systems, operating systems, all kind of sensors that can be connected together to collect data. Among them we can find detectors for a physic experiment, sensors to register earthquakes or parameters of the air such, for example, humidity, temperature, images recorded by a telescope or any medical imagining instrument, etc. All the hardware that can record any kind of information useful to further study a certain aspect of science, medicine, economic or other can be classified in this layer.
- **Middleware layer**: in this part of the stack we can classify the software that gives the access to the elements in the layer below. The software developed within many of the previously mentioned projects can be classified here. It

may include, services for authentication, authorization, job submission, job execution, file retrieval, etc. It provides the necessary tools, for the *Application layer* to exploit the lower levels.

- **Application layer**: this is the top layer which includes applications written by scientists, researchers and other users of the Grid. Those applications can be written specifically for the use into the Grid or can be an adaptation of previously written software. In this part of the stack we can also classify web portals that give access to information on the utilized resources, accounting information and, more in general, all the information that can be of some interest for the user.

As we have seen both the *Middleware layer* and the *Application layer* provide software. The main difference between them is that the latter is accessed directly by the final users while the former should be invisible to them. The *Middleware layer* is fundamental for the upper layer as it provides all the necessary services.

## 1.4 How to join the Grid

Providing the hardware layer and software is the first necessary step necessary to join a Grid infrastructure. After that, a series of other requirements must be met:

- **No centralized control on the resources**: users should be able to access heterogeneous resources, from different institutes, universities, countries, etc. The complexity and the details of the infrastructure should be transparent to the user. The goal is to have the user request some type of resource, the job requested is sent to this resource independently on his location and the user pays proportionally to the usage.

- **Standards protocols and interfaces are used**: standard protocols and interfaces are necessary to communicates between different systems that might be installed in the same institute or in different places, countries, etc. A user should be able to use all the possible resources using a standard interface and common protocols. Without those standard the result is to have separate Grids that can not intercommunicate with each other. The aim of projects such as

Standards and Interoperability for eInfrastructure Implementation Initiative (SIENA) or the OGF forum is the definition of such standards.

- **Quality of service**: a minimal quality of service is required to be allowed to join the Grid. A site should sign a Memorandum of Understanding (MoU) where a set of minimum requirements is defined. These requirement should be met to guarantee a minimum quality of service. An example could be the percent of time the site has been available and its reliability. If those requirements are not met for a defined amount of time, appropriate actions can be undertaken. E.g.: the site could be suspended and it would need to pass all the certification tests to became again member of the Grid.

## 1.5 The EGI-InSPIRE project

The European Grid Initiative or EGI-InSPIRE represents an effort to establish a sustainable Grid infrastructure in Europe. Driven by the needs and requirements of the research community, it is expected to enable the next leap in research infrastructures, thereby supporting collaborative scientific discoveries in the European Research Area[8]. It represent the follow up and, at the same time the reorganization of the knowledge acquired during the last six years with the EGEE projects. In Figure 1.2 are shown the stages in the organization of this new entity.

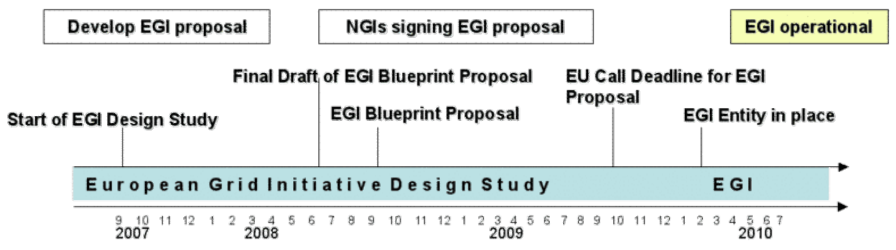

Develop EGI proposal
NGIs signing EGI proposal
EGI operational
Final Draft of EGI Blueprint Proposal
EU Call Deadline for EGI Proposal
Start of EGI Design Study
EGI Blueprint Proposal
EGI Entity in place
European Grid Initiative Design Study
EGI
2007
2008
2009
2010

Figure 1.2: EGI-InSPIRE timeline

From the beginning of 2010 EGI-InSPIRE has started its work of organization and coordination of the NGIs that, at this moment, are 42[9]. In Figure 1.3, a schema with the relationships present between EGI Global Service and the NGIs Opertaion Centre providing local services is shown.

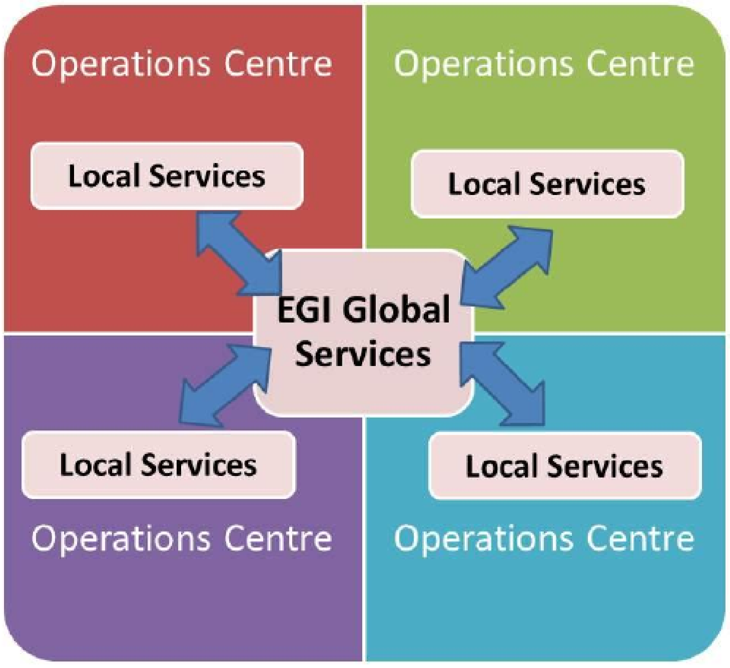


Figure 1.3: National Grid Initiatives

| **Application domain** | **# VOs** | **# users** |
|---|---|---|
| Computer Science and Mathematics | 8 | 22 |
| Multidisciplinary VOs | 33 | 1941 |
| Astronomy, Astrophysics and Astro-Particle Physics | 22 | 330 |
| Life Sciences | 16 | 709 |
| Computational Chemistry | 5 | 476 |
| Earth Sciences | 13 | 314 |
| Fusion | 2 | 13 |
| High-Energy Physics | 45 | 5732 |
| Infrastructure | 31 | 2042 |
| Others | 40 | 2105 |
| **TOTAL** | **215** | **13684** |

Table 1.1: Number of active VOs and users for each VO[10]

At local level, NGIs are required to perform some task like authentication, authorization, setup of Virtual Organizations (VOs), monitoring of the resources and the jobs submitted and processed, accounting of the user of a particular NGI, scheduled update of the software and maintaining a continuous service of the Grid. Those operations are necessary to maintain proper interaction with EGI-InSPIRE and the interoperability with the other NGIs. If we check the numbers involved it became clear why this kind of organization is necessary. In Table 1.1 is shown the number of VOs and users divided for each field of research. Along with the increase on the number of VOs and users, Figure 1.4 shows the number of jobs executed from the beginning of 2004 when EGEE started. It shows a constant increase in the utilization of the Grid resources during the last 6 years in the 56 countries[11] that participate in the project for a total of about 350 sites.

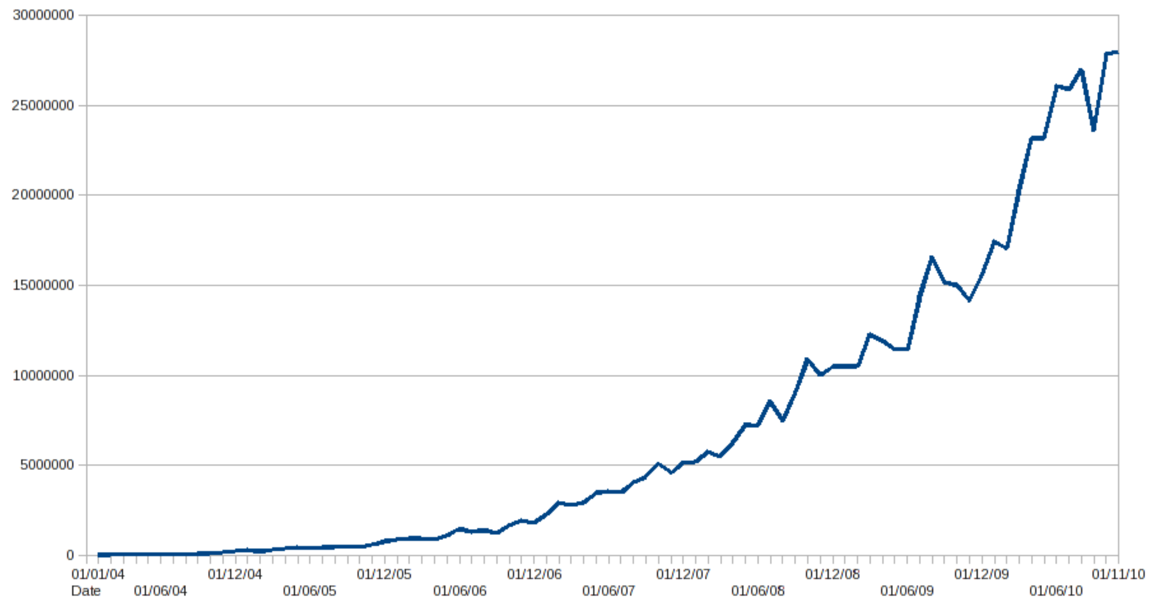


Figure 1.4: Number of job in the EGEE and EGI-InSPIRE[12]

## 1.6 gLite middleware components

The middleware of a Grid system should give, to the end users, a homogeneous view of the different components that are part of it. It is placed on top of the operating system and below the user applications. Associated with it are services with specific capabilities as defined in the EGI-InSPIRE Unified Middleware Roadmap

(UMD)[13]. It is possible to have different implementations of the same service provided that there is a standard interface to them. Those different implementation are usually chosen by the site for some specific characteristics that are needed or preferred in that particular site. The standardization of the interfaces ensure that, from the point of view of the users, there is no difference in using one implementation or another. In Figure 1.5 is shown a schema of the main components of the middleware.

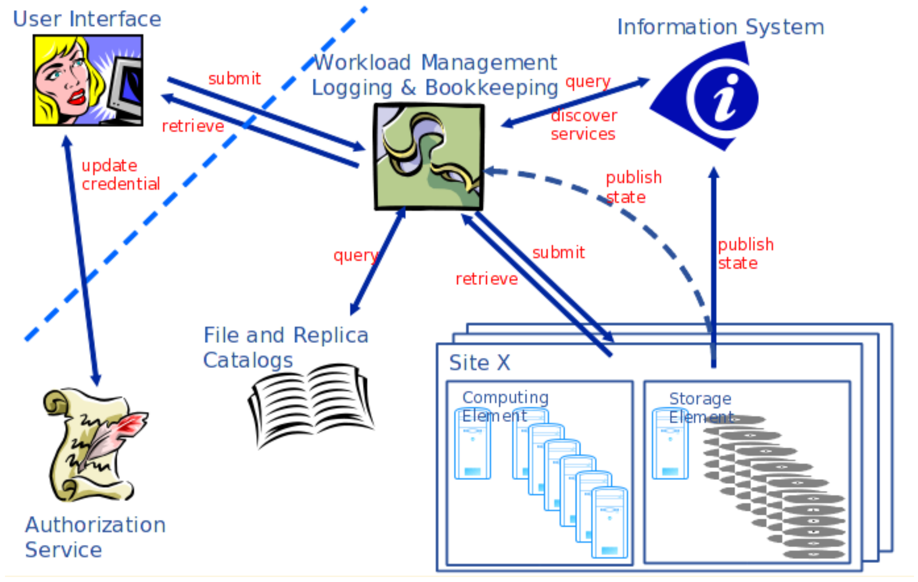
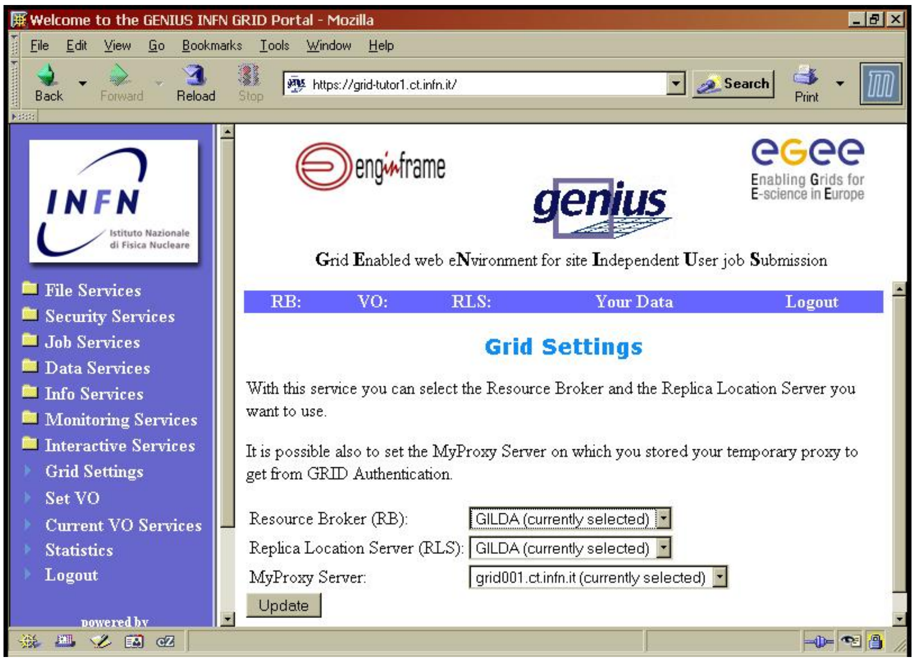


Figure 1.5: gLite architecture

The authentication and authorization components are only some of the elements of the infrastructure but they are central for the correct identification of the user requesting the Grid resources and they grant or deny the access based on the site policy. The authorization on EGI-InSPIRE is based on x.509 certificates which is an International Organization for Standardization (ISO) standard. For the Italian NGI, those certificates are distributed by the INFN Certification Authority (INFN CA) after a procedure that requires the personal identification of the person requiring it. User credentials are divided in two parts. One is the public certificate that is signed by the CA and the other is the private key of the users. Upon authentication the user must obtain an authorization to execute the required operations. There are different levels of authorization, called roles. A user can have administrative privileges, for example, or only user privileges. The authorization to access the resources at VO level, providing support to group membership and roles, is demanded to the Virtual Organization Membership Service (VOMS) service[14]. After being authenticated and authorized, the user can utilize all the resources available to its VO and role. Depending on internal policies, founding or, e.g. fields of interest, each site can choose the VOs to support. If a site choose to support a certain VO it should consequently support all the roles associated to this VO. The next sections will describe the key components of Grid. The focus will be on the gLite middleware because it is the one used in the Italian Grid.

### 1.6.1 User Interface

The User Interface (UI) is the entry point to the Grid. It is usually a standard Unix system in which a set of client tools and Application Programming Interface (API) are installed. The user can log in with a standard login and password. After successfully accessing the system, the tools and API are available. Some other kind of UI exists like the GENIUS Portal shown in Figure 1.6[15]. It is a web portal studied to make the process of accessing the Grid resources an easier task.


Welcome to the GENIUS INFN GRID Portal - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
https://grid-tutor1.ct.infn.it/
Search
Print
INFN Istituto Nazionale di Fisica Nucleare
File Services
Security Services
Job Services
Data Services
Info Services
Monitoring Services
Interactive Services
Grid Settings
Set VO
Current VO Services
Statistics
Logout
powered by
enginframe
genius
egee Enabling Grids for E-science in Europe
RB: VO: RLS: Your Data Logout
Update

Grid Enabled web eNvironment for site Independent User job Submission

Grid Settings

With this service you can select the Resource Broker and the Replica Location Server you want to use.

It is possible also to set the MyProxy Server on which you stored your temporary proxy to get from GRID Authentication.

Resource Broker (RB): GILDA (currently selected)
Replica Location Server (RLS): GILDA (currently selected)
MyProxy Server: grid001.ct.infn.it (currently selected)

Figure 1.6: GENIUS Portal

The authentication process, in both cases is based on *x.509* certificates that are either copied on the personal account of the UI utilized or installed in the web browser. When a user submit a request, he or she is mapped to a local user, chosen from a local user pool determined by the VO membership and role. Thanks to this mechanism the local resource do not need to have a list of all the possible users that can have access but they only need to keep a limited pool of user for each VO and role.

### 1.6.2 The Information Service

A series of modules collect information on the different resources provided by the different sites. On the base of the information collected a set of different action can be taken. For example some of the information provided are the available CPUs, their type or architecture, software installed, etc. Based on those information the Workload Management System (WMS) can decide to send a job to a site instead of another. All data is published by a Berkeley Database Information Index (BDII) that consists of a Lightweight Directory Access Protocol (LDAP) database which is updated by an external process. The update process obtains LDAP Data Interchange Format (LDIF) fresh information from a number of sources and merges them. It then compares this to the contents of the database and creates an LDIF file of the differences. This is then used to update the database[16]. A schema of this process is shown on Figure 1.7.

The information present in the BDII are published using the Grid Laboratory for a Uniform Environment (GLUE) schema defined by the corresponding GLUE-WG (GLUE Working Group) which is part of OGF. Moreover the BDII can be organized in a hierarchy, Figure 1.8, in which the top level BDIIs are configured to read from a set of sites giving, as result, a view of the overall resources available to those sites[17].

### 1.6.3 Workload Management System and Logging & Bookkeeping

The WMS is the component that allows users to submit jobs and performs all tasks required to execute them without exposing the complexity of the Grid[16]. The user provides a description of the job and the requirements in term of resources, software and others attributes by defining a Job Description Language (JDL) file for that

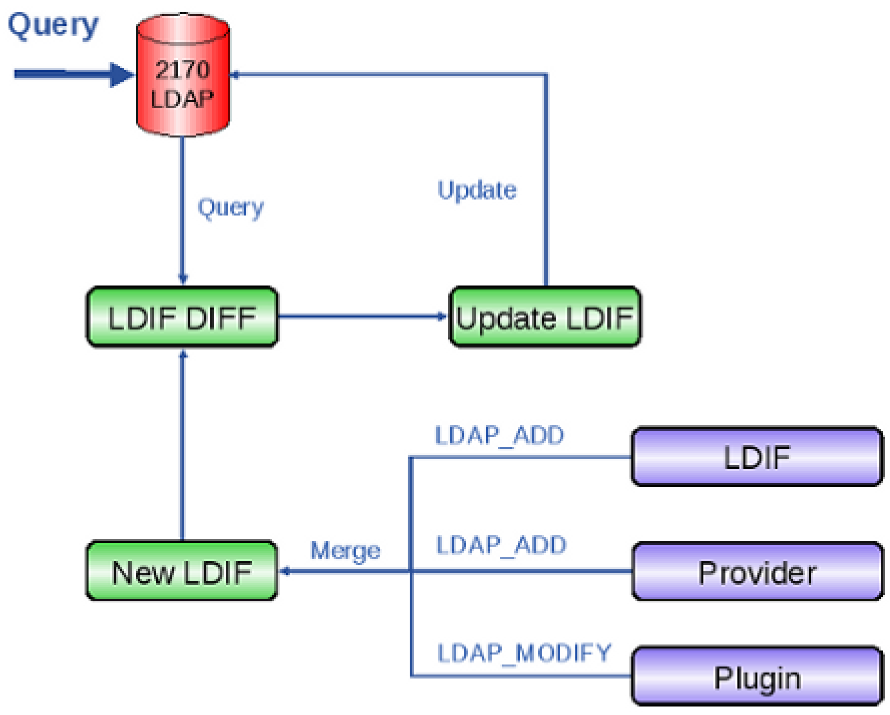


Figure 1.7: BDII update schema

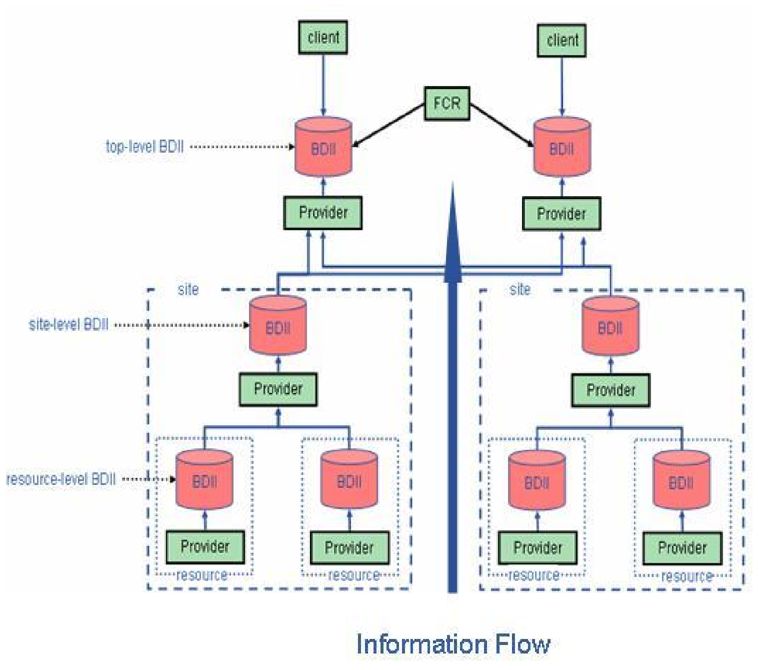


Figure 1.8: BDII hierarchy schema

particular job. JDL is the scripting language used to define the requirements, input and output, resources needed and other details. The WMS decides which resource is more suitable to execute the job submitted taking decisions based on the JDL file submitted. Once the job reaches the WMS it goes trough a series of intermediate steps to prepare its execution, match it to the appropriate resource, send it for execution and check its status. After the submission, the WMS, keeps tracks of the progress. Depending on the configuration given by the JDL written by the user, it might resubmit the job in case of errors. Figure 1.9 shows a schema of the different job statuses and the internal components of the WMS. The interactions between the WMS and the other Grid components are also shown. The WMS is in fact a fundamental part of this workflow, it is the glue between the UI, the Information System (IS) and the Computing Elements (CEs).

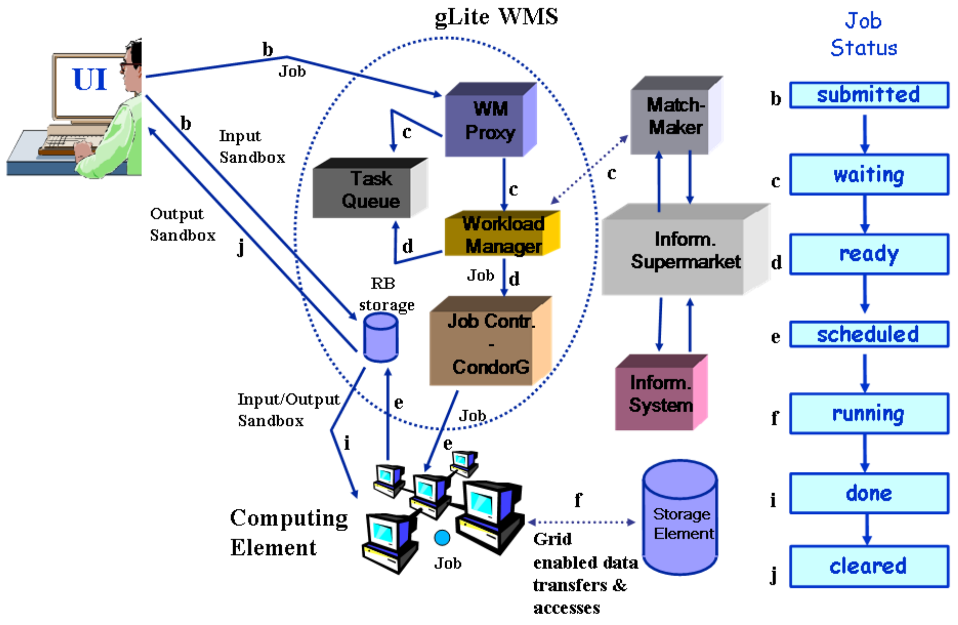


Figure 1.9: WMS job flow

The Logging & Bookkeeping (LB) is part of the WMS but, in the latest software distribution, a dedicated machine is recommended. It is essentially a database that contains all the details available on each job submitted to the WMS, the changes in their status, errors, resubmissions, etc. A user with the right privileges, using the unique job ID that is generated when the job is submitted, can query the LB to check the status of a given job. On the other hand, the output files resulting from the job are stored on the WMS and are not removed till the user clear the job or a defined amount of time, typically two weeks, has passed. In the latest releases, the LB reside, for convenience, in a different machine. Because of the different utilization of resources, several WMS can point to the same LB to store their logging data.

### 1.6.4 Computing Element

The CE is the service that interfaces a Local Resource Management System (LRMS) like Load Sharing Facility (LSF), Portable Batch System (PBS) or Sun Grid Engine (SGE), to the rest of the Grid. The available implementations are LHC Computing Grid CE (lcgCE), based on the Globus Toolkit, an open source software toolkit used for building Grids that is being developed by the Globus Alliance and many others all over the world[18], or the Computing Resource Execution And Management (CREAM CE). The former is the old implementation and is being replaced by the CREAM CE which is a simple, lightweight service for job management[19]. Jobs can be sent to the CREAM CE directly or by using the WMS through the Interface to CREAM Environment (ICE). When using the WMS two possible configuration are possible:

- **pull mode**: the CE requires new jobs to the WMS when it has free resources. While requesting those new jobs it specify the availability of the free resources to allow the WMS to choose the right job, if any available. There are two strategy that can be used when the CE is configured in *pull mode*:

  - a job is requested to a WMS, if none satisfy the request another WMS is contacted;
  - a job is requested to a series of WMS and the first job sent is accepted while the others are discarded.

  Which choice is the best depends on the particular situation and some test should be performed for different experiment, workload, etc.

- **push mode**: in this configuration the CE waits for jobs coming from an external source like the WMS or directly from the user.

### 1.6.5 Worker Node

When 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 Worker Node (WN). The WN is the machine that executes the process requested. It is essentially, a UI that includes commands to manipulate the jobs and has access to a software area where are installed the programs required. The job is executed by assigning an appropriate local user to the Grid user.

### 1.6.6 Storage Element

The Storage Element (SE), using standard interfaces, provides access to the storage resources deployed in a certain site. Those resources can be heterogeneous, e.g.: disk servers, Storage Area Network (SAN), tape server, etc. The implementations available in gLite are Storage Resource Manager (StoRM), dCache or Disk Pool Manager (DPM). They all implement the Storage Resource Manager (SRM) interface to interact with the underlying storage and, at the same time, keeping the implementation details hided to the user. They also offer standard transfer mechanisms for remote access. Like for all other services, the access to the SE depends on the user credentials. In this way the system can allow or deny access and manage the space with quota, space reservation and pinning. Because of the intrinsic distributed nature of the Grid, one problem is to uniquely identify a file. Moreover, it is important do identify replicas of the same file or the same file present in different location and with different names.

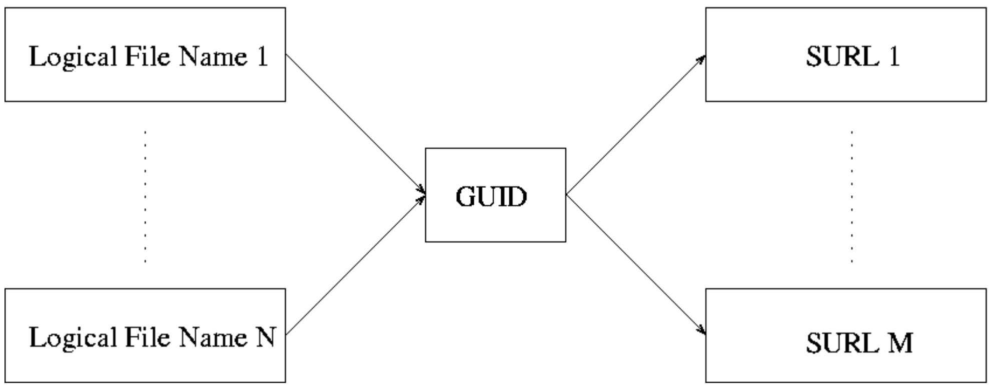


Figure 1.10: File name in gLite and their relation

In Figure 1.10 are shown the names and their relationships that a file can assume. The Globally Unique IDentifier (GUID) and the Logical File Name (LFN) are independent from the location of the file. A file can be unambiguously identified by its GUID which is assigned the first time the file is registered in the Grid. It is based on the Universally Unique IDentifier (UUID) standard to guarantee its uniqueness. The UUID use a combination of a Media Access Control (MAC) address and a timestamp to ensure that all UUIDs are distinct[16]. To access the desired file the user utilize the LFN that are defined by the user. Local File Catalog (LFC) servers are databases containing a list of correspondences between LFN, GUID and Storage URL (SURL). When accessing a SE, the standard protocols use the Transport URL (TURL) address, given upon request by the SE itself, to write or retrieve a physical replica. The service that manage the transfers is File Transfer Service (FTS). It interacts with the SRM source and destination using the *gridFTP* protocol. This allows the transfer of huge amount of data using multiple streams, queue of transfers, detection of errors and rollback.

# Chapter 2

# Accounting of computational resources

*"If you want an accounting of your worth, count your friends."*
Merry Brown, Author

## 2.1 Accounting on a distributed environment

Grid accounting encompasses the production and management of large amount of accounting data, their reliable transfer to central repositories as well as end users data access to it. The amount of running jobs on nowadays Grids and the multi Petabyte data volume already deployed, require an accounting service that collects different data usage information to allow the presentation of the accounting data to the experiments, the site and the user. Accounting systems must be capable of showing the accounting information from different perspectives. In general, the systems developed up to now build up from existing infrastructure. An accounting system can be defined as one or more applications that allows the storing accounting information related to resources and should be able to classify them according to different classes (users, groups, VOs, etc.). To complete the set of applications we also need instruments to retrieve those information for analysis and for further billing of the used resources. The steps necessary for this process are the following:

- *metering*: a set of necessary metering values should be identified for the accounting system, which defines what should be measured and the relevant information;

- *distribution* and *collection*: the information can be locally collected and when necessary they should be sent to one or more distributed accounting repositories to be stored, e.g. in a database, for further analysis;
- *usage analysis*: the accounting information can then be analyzed, reports can be produced and Billing/Pricing can be applied.

Different accounting systems use different methods to collect, distribute and analyze the information. In general, they all rely on sensors installed on the resource to be accounted. A record containing the relevant information related to the resource to be accounted is produced and collected in a repository for further analysis. A web page or higher level tools might exist to make the last operation easier. The most important accounting systems, developed on the framework of the European projects mentioned are SweGrid Accounting System[20] (SGAS), Accounting Processor for Event Logs[21] (APEL) and DGAS. Among them I have developed my work around DGAS because of its flexibility and because it is the tool used on the Italian Grid which counts more than 50 sites.

## 2.2 DGAS accounting system

DGAS is an accounting system developed for the accounting of computational resources. It has been adopted as the accounting tool for the Italian Grid and thus deployed in every Italian site. It is worth mentioning that some other countries and projects are using DGAS, e.g. sites in Greece and Germany and projects such WeNMR, whose main objective is to provide the European biomolecular Nuclear Magnetic Resonance (NMR) user community with a platform integrating and streamlining the computational approaches necessary for biomolecular NMR data analysis (e-NMR)[22]. DGAS is composed of:

- specialized sensors that are installed on the computing resource;
- a server that stores the records collected by the sensors.

As we will see in more detail in the following sections, DGAS's design is flexible and allows a hierarchical structure for the database repositories, where the information are collected at different levels in servers called HLR. The records can be stored at a site HLR and then forwarded to a national or regional HLR. Finally data can be accessed through different interfaces as shown in Figure 2.1. Since DGAS has been designed for a distributed system, there is no need for a central repository and a flexible configuration can be made to suit all common scenarios.

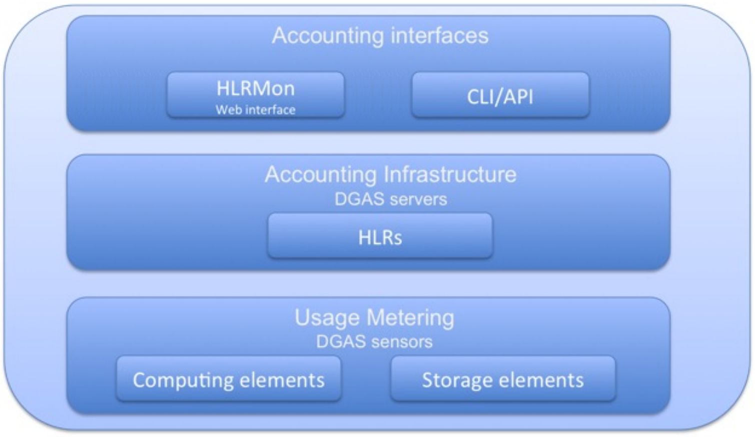


Figure 2.1: DGAS layers[23]

Sensors and HLR installation is possible with an automated procedure or manually. YAIM Ain't an Installation Manager[1] (YAIM) is used for the automated procedure. This tool allows, for each Grid service (included DGAS), a specific profile that make the installation and configuration an automated process. Before starting the installation, a customization of the profiles is required; as a result, all the configuration files will be generated with the proper variables set. Manual installation uses RPM Package Manager (RPM) packages. Using this method, the configuration files must be manually edited after installation.

### 2.2.1 Sensors

Each resource needs specialized sensors. For computational resources, specific sensors for LSF, PBS and SGE batch systems and for both the CREAM CE and lcgCE have been developed. Figure 2.2 shows a schema of this process.

[1]YAIM is a tool developed during EGEE project that provides a simple installation and configuration method that can be used to set up a simple Grid site but can be easily adapted and extended to meet the need of larger sites[24]

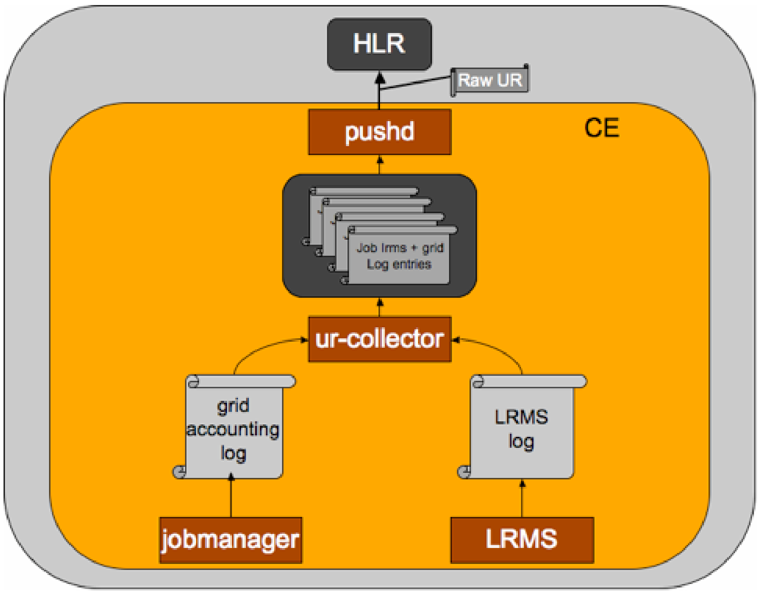


Figure 2.2: DGAS sensor schema for a CE[25]

The information necessary to build the UR are collected from the CE itself. The sensor consists of two components:

- *urcollector*: regularly reads both the grid accounting and the specific LRMS logs to build an accounting record for each corresponding job. Each record is saved on the CE as an eXtensible Markup Language (XML) file in a configurable location. The process of reading and generating URs is customizable and it is possible to reprocess old log if necessary.

- *pushd*: this process checks the location where the *urcollector* stores the URs. When it finds a record, it tries to send it to the configured HLR. In case of failure it retries a configurable number of times before moving the UR to a specific error directory containing unsent records. This allows to reprocess URs by simply moving back to the previous location.

### 2.2.2 Home Location Register repository

The Home Location Register (HLR) consists of two main parts:

- *hlrd* daemon: this process runs on the HLR and listens for messages coming from the *pushd* daemon on the CE. The CE sending the messages can be on the same site of the HLR or somewhere else. After being authenticated and authorized it is in fact possible, for small Grid installations or for a specific VO, to send their URs to a multi site or VO specific HLR.

- *database*: the database stores all the UR received from different sources. Several tables have been defined, among which the table that contains the list of resources allowed to account to the specific HLR, list of users allowed to query the database through the HLR interface and the table that contains the accounting data. This last table is used by higher level tools like HLRmon, a web interface that allows the visualization and aggregation of the accounting data, and by authenticated and authorized users, to visualize the accounting data present in the HLR database.

### 2.2.3 HLR configurations

Due to the flexibility of DGAS architecture, several configurations of the HLR are possible. Some scenarios will be described in this section. In Figure 2.3 a first scenario where each site has its own HLR collecting accounting data from local CEs and SEs is shown.

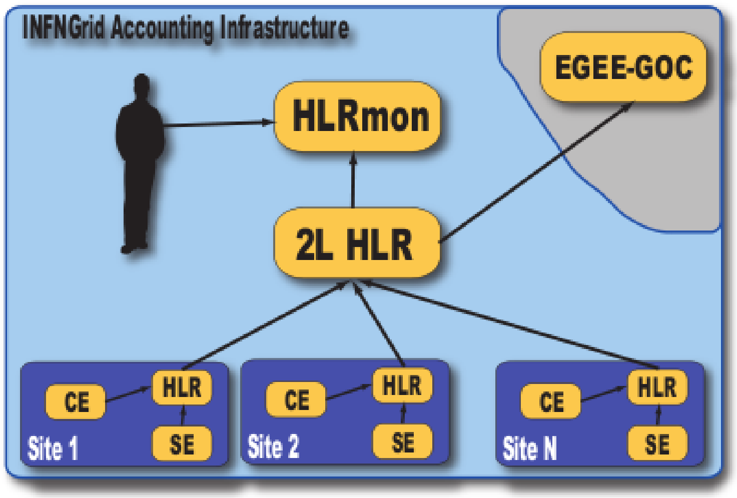


Figure 2.3: DGAS sensor example schema 1[26]

A second level HLR collects all data from first level HLRs. HLRmon is interfaced to the second level HLR. From the second level HLR it is also possible to send the collected URs to an external repository. For the EGEE and the subsequent

EGI-InSPIRE projects, the central European repository is hosted at Centro de Supercomputación de Galicia (CESGA). A second different scenario is shown on Figure 2.4. In this case HLRs collect accounting data for all the first level HLRs within a NGI in order to have fault tolerance. Data collected from both those second level HLRs can be then queried by HLRmon. Another second level HLR, that collects only the records belonging to a subset of VOs, could also be configured and it can get data from different HLRs belonging to different NGIs.

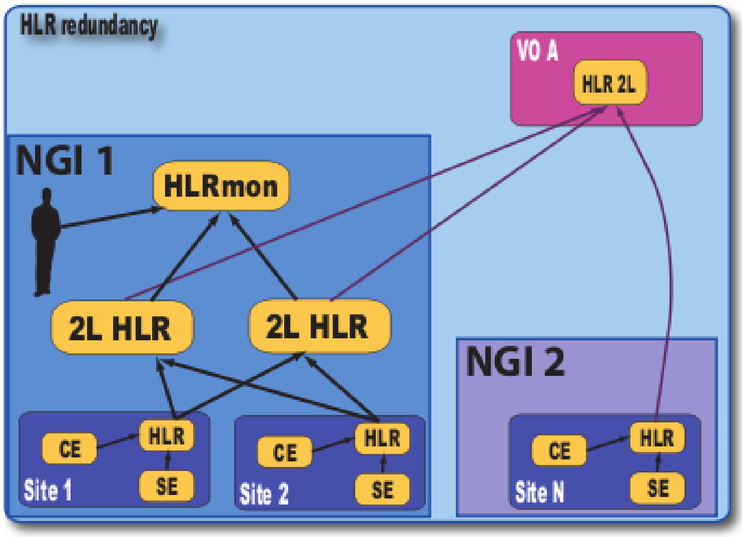


Figure 2.4: DGAS sensor example schema 2[26]

Figure 2.5 shows an extension of the previous scenario where multiple second level HLRs collect different subset of records belonging to different VOs. This configuration is very useful where a VO wants to keep control over the accounting records belonging to itself. These scenarios have been presented in the poster "*Implementing a National grid accounting infrastructure with DGAS*" during the EGEE'09 Conference held in Barcellona from the 21$^{st}$ to the 25$^{th}$ of September 2010[26].

### 2.2.4 HLRmon

HLRmon is the tool used to produce reports for the accounting data collected by DGAS. It can produce different aggregates of data with different level of detail. The architecture of the system is shown in Figure 2.6. HLRmon server retrieves the accounting data from the HLR by means of the DGAS client. The client contacts the

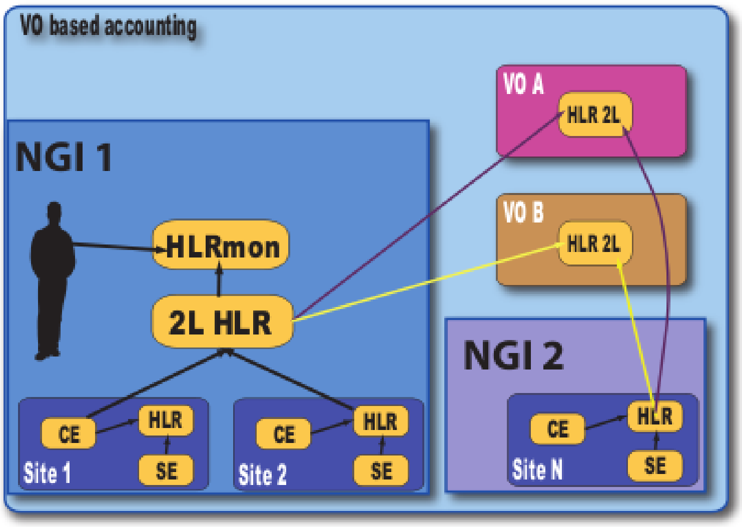


Figure 2.5: DGAS sensor example schema 3[26]

HLR and can be used for advanced queries, such as to retrieve lists of transactions as well as aggregated information on users, resources, groups, roles and VOs. The authorization to access the accounting data is based on a valid x.509 certificate that is checked by the HLR server every time a query is submitted. Grid users have access to all aggregate statistics and can retrieve information related to jobs submitted using their own x.509 certificate. Only users mapped as HLR administrators can access all the usage records using this DGAS client.

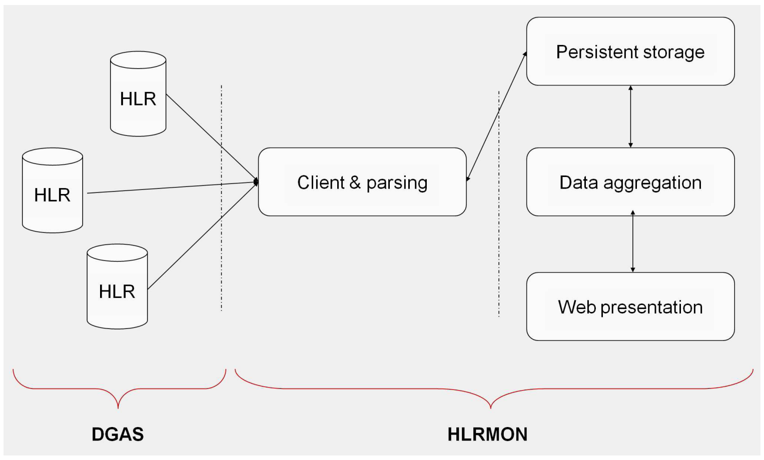


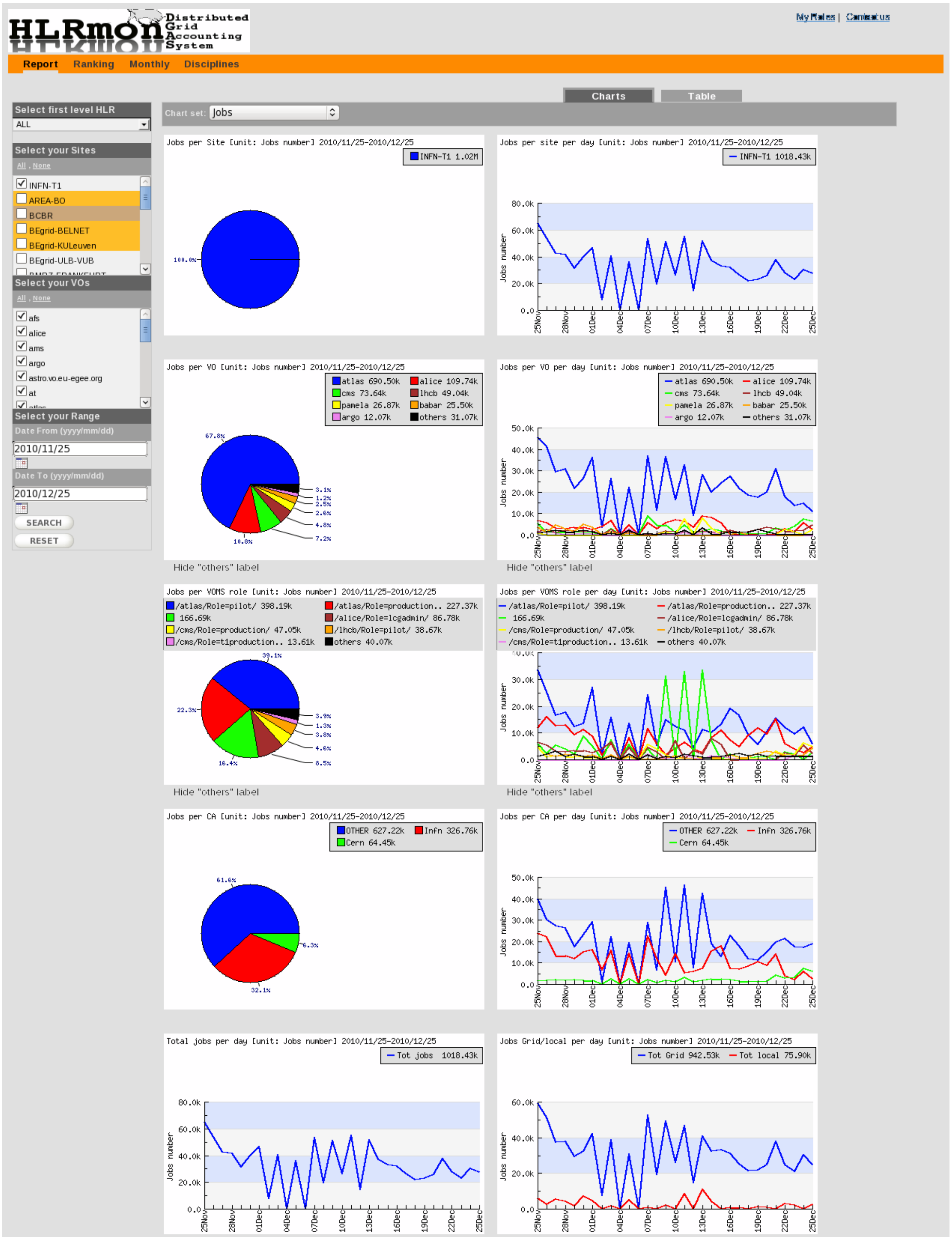
Figure 2.6: HLRmon Architecture[27]

The result of the queries made by HLRmon is saved in a local MySQL[28] database after perfoming some aggregation. The process also creates some static images to allow a faster response of the server. In Figure 2.7 we can observe a snapshot of the monthly statistics for the computational resources of the main Italian Grid site, the INFN-T1. Pages with the details of the per user accounting are accessible only after a registration and authorization procedure available on the website. The access is based on a x.509 personal certificate both at web server and application level. The authorization process for the different roles and access to data from different sites and VOs is done by the administrator of the HLRmon service.

In Figure 2.7 we can see:

- on the left there is a selection form where we can select sites, VOs and the period of time we want to visualize;
- on the central part of the page, a series of pie charts and line graphs with the details of the selected accounting data is shown;
- on the top of the page, just over the graphs, it is possible to choose which kind of information we want to display (e.g.: jobs, CPUtime, Wallclock time, etc.);
- on the top right side of the page a tab to switch the visualization to a tabular view is also present.

Dedicated pages for the visualization of a summary of sites utilization and the storage accounting for the LHC experiments are available. The page dedicated to the storage accounting will be discussed in more detail in Chapter 4.


HLRmon Distributed Grid Accounting System
My Roles | Contact us
Report
Ranking
Monthly
Disciplines
Charts
Table
SEARCH
RESET
Hide "others" label

Select first level HLR: ALL

Select your Sites: All , None
- [✓] INFN-T1
- [ ] AREA-BO
- [ ] BCBR
- [ ] BEgrid-BELNET
- [ ] BEgrid-KULeuven
- [ ] BEgrid-ULB-VUB

Select your VOs: All , None
- [✓] afs
- [✓] alice
- [✓] ams
- [✓] argo
- [✓] astro.vo.eu-egee.org
- [✓] at

Select your Range

Date From (yyyy/mm/dd): 2010/11/25

Date To (yyyy/mm/dd): 2010/12/25

Chart set: Jobs

Jobs per Site [unit: Jobs number] 2010/11/25-2010/12/25
INFN-T1 1.02M
100.0%
Jobs per site per day [unit: Jobs number] 2010/11/25-2010/12/25
INFN-T1 1018.43k
Jobs number
Jobs per VO [unit: Jobs number] 2010/11/25-2010/12/25
atlas 690.50k
alice 109.74k
cms 73.64k
lhcb 49.04k
pamela 26.87k
babar 25.50k
argo 12.07k
others 31.07k
67.8%
10.8%
7.2%
4.8%
2.6%
2.5%
1.2%
3.1%
Jobs per VO per day [unit: Jobs number] 2010/11/25-2010/12/25
Jobs per VOMS role [unit: Jobs number] 2010/11/25-2010/12/25
/atlas/Role=pilot/ 398.19k
/atlas/Role=production.. 227.37k
166.69k
/alice/Role=lcgadmin/ 86.78k
/cms/Role=production/ 47.05k
/lhcb/Role=pilot/ 38.67k
/cms/Role=t1production.. 13.61k
others 40.07k
39.1%
22.3%
16.4%
8.5%
4.6%
3.8%
1.3%
3.9%
Jobs per VOMS role per day [unit: Jobs number] 2010/11/25-2010/12/25
Jobs per CA [unit: Jobs number] 2010/11/25-2010/12/25
OTHER 627.22k
Infn 326.76k
Cern 64.45k
61.6%
32.1%
6.3%
Jobs per CA per day [unit: Jobs number] 2010/11/25-2010/12/25
Total jobs per day [unit: Jobs number] 2010/11/25-2010/12/25
Tot jobs 1018.43k
Jobs Grid/local per day [unit: Jobs number] 2010/11/25-2010/12/25
Tot Grid 942.53k
Tot local 75.90k
25Nov
28Nov
01Dec
04Dec
07Dec
10Dec
13Dec
16Dec
19Dec
22Dec
25Dec

Figure 2.7: HLRmon computational resources[29]

# Chapter 3

# DGAS assessment

*"Don't worry if it doesn't work right. If everything did, you'd be out of a job."*
Mosher's Law of Software Engineering

## 3.1 Accounting complexities

As we saw in the previous sections, accounting systems are, usually, composed of different parts that must interact with each other. Accounting information must be produced after the analysis of several log files and then sent to a first level HLR to be processed. The validation of the collected data is therefore necessary because all those steps must be working properly in order to have correct accounting information. When an anomaly is detected, it is possible to track it down following the accounting path and correct it. However, it is often very difficult to spot e.g.: a misconfiguration thus some other method must be devised. The INFN-CNAF's Tier1 uses a tool, internally developed, called Redeye[30]. Among other things it provides information on the use of the computational resources of the INFN-T1 and INFN-CNAF-LHCB sites. Both systems, DGAS and Redeye, analyze the log files generated by the batch system that in this particular case is LSF and the grid system which save important accounting information on the CEs' log files. Because of the presence of those two accounting systems on the same site it is possible to compare the results shown by both. This can highlight possible discrepancies between them caused by problems in one of the two tools or changes in the configuration of the accounted resource, e.g.: a new queue has been added to the batch system but not in DGAS. For this purpose a dedicated tool called *cross-check* that queries those two systems and compare them with a third one that, with a simple Perl script, extracts the same kind of information from the batch log files has been developed during the thesis work. After the validation of the accounting data, the tool is helping in keeping the accounting status of INFN-T1 and INFN-CNAF-LHCB sites under constant control and to detect possible problems or changes in the configuration of the system that could affect the accounting itself. For this reason, it is still used to regularly check the two installation that are in place. Those installations are independent from each other allowing a separate configuration and personalization.

### 3.1.1 cross-check's architecture

cross-check is composed of three parts:

- **scripts**: a series of scripts queries Redeye and HLR site database and populate the cross-check local database. A third source of information is also added, resulting from the analysis of the batch log files, is added. The information collected by the three system is slightly different because of the way it is collected and because the systems they were built for were meant for different purposes. For those reasons another series of scripts is necessary to modify those information allowing an easier comparison. Finally the data are aggregated in a separated table that is used by the web interface. Because of the staticity of the information contained in the database, the daily aggregate are generated offline for a faster response. All the updates and aggregation scripts are run once a day and there is no need to run them more often because the tool is not intended for real time checks;
- **database**: cross-check's MySQL database contains all the tables necessary to store the information extracted with the previously described script as well as the daily aggregate;
- **web interface**: the web interface is the front end to the database. Using that interface the user can check if there are discrepancies on the different accounting systems. Some of the differences might be normal because of the different way the accounting data are collected. To minimize them, some of the scripts might be optimized in a specific way dependent on the site.

In Figure 3.1 a schema of the scripts that are run daily and their relationships is shown. The schema is divided in four sections.

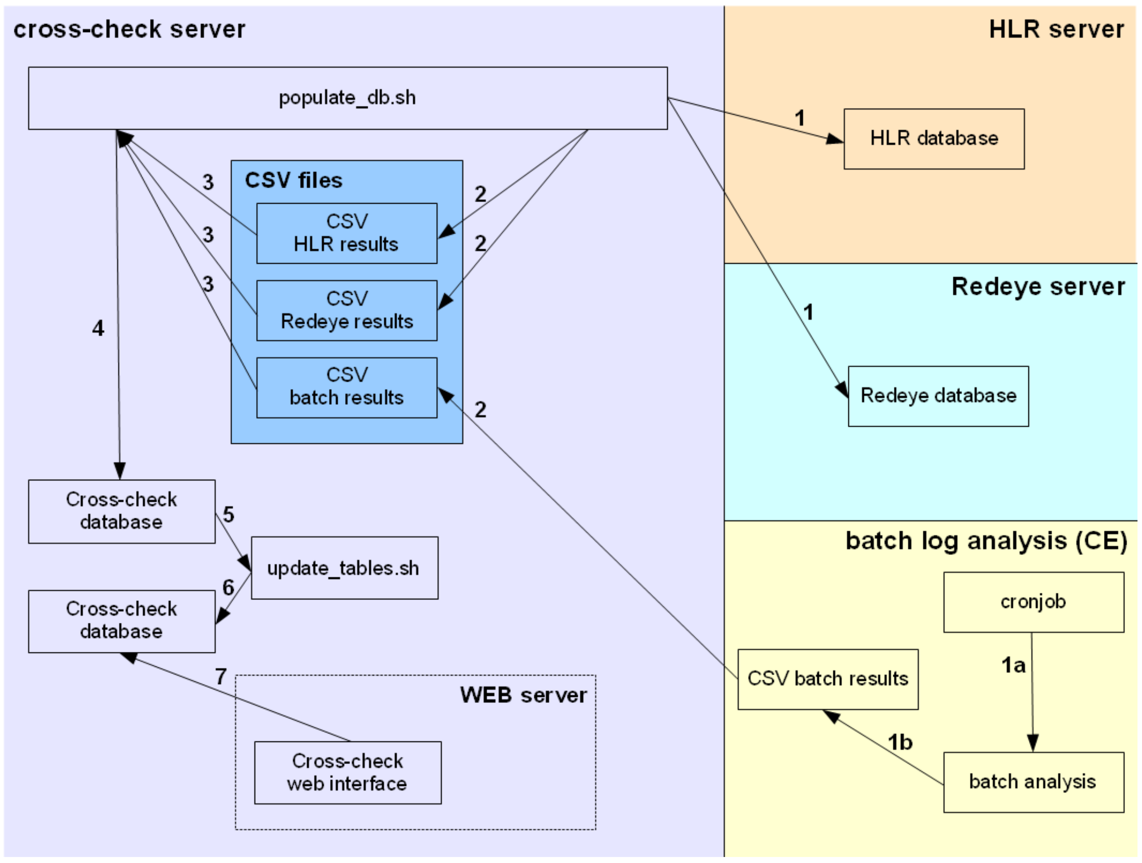


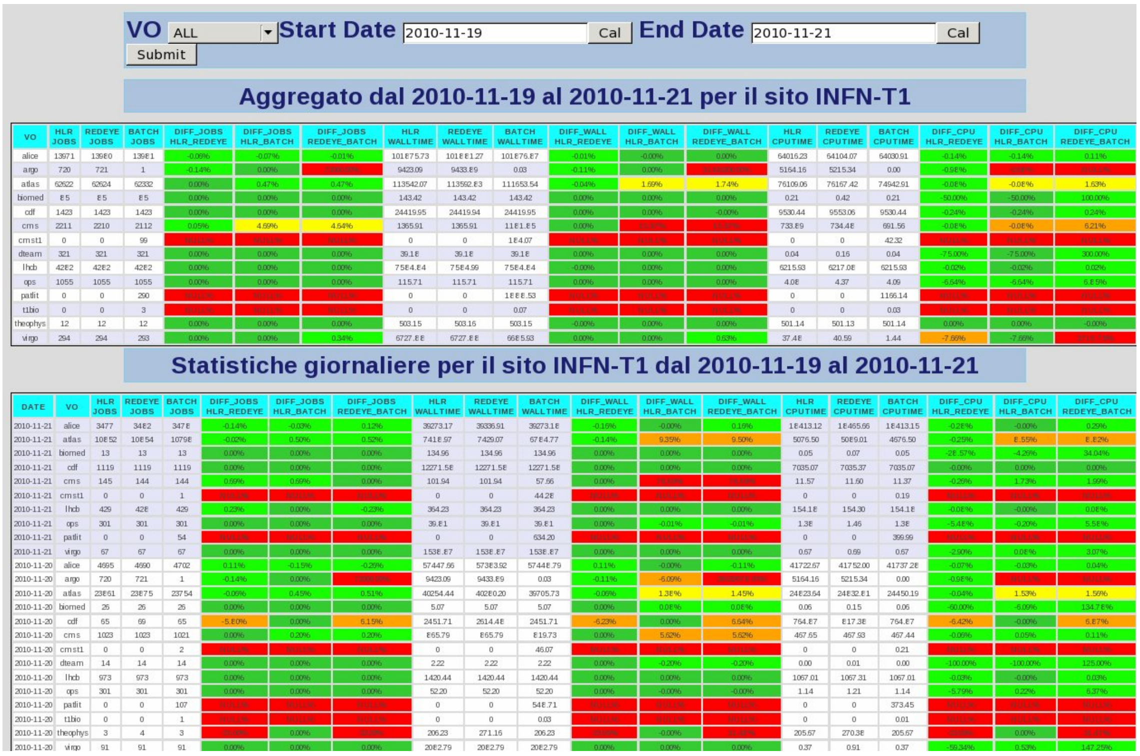
Figure 3.1: *cross-check* script relationships

On the left side is the server that contains cross-check scripts, while on the right side are the three servers that contains the different data needed by the system. Starting from the top are: the HLR server, the Redeye server and a CE from which the batch system log files are readable. Because of the particular configuration used at INFN-T1 and INFN-CNAF-LHCB sites, all the CEs have access to the full log files of the batch system. For this reason it is enough to analyze the logs on only one of the available CEs because the logs are contained in a shared disk area and is the same for all of them. The information necessary to cross-check is collected in three Comma Separated Value (CSV) files. The one on the CE is produced on the same computer by running the script *convert-batch_log.sh* every night. The content of this script, and all the other that are part of cross-check tool, is on Appendix A. This script must be configured, depending on the particular CE and batch system installed. It requires to know where to find log files and, consequently, which Perl script needed to analyze them. Two scripts, one for LSF and one for PBS, are available. The result of the the log analysis (steps 1a and 1b on Figure 3.1) is a CSV file that contains the record extracted. The last operation performed is the copy to the cross-check server (step 2 from the CE to the cross-check server). Next, the *populate_ db.sh* and *update_ tables.sh* scripts are run. This creates the other two CSV files by directly querying the HLR and Redeye (step 1). The result of those queries are the remaining two CSV files that are recorded on the local cross-check's disk (steps 2 from *populate_ db.sh* to the HLR and Redeye CSV files). When all the CSV files are ready to be processed the *populate_ db.sh* proceed by inserting them in the MySQL database (steps 3 and 4). At this point the *populate_ db.sh* ends and the *update_ tables.sh* starts. It queries the database, update the tables used to insert the previous records and prepare them for the visualization on the web interface (steps 5 and 6). The web interface requires only a web server that supports PHP: Hypertext Preprocessor (PHP). It is configured to access the database to retrieve the needed records (step 7). It can be installed in the same machine in which the cross-check system is installed, or in another one. A proper configuration of MySQL database might be required to allow the network connection from the web server.

### 3.1.2 cross-check's web interface

In Figure 3.2 the cross-check web interface for the INFN-T1 site is shown. It is divided in three parts:

- **date range selection**: it is the top section where the user can choose the date range that he or she wants to visualize. When the page is first loaded, the default is to show the last week of data;
- **summary accounting**: this section is below the date range selection and shows an aggregation of all the records in the range selected. It is divided only per VO and the other values are the sum for the period selected;
- **daily accounting**: can be found on the lower part of the page and shows similar information as in *summary accounting*. The only difference is that they are divided in a per day basis on the selected range.



VO ALL Start Date 2010-11-19 Cal End Date 2010-11-21 Cal
Submit

Aggregato dal 2010-11-19 al 2010-11-21 per il sito INFN-T1

| VO | HLR JOBS | REDEYE JOBS | BATCH JOBS | DIFF_JOBS HLR_REDEYE | DIFF_JOBS HLR_BATCH | DIFF_JOBS REDEYE_BATCH | HLR WALLTIME | REDEYE WALLTIME | BATCH WALLTIME | DIFF_WALL HLR_REDEYE | DIFF_WALL HLR_BATCH | DIFF_WALL REDEYE_BATCH | HLR CPUTIME | REDEYE CPUTIME | BATCH CPUTIME | DIFF_CPU HLR_REDEYE | DIFF_CPU HLR_BATCH | DIFF_CPU REDEYE_BATCH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| alice | 13971 | 13980 | 13981 | -0.06% | -0.07% | -0.01% | 101875.73 | 101881.27 | 101876.87 | -0.01% | -0.00% | 0.00% | 64016.23 | 64104.07 | 64030.91 | -0.14% | -0.14% | 0.11% |
| argo | 720 | 721 | 1 | -0.14% | 0.00% | [illegible] | 9423.09 | 9433.89 | 0.03 | -0.11% | 0.00% | [illegible] | 5164.16 | 5215.34 | 0.00 | -0.98% | [illegible] | [illegible] |
| atlas | 62622 | 62624 | 62332 | 0.00% | 0.47% | 0.47% | 113542.07 | 113592.83 | 111653.54 | -0.04% | 1.69% | 1.74% | 76109.06 | 76167.42 | 74942.91 | -0.08% | -0.08% | 1.63% |
| biomed | 85 | 85 | 85 | 0.00% | 0.00% | 0.00% | 143.42 | 143.42 | 143.42 | 0.00% | 0.00% | 0.00% | 0.21 | 0.42 | 0.21 | -50.00% | -50.00% | 100.00% |
| cdf | 1423 | 1423 | 1423 | 0.00% | 0.00% | 0.00% | 24419.95 | 24419.94 | 24419.95 | 0.00% | 0.00% | -0.00% | 9530.44 | 9553.06 | 9530.44 | -0.24% | -0.24% | 0.24% |
| cms | 2211 | 2210 | 2112 | 0.05% | 4.69% | 4.64% | 1365.91 | 1365.91 | 1181.85 | 0.00% | [illegible] | [illegible] | 733.89 | 734.48 | 691.56 | -0.08% | -0.08% | 6.21% |
| cmst1 | 0 | 0 | 99 | [illegible] | [illegible] | [illegible] | 0 | 0 | 184.07 | [illegible] | [illegible] | [illegible] | 0 | 0 | 42.32 | [illegible] | [illegible] | [illegible] |
| dteam | 321 | 321 | 321 | 0.00% | 0.00% | 0.00% | 39.18 | 39.18 | 39.18 | 0.00% | 0.00% | 0.00% | 0.04 | 0.16 | 0.04 | -75.00% | -75.00% | 300.00% |
| lhcb | 4282 | 4282 | 4282 | 0.00% | 0.00% | 0.00% | 7584.84 | 7584.99 | 7584.84 | -0.00% | 0.00% | 0.00% | 6215.93 | 6217.08 | 6215.93 | -0.02% | -0.02% | 0.02% |
| ops | 1055 | 1055 | 1055 | 0.00% | 0.00% | 0.00% | 115.71 | 115.71 | 115.71 | 0.00% | 0.00% | 0.00% | 4.08 | 4.37 | 4.09 | -6.64% | -6.64% | 6.85% |
| patlit | 0 | 0 | 290 | [illegible] | [illegible] | [illegible] | 0 | 0 | 1888.53 | [illegible] | [illegible] | [illegible] | 0 | 0 | 1166.14 | [illegible] | [illegible] | [illegible] |
| t1bio | 0 | 0 | 3 | [illegible] | [illegible] | [illegible] | 0 | 0 | 0.07 | [illegible] | [illegible] | [illegible] | 0 | 0 | 0.03 | [illegible] | [illegible] | [illegible] |
| theophys | 12 | 12 | 12 | 0.00% | 0.00% | 0.00% | 503.15 | 503.16 | 503.15 | -0.00% | 0.00% | 0.00% | 501.14 | 501.13 | 501.14 | 0.00% | 0.00% | -0.00% |
| virgo | 294 | 294 | 293 | 0.00% | 0.00% | 0.34% | 6727.88 | 6727.88 | 6685.93 | 0.00% | 0.00% | 0.63% | 37.48 | 40.59 | 1.44 | -7.66% | -7.66% | [illegible] |

Statistiche giornaliere per il sito INFN-T1 dal 2010-11-19 al 2010-11-21

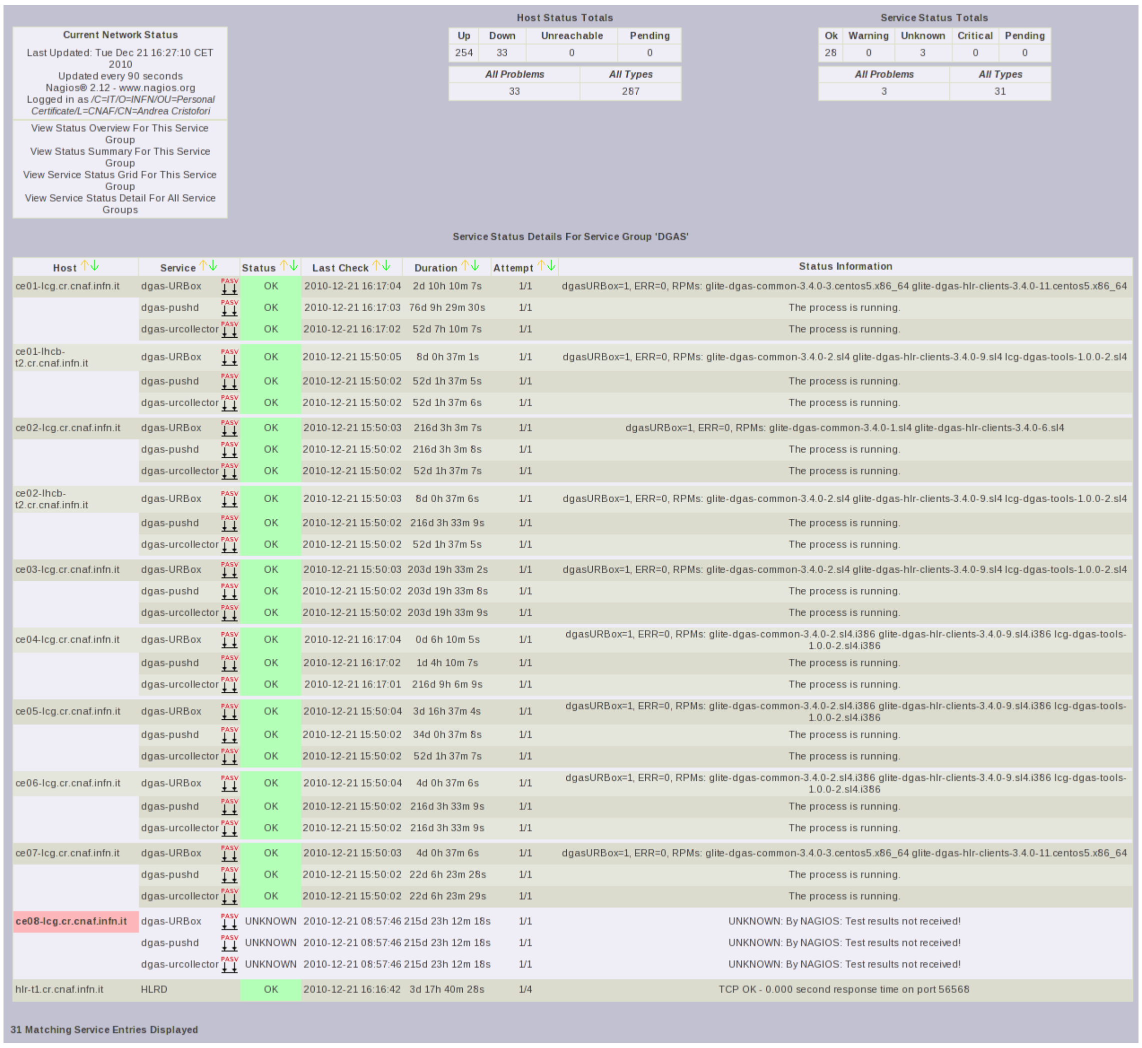
| DATE | VO | HLR JOBS | REDEYE JOBS | BATCH JOBS | DIFF_JOBS HLR_REDEYE | DIFF_JOBS HLR_BATCH | DIFF_JOBS REDEYE_BATCH | HLR WALLTIME | REDEYE WALLTIME | BATCH WALLTIME | DIFF_WALL HLR_REDEYE | DIFF_WALL HLR_BATCH | DIFF_WALL REDEYE_BATCH | HLR CPUTIME | REDEYE CPUTIME | BATCH CPUTIME | DIFF_CPU HLR_REDEYE | DIFF_CPU HLR_BATCH | DIFF_CPU REDEYE_BATCH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2010-11-21 | alice | 3477 | 3482 | 3478 | -0.14% | -0.03% | 0.12% | 39273.17 | 39336.91 | 39273.18 | -0.16% | -0.00% | 0.16% | 18413.12 | 18465.66 | 18413.15 | -0.28% | -0.00% | 0.29% |
| 2010-11-21 | atlas | 10852 | 10854 | 10798 | -0.02% | 0.50% | 0.52% | 7418.97 | 7429.07 | 6784.77 | -0.14% | 9.35% | 9.50% | 5076.50 | 5089.01 | 4676.50 | -0.25% | 8.55% | 8.82% |
| 2010-11-21 | biomed | 13 | 13 | 13 | 0.00% | 0.00% | 0.00% | 134.96 | 134.96 | 134.96 | 0.00% | 0.00% | 0.00% | 0.05 | 0.07 | 0.05 | -28.57% | -4.26% | 34.04% |
| 2010-11-21 | cdf | 1119 | 1119 | 1119 | 0.00% | 0.00% | 0.00% | 12271.58 | 12271.58 | 12271.58 | 0.00% | 0.00% | 0.00% | 7035.07 | 7035.37 | 7035.07 | -0.00% | 0.00% | 0.00% |
| 2010-11-21 | cms | 145 | 144 | 144 | 0.69% | 0.69% | 0.00% | 101.94 | 101.94 | 57.66 | 0.00% | [illegible] | [illegible] | 11.57 | 11.60 | 11.37 | -0.26% | 1.73% | 1.99% |
| 2010-11-21 | cmst1 | 0 | 0 | 1 | [illegible] | [illegible] | [illegible] | 0 | 0 | 44.28 | [illegible] | [illegible] | [illegible] | 0 | 0 | 0.19 | [illegible] | [illegible] | [illegible] |
| 2010-11-21 | lhcb | 429 | 428 | 429 | 0.23% | 0.00% | -0.23% | 364.23 | 364.23 | 364.23 | 0.00% | 0.00% | 0.00% | 154.18 | 154.30 | 154.18 | -0.08% | -0.00% | 0.08% |
| 2010-11-21 | ops | 301 | 301 | 301 | 0.00% | 0.00% | 0.00% | 39.81 | 39.81 | 39.81 | 0.00% | -0.01% | -0.01% | 1.38 | 1.46 | 1.38 | -5.48% | -0.20% | 5.58% |
| 2010-11-21 | patlit | 0 | 0 | 54 | [illegible] | [illegible] | [illegible] | 0 | 0 | 634.20 | [illegible] | [illegible] | [illegible] | 0 | 0 | 399.99 | [illegible] | [illegible] | [illegible] |
| 2010-11-21 | virgo | 67 | 67 | 67 | 0.00% | 0.00% | 0.00% | 1538.87 | 1538.87 | 1538.87 | 0.00% | 0.00% | 0.00% | 0.67 | 0.69 | 0.67 | -2.90% | 0.08% | 3.07% |
| 2010-11-20 | alice | 4695 | 4690 | 4702 | 0.11% | -0.15% | -0.26% | 57447.66 | 57383.92 | 57448.79 | 0.11% | -0.00% | -0.11% | 41722.67 | 41752.00 | 41737.28 | -0.07% | -0.03% | 0.04% |
| 2010-11-20 | argo | 720 | 721 | 1 | -0.14% | 0.00% | [illegible] | 9423.09 | 9433.89 | 0.03 | -0.11% | -6.09% | [illegible] | 5164.16 | 5215.34 | 0.00 | -0.98% | [illegible] | [illegible] |
| 2010-11-20 | atlas | 23861 | 23875 | 23754 | -0.06% | 0.45% | 0.51% | 40254.44 | 40280.20 | 39705.73 | -0.06% | 1.38% | 1.45% | 24823.64 | 24832.81 | 24450.19 | -0.04% | 1.53% | 1.56% |
| 2010-11-20 | biomed | 26 | 26 | 26 | 0.00% | 0.00% | 0.00% | 5.07 | 5.07 | 5.07 | 0.00% | 0.08% | 0.08% | 0.06 | 0.15 | 0.06 | -60.00% | -6.09% | 134.78% |
| 2010-11-20 | cdf | 65 | 69 | 65 | -5.80% | 0.00% | 6.15% | 2451.71 | 2614.48 | 2451.71 | -6.23% | 0.00% | 6.64% | 764.87 | 817.38 | 764.87 | -6.42% | -0.00% | 6.87% |
| 2010-11-20 | cms | 1023 | 1023 | 1021 | 0.00% | 0.20% | 0.20% | 865.79 | 865.79 | 819.73 | 0.00% | 5.62% | 5.62% | 467.65 | 467.93 | 467.44 | -0.06% | 0.05% | 0.11% |
| 2010-11-20 | cmst1 | 0 | 0 | 2 | [illegible] | [illegible] | [illegible] | 0 | 0 | 46.07 | [illegible] | [illegible] | [illegible] | 0 | 0 | 0.21 | [illegible] | [illegible] | [illegible] |
| 2010-11-20 | dteam | 14 | 14 | 14 | 0.00% | 0.00% | 0.00% | 2.22 | 2.22 | 2.22 | 0.00% | -0.20% | -0.20% | 0.00 | 0.01 | 0.00 | -100.00% | -100.00% | 125.00% |
| 2010-11-20 | lhcb | 973 | 973 | 973 | 0.00% | 0.00% | 0.00% | 1420.44 | 1420.44 | 1420.44 | 0.00% | 0.00% | 0.00% | 1067.01 | 1067.31 | 1067.01 | -0.03% | -0.00% | 0.03% |
| 2010-11-20 | ops | 301 | 301 | 301 | 0.00% | 0.00% | 0.00% | 52.20 | 52.20 | 52.20 | 0.00% | -0.00% | -0.00% | 1.14 | 1.21 | 1.14 | -5.79% | 0.22% | 6.37% |
| 2010-11-20 | patlit | 0 | 0 | 107 | [illegible] | [illegible] | [illegible] | 0 | 0 | 548.71 | [illegible] | [illegible] | [illegible] | 0 | 0 | 373.45 | [illegible] | [illegible] | [illegible] |
| 2010-11-20 | t1bio | 0 | 0 | 1 | [illegible] | [illegible] | [illegible] | 0 | 0 | 0.03 | [illegible] | [illegible] | [illegible] | 0 | 0 | 0.01 | [illegible] | [illegible] | [illegible] |
| 2010-11-20 | theophys | 3 | 4 | 3 | [illegible] | 0.00% | [illegible] | 206.23 | 271.16 | 206.23 | [illegible] | -0.00% | [illegible] | 205.67 | 270.38 | 205.67 | [illegible] | 0.00% | [illegible] |
| 2010-11-20 | virgo | 91 | 91 | 91 | 0.00% | 0.00% | 0.00% | 2082.79 | 2082.79 | 2082.79 | 0.00% | 0.00% | 0.00% | 0.37 | 0.91 | 0.37 | -59.34% | 0.53% | 147.25% |

Figure 3.2: *cross-check* web interface[31]

The data vertically shown on the *summary accounting* and *daily accounting* sections is divided in three parts. The first shows the number of jobs, the second the CPU time and the third the Wall Clock Time measured by the three systems. Each group contains six columns where the first three show the real data and the other three the percentage difference compared respectively between HLR and Redeye, HLR and scripts, Redeye and scripts. The only exception is for the values related to the DGAS accounting system because it can distinguish between local and Grid job. In this case the value shown is the sum of the two and, if we move the mouse over this value, the details about the two components, local and grid, that contribute to the sum, appear.

## 3.2 DGAS monitoring

In order to verify that DGAS services are working properly we decided to utilize Nagios to monitor the status of the services on the CEs and on the site HLR. This monitoring tool has been chosen because already used at the INFN-T1 site and on the sites part of the EGI-InSPIRE project as monitor and alarm system. Compared to the checks performed by *cross-check* it only gives information on the status of the daemons on the different servers on the accounting system. It does not give any information about the correctness of data itself but it is nevertheless useful, because it can promptly raise an alarm if some problem arise. After a warning, that can be either sent by email, shown in a web page or both, the site administrator can take actions in order to solve the problem. A dedicated page has been created for the accounting system and the monitored resources.



Current Network Status

Last Updated: Tue Dec 21 16:27:10 CET 2010
Updated every 90 seconds
Nagios® 2.12 - www.nagios.org
Logged in as */C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Andrea Cristofori*

View Status Overview For This Service Group
View Status Summary For This Service Group
View Service Status Grid For This Service Group
View Service Status Detail For All Service Groups

Host Status Totals

| Up | Down | Unreachable | Pending |
|---|---|---|---|
| 254 | 33 | 0 | 0 |

| *All Problems* | *All Types* |
|---|---|
| 33 | 287 |

Service Status Totals

| Ok | Warning | Unknown | Critical | Pending |
|---|---|---|---|---|
| 28 | 0 | 3 | 0 | 0 |

| *All Problems* | *All Types* |
|---|---|
| 3 | 31 |

Service Status Details For Service Group 'DGAS'

| Host | Service | Status | Last Check | Duration | Attempt | Status Information |
|---|---|---|---|---|---|---|
| ce01-lcg.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 16:17:04 | 2d 10h 10m 7s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-3.centos5.x86_64 glite-dgas-hlr-clients-3.4.0-11.centos5.x86_64 |
| | dgas-pushd | OK | 2010-12-21 16:17:03 | 76d 9h 29m 30s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 16:17:02 | 52d 7h 10m 7s | 1/1 | The process is running. |
| ce01-lhcb-t2.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 15:50:05 | 8d 0h 37m 1s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-2.sl4 glite-dgas-hlr-clients-3.4.0-9.sl4 lcg-dgas-tools-1.0.0-2.sl4 |
| | dgas-pushd | OK | 2010-12-21 15:50:02 | 52d 1h 37m 5s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 15:50:02 | 52d 1h 37m 6s | 1/1 | The process is running. |
| ce02-lcg.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 15:50:03 | 216d 3h 3m 7s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-1.sl4 glite-dgas-hlr-clients-3.4.0-6.sl4 |
| | dgas-pushd | OK | 2010-12-21 15:50:02 | 216d 3h 3m 8s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 15:50:02 | 52d 1h 37m 7s | 1/1 | The process is running. |
| ce02-lhcb-t2.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 15:50:03 | 8d 0h 37m 6s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-2.sl4 glite-dgas-hlr-clients-3.4.0-9.sl4 lcg-dgas-tools-1.0.0-2.sl4 |
| | dgas-pushd | OK | 2010-12-21 15:50:02 | 216d 3h 33m 9s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 15:50:02 | 52d 1h 37m 5s | 1/1 | The process is running. |
| ce03-lcg.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 15:50:03 | 203d 19h 33m 2s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-2.sl4 glite-dgas-hlr-clients-3.4.0-9.sl4 lcg-dgas-tools-1.0.0-2.sl4 |
| | dgas-pushd | OK | 2010-12-21 15:50:02 | 203d 19h 33m 8s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 15:50:02 | 203d 19h 33m 9s | 1/1 | The process is running. |
| ce04-lcg.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 16:17:04 | 0d 6h 10m 5s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-2.sl4.i386 glite-dgas-hlr-clients-3.4.0-9.sl4.i386 lcg-dgas-tools-1.0.0-2.sl4.i386 |
| | dgas-pushd | OK | 2010-12-21 16:17:02 | 1d 4h 10m 7s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 16:17:01 | 216d 9h 6m 9s | 1/1 | The process is running. |
| ce05-lcg.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 15:50:04 | 3d 16h 37m 4s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-2.sl4.i386 glite-dgas-hlr-clients-3.4.0-9.sl4.i386 lcg-dgas-tools-1.0.0-2.sl4.i386 |
| | dgas-pushd | OK | 2010-12-21 15:50:02 | 34d 0h 37m 8s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 15:50:02 | 52d 1h 37m 7s | 1/1 | The process is running. |
| ce06-lcg.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 15:50:04 | 4d 0h 37m 6s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-2.sl4.i386 glite-dgas-hlr-clients-3.4.0-9.sl4.i386 lcg-dgas-tools-1.0.0-2.sl4.i386 |
| | dgas-pushd | OK | 2010-12-21 15:50:02 | 216d 3h 33m 9s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 15:50:02 | 216d 3h 33m 9s | 1/1 | The process is running. |
| ce07-lcg.cr.cnaf.infn.it | dgas-URBox | OK | 2010-12-21 15:50:03 | 4d 0h 37m 6s | 1/1 | dgasURBox=1, ERR=0, RPMs: glite-dgas-common-3.4.0-3.centos5.x86_64 glite-dgas-hlr-clients-3.4.0-11.centos5.x86_64 |
| | dgas-pushd | OK | 2010-12-21 15:50:02 | 22d 6h 23m 28s | 1/1 | The process is running. |
| | dgas-urcollector | OK | 2010-12-21 15:50:02 | 22d 6h 23m 29s | 1/1 | The process is running. |
| **ce08-lcg.cr.cnaf.infn.it** | dgas-URBox | UNKNOWN | 2010-12-21 08:57:46 | 215d 23h 12m 18s | 1/1 | UNKNOWN: By NAGIOS: Test results not received! |
| | dgas-pushd | UNKNOWN | 2010-12-21 08:57:46 | 215d 23h 12m 18s | 1/1 | UNKNOWN: By NAGIOS: Test results not received! |
| | dgas-urcollector | UNKNOWN | 2010-12-21 08:57:46 | 215d 23h 12m 18s | 1/1 | UNKNOWN: By NAGIOS: Test results not received! |
| hlr-t1.cr.cnaf.infn.it | HLRD | OK | 2010-12-21 16:16:42 | 3d 17h 40m 28s | 1/4 | TCP OK - 0.000 second response time on port 56568 |

31 Matching Service Entries Displayed

Figure 3.3: Service Status Details For Service Group DGAS[32]

The services monitored are those installed in the CEs and in the site HLR. All the checks are passive and performed with scripts executed regularly on the machine that hosts the service. In Figure 3.3 is shown a screen shot of the page described. For each CE it shows the number of URs created that have not yet been transferred to the HLR, the number of those in the error directory and the sensors version. On the section dedicated to the HLR are shown the packages version and the service status. With this single page is possible to have an overview on the status of all the services related to the DGAS accounting system. A warning is raised if the number of records not transferred from the CE to the HLR is higher than 100 and an error if this number is higher than 1000. An alert is also shown if the HLR or any of the other services is not active. In conjunction with the cross-check tool, we can therefore obtain a system that checks the availability with the Nagios tests and its reliability with the cross-check tool.

# Chapter 4

# Accounting of storage resources

*"Standards are always out of date. That's what makes them standards."*
Alan Bennett

## 4.1 Characteristics of the storage accounting

As we have introduced in Chapter 2, an accounting system can be defined as a collection of applications that allows the retrieval, the recording of the used resources and the interfaces necessary for billing, pricing or further analysis. It can also be possible to aggregate URs using different criteria, e.g.: users, groups, VOs, etc. The same general definition is valid for storage accounting. In the Grid environment, we can identify some of the values related to the metering for the specific resources that are necessary to account. If we consider the CPU accounting, we could use the Wall Clock Time (WCT) and the CPU time (CPT). Other metrics might be also useful to identify the user associated with each UR and the VO whom the user belongs to. In Chapter 2, we have seen how the problematic related to computational resources have been solved on the DGAS accounting system. We will now analyze some of the existing storage accounting systems.

## 4.2 Existing storage accounting systems

### 4.2.1 GridPP Storage Accounting System

The approach of the GridPP accounting system is to regularly query a top level BDII to collect information about the available and used storage for the different

VOs related to the SEs that publish their data on that particular BDII. In particular the GlueSEUniqueID, the attribute defined in the GLUE schema for the Unique Identifier of the SE[33], is parsed and the information are collected in a MySQL database[34]. In Figure 4.1 a screen shot of the visualization page for this system and the attributes recorded on Table 4.1 are shown.

| **Column name** | **Type** | **Primary key** | **Can be NULL** |
|---|---|---|---|
| RecordIdentity | VARCHAR(255) | Yes | No |
| ResourceIdentity | VARCHAR(255) | No | Yes |
| Grid | VARCHAR(50) | No | Yes |
| ExecutingSite | VARCHAR(50) | No | Yes |
| VO | VARCHAR(50) | No | Yes |
| SpaceUsed | INTEGER | No | Yes |
| SpaceAvailable | INTEGER | No | Yes |
| Total | INTEGER | No | Yes |
| Unit | VARCHAR(50) | No | Yes |
| SEArchitecture | VARCHAR(50) | No | Yes |
| Type | VARCHAR(50) | No | Yes |
| srmType | VARCHAR(50) | No | No |
| EventDate | DATE | No | No |
| EventTime | TIME | No | No |
| MeasurementDate | DATE | No | No |
| MeasurementTime | TIME | No | No |

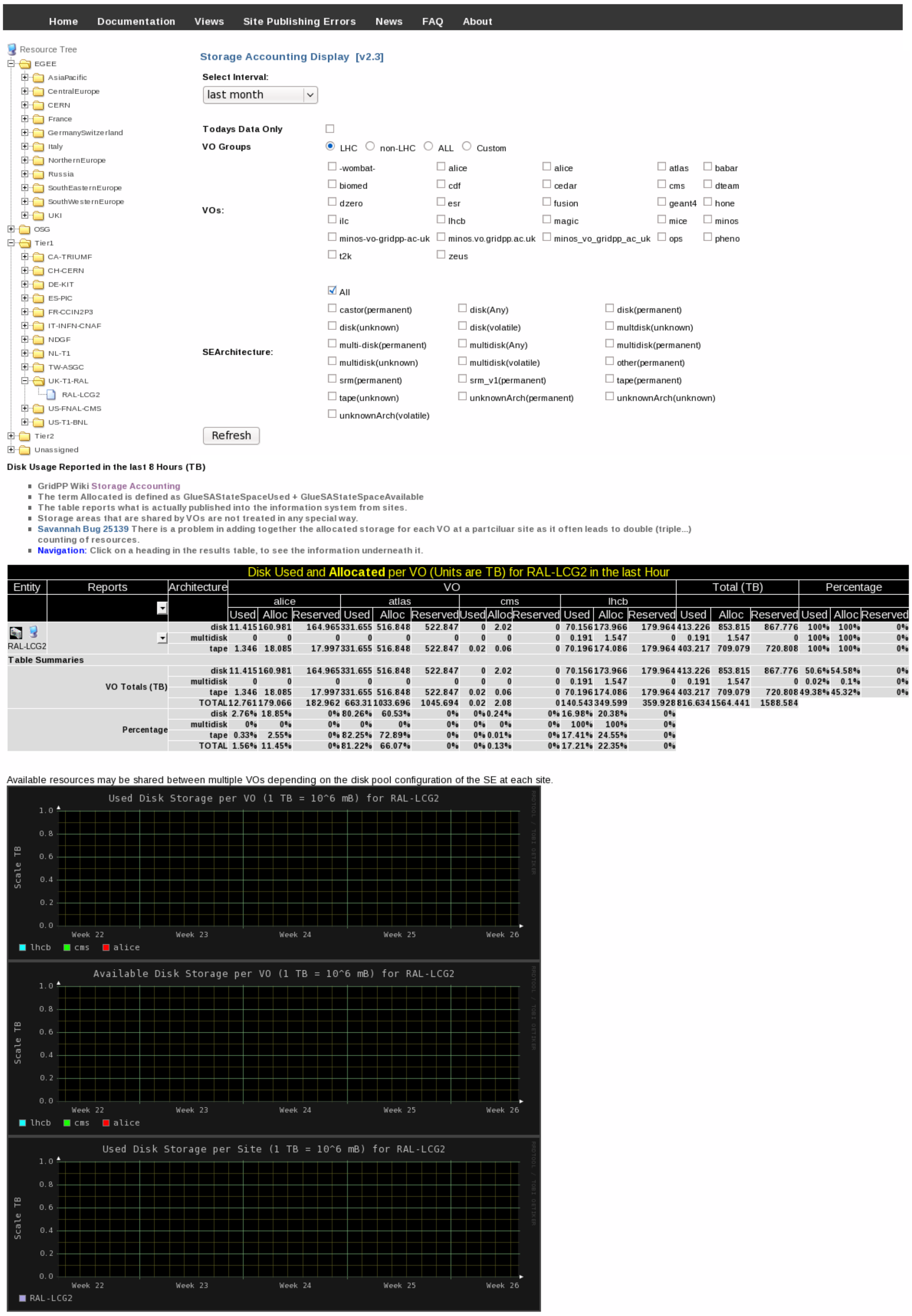
Table 4.1: GridPP Storage Accounting System: recorded values

**Advantages**:

- this approach allows the collection of the storage accounting information transparently. The information are collected using a top level BDII. For this reason, each well configured site already publishes all the information needed by this system;
- no need for the installation of any additional package or sensor.

**Disadvantages**:

- everything depends on the correct configuration of the site and top level BDII. If something in the chain is not working the information can not be retrieved;



**Storage Accounting Display [v2.3]**

**Select Interval:** last month

**Todays Data Only** ☐

**VO Groups** [✓ LHC] non-LHC ALL Custom

**VOs:** ☐ -wombat- ☐ alice ☐ alice ☐ atlas ☐ babar ☐ biomed ☐ cdf ☐ cedar ☐ cms ☐ dteam ☐ dzero ☐ esr ☐ fusion ☐ geant4 ☐ hone ☐ ilc ☐ lhcb ☐ magic ☐ mice ☐ minos ☐ minos-vo-gridpp-ac-uk ☐ minos.vo.gridpp.ac.uk ☐ minos_vo_gridpp_ac_uk ☐ ops ☐ pheno ☐ t2k ☐ zeus

**SEArchitecture:** [✓ All] ☐ castor(permanent) ☐ disk(Any) ☐ disk(permanent) ☐ disk(unknown) ☐ disk(volatile) ☐ multdisk(unknown) ☐ multi-disk(permanent) ☐ multidisk(Any) ☐ multidisk(permanent) ☐ multidisk(unknown) ☐ multidisk(volatile) ☐ other(permanent) ☐ srm(permanent) ☐ srm_v1(permanent) ☐ tape(permanent) ☐ tape(unknown) ☐ unknownArch(permanent) ☐ unknownArch(unknown) ☐ unknownArch(volatile)

**Disk Usage Reported in the last 8 Hours (TB)**

- GridPP Wiki Storage Accounting
- The term Allocated is defined as GlueSAStateSpaceUsed + GlueSAStateSpaceAvailable
- The table reports what is actually published into the information system from sites.
- Storage areas that are shared by VOs are not treated in any special way.
- Savannah Bug 25139 There is a problem in adding together the allocated storage for each VO at a partciluar site as it often leads to double (triple...) counting of resources.
- **Navigation:** Click on a heading in the results table, to see the information underneath it.

Disk Used and **Allocated** per VO (Units are TB) for RAL-LCG2 in the last Hour

| Entity | Reports | Architecture | VO | | | | | | | | | | | | Total (TB) | | | Percentage | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | alice | | | atlas | | | cms | | | lhcb | | | | | | | | |
| | | | Used | Alloc | Reserved | Used | Alloc | Reserved | Used | Alloc | Reserved | Used | Alloc | Reserved | Used | Alloc | Reserved | Used | Alloc | Reserved |
| RAL-LCG2 | | disk | 11.415 | 160.981 | 164.965 | 331.655 | 516.848 | 522.847 | 0 | 2.02 | 0 | 70.156 | 173.966 | 179.964 | 413.226 | 853.815 | 867.776 | 100% | 100% | 0% |
| | | multidisk | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.191 | 1.547 | 0 | 0.191 | 1.547 | 0 | 100% | 100% | 0% |
| | | tape | 1.346 | 18.085 | 17.997 | 331.655 | 516.848 | 522.847 | 0.02 | 0.06 | 0 | 70.196 | 174.086 | 179.964 | 403.217 | 709.079 | 720.808 | 100% | 100% | 0% |
| **Table Summaries** | | | | | | | | | | | | | | | | | | | | |
| | VO Totals (TB) | disk | 11.415 | 160.981 | 164.965 | 331.655 | 516.848 | 522.847 | 0 | 2.02 | 0 | 70.156 | 173.966 | 179.964 | 413.226 | 853.815 | 867.776 | 50.6% | 54.58% | 0% |
| | | multidisk | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.191 | 1.547 | 0 | 0.191 | 1.547 | 0 | 0.02% | 0.1% | 0% |
| | | tape | 1.346 | 18.085 | 17.997 | 331.655 | 516.848 | 522.847 | 0.02 | 0.06 | 0 | 70.196 | 174.086 | 179.964 | 403.217 | 709.079 | 720.808 | 49.38% | 45.32% | 0% |
| | | TOTAL | 12.761 | 179.066 | 182.962 | 663.31 | 1033.696 | 1045.694 | 0.02 | 2.08 | 0 | 140.543 | 349.599 | 359.928 | 816.634 | 1564.441 | 1588.584 | | | |
| | Percentage | disk | 2.76% | 18.85% | 0% | 80.26% | 60.53% | 0% | 0% | 0.24% | 0% | 16.98% | 20.38% | 0% | | | | | | |
| | | multidisk | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 100% | 100% | 0% | | | | | | |
| | | tape | 0.33% | 2.55% | 0% | 82.25% | 72.89% | 0% | 0% | 0.01% | 0% | 17.41% | 24.55% | 0% | | | | | | |
| | | TOTAL | 1.56% | 11.45% | 0% | 81.22% | 66.07% | 0% | 0% | 0.13% | 0% | 17.21% | 22.35% | 0% | | | | | | |

Available resources may be shared between multiple VOs depending on the disk pool configuration of the SE at each site.

Figure 4.1: GridPP storage accounting web interface

- there is not any information between two different checks. If the BDII is not available for this period of time no information can be collected and no information can be recovered;
- no information can be recovered also in the case that the accounting system can not query the top BDII.

### 4.2.2 HLRmon

HLRmon has been developed as a tool to provide a graphic interface for the accounting of computational resources collected with DGAS. It has been then extended to allow some storage accounting capabilities. Every site that wants to monitor its accounting capabilities, needs to install a sensor that collects information related to the storage. Those records contain information on the VO, available disk space and used disk space. They are written by the sensor in a XML file sent via email to a proper account and stored in the HLRmon database. The information are collected once a day and a cronjob process those data and produces the reports. An example of the file, in XML format, is shown below while Figure 4.2 is a screen shot of the visualization reports produced by the tool.

```
<?xml version='1.0' standalone='yes'?>
<site name="INFN-NAPOLI-ATLAS">
  <SE name="t2-dpm-01.na.infn.it" timestamp="1245916801">
     <VO name="atlas">
        <class name="DATADISK">
        <free>45739683715481</free>
        <used>14733455812198</used>
     </class>
        <class name="MCDISK">
        <free>13798870928588</free>
        <used>41176710460211</used>
     </class>
        <class name="PRODDISK">
        <free>4837851162214</free>
        <used>659706976665</used>
    </class>
      <class name="USERDISK">
      <free>990279872020</free>
      <used>109231755755</used>
    </class>
      <class name="GROUPDISK">
      <free>1099511627776</free>
      <used>0</used>
    </class>
      <class name="LOCALGROUPDISK">
      <free>2122057441607</free>
      <used>11072082091704</used>
    </class>
      <class name="CALIBDISK">
      <free>14502558370365</free>
      <used>7487674185154</used>
    </class>
      <class name="SCRATCHDISK">
      <free>5310641162158</free>
      <used>186916976721</used>
    </class>
      <class name="other">
      <free>40549859983360</free>
      <used>12833362280455</used>
    </class>
      <class name="total">
      <free>128951314263569</free>
      <used>88259140538863</used>
    </class>
    </VO>
    <free>128951314263569</free>
    <size>217210454802432</size>
  </SE>
  <date>Thu Jun 25 10:00:01 CEST 2009</date>
</site>
```

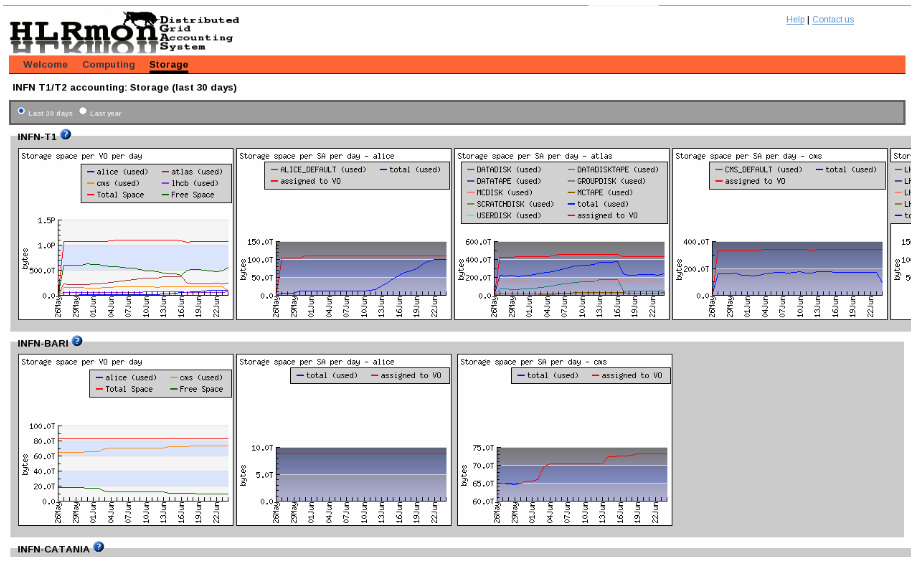


Figure 4.2: HLRmon storage accounting web interface[35]

This tool is currently used by Italian NGI, Italian Grid Initiative (IGI). However it needs further development because it does not provide the needed level of detail and it does not provide a standard UR.

### 4.2.3 SAGE

Storage Accounting for Grid Environments's (SAGE) approach to the storage accounting is quite different from the two described earlier. It consists, basically of two parts:

- a sensor that is installed in each SE available (at the time of writing only DPM is supported);
- a database where the collected information is stored.

Information about user's activity is composed essentially by:

- time stamp of when an activity started and terminated;
- the type of the activity (e.g.: the file has been stored, retrieved, etc);

- the subject of the grid user's certificate;
- the name of the file accessed;
- the amount of bytes added/removed, if any;
- the protocol used to access the files on the SE;
- the VO which the user belong to[36].

SAGE defines a function called *Disk Energy* as 'the disk energy consumed by a file saved on a storage device is given by the integral of its size in time'[36].

**Advantages**:

- it is a true accounting system that can store information of the activities that are made on the storage;
- the log files produced by the SE can be reprocessed in case of problems.

**Disadvantages**:

- only available for DPM;
- development has been discontinued.

### 4.2.4 Amazon S3

Amazon Simple Storage Service (Amazon S3) service offers an online storage system. The details of the implementation are not public, but it is interesting to describe here because of the metrics considered and its billing system that can be used as a reference. Its purpose is to charge the user depending on the amount of data and the access to those data. The GBs of storage billed in a month is the average storage used throughout that month. This includes all object data and meta data stored in buckets that the user created under his account. Amazon S3 is intentionally built with a minimal feature set:

- write, read, and delete objects containing from 1 byte to 5 gigabytes of data each. The number of objects the user can store is unlimited;
- each object is stored in a bucket and retrieved via a unique, developer assigned key
- authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users;
- uses standards-based Representational State Transfer (REST) and SOAP interfaces designed to work with any Internet-development toolkit;

- built to be flexible so that protocol or functional layers can easily be added. Default download protocol is HTTP but a BitTorrent$^{TM}$[1] interface is also available. This protocol, when applied to Amazon S3, lower costs for high-scale distribution. Additional interfaces will be added in the future.

Data transfer ”in” and ”out” refers to transfers into and out of an Amazon S3 location (e.g.: US or EU). Data transferred within an Amazon S3 location via a COPY request is free of charge. Data transferred via a COPY request between locations is charged at regular rates. Data transferred between Amazon Elastic Compute Cloud (Amazon EC2), which is a web service that provides resizable compute capacity in the cloud and is designed to make web-scale computing easier for developers[38], and Amazon S3 within the same region is free of charge. Data transferred between Amazon EC2 and Amazon S3 across regions (e.g.: between US and EU), will be charged at Internet Data Transfer rates on both sides of the transfer. Storage and bandwidth size includes all file overhead[39].

**Advantages**:

- S3 offer a complete system for accounting and billing.

**Disadvantages**:

- it is a proprietary system. API are available for accessing the storage but the system itself is not available nor it is open;

- it is not possible to dig into the details like user, groups, file.

## 4.3 A proposal for a new system

With the exceptions of SAGE and Amazon S3, most of the described systems can not be defined as real storage accounting system but more as monitoring systems. The general problem, due to the kind of approach is that they tend to present a static view of the status of the storage in a precise time. No information is usually available between two different checks. Due to the nature of the source of information it is not possible to recover old data, because of the limitation of the systems considered, a new approach has been devised. The goal is to define a reliable system that can record all the events on the files stored on a SE. From those events, it must be possible to efficiently create URs that can be used for the storage accounting. To do that a new definition of UR has been devised and tested during the thesis work at INFN-CNAF.

[1]BitTorrent$^{TM}$ is a protocol for distributing files. It identifies content by URL and is designed to integrate seamlessly with the web. Its advantage over plain HTTP is that when multiple downloads of the same file happen concurrently, the downloaders upload to each other, making it possible for the file source to support very large numbers of downloaders with only a modest increase in its load[37].

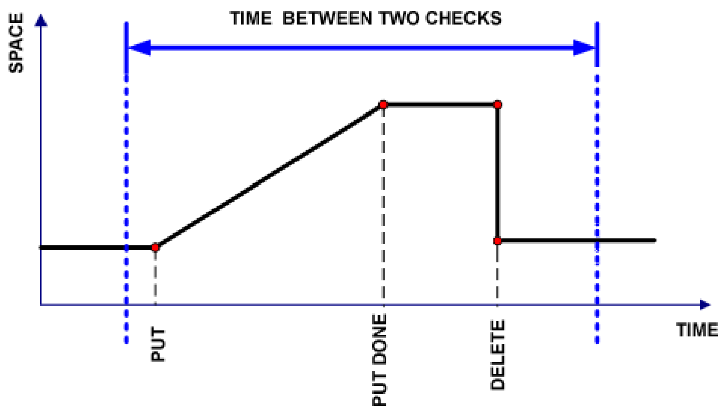


Figure 4.3: Example of a file created and deleted but not reported by most of the accounting systems

In Figure 4.3, a possible problem that might arise using the systems described is presented: a file that is written just after one check, and is then deleted just before another one does not leave any record of its presence in an accounting system based on regular scans or checks of the total, used and available storage space. One solution to this problem could be to increase the frequency of scans on the file system to a reasonably level, e.g.: if we want to be sure to spot a user that stores files of 10 GB, and the transfer rate is 1 Gb/s, we should check the file system at least every 80 seconds or less. Files produced in High Energy Physics (HEP) experiments are often written once, read several times and kept on storage for long periods. For those kind of applications the approach described could be sufficient and the frequency of the checks could be reduced. But in a more generalized environment it is not possible to exclude that files have a shorter life time. In those cases, the regular checks of the disk is not a viable solution because it would require a scan of the entire storage every few seconds especially if files have a smaller size. Another problem related to this system is that it does not provide any information about the owner of the files present in the storage. Information are limited to the VO and Storage Area (SA), a logical portion of storage assigned to a VO. In future versions of StoRM, Extended Attributed (EA), supported on ext3[40], XFS[41], General Parallel File System[42] (GPFS) and Lustre[43], will be used to store user defined attribute for each file or directory like, e.g.: the owner of the file. For file systems that support EA, and for field of research like HEP that store files for long periods of time, this could be a viable solution to solve the problem of retrieving the information necessary for the storage accounting. During the scheduled checks of the disk space it would be possible to read also those extra attributes and collect information on the user owning the file. However this solution is not suitable for all the use cases especially when the lifetime of the files is shorter than the periodic check. To obtain more precise data on the used resources, the events related to operation in a file should be recorded and collected for further analysis as soon as they are produced. When a request for an operation is made to the storage system, this request should be logged immediately. In this way, it is possible to record all the events and calculate the effective amount of data stored and the period. In Figure 4.3 we can see the three moment that should be recorded. When the client start to copy the file *PUT*, when the file is successfully copied to the storage *PUT DONE* and when the file is finally deleted *DELETE*.

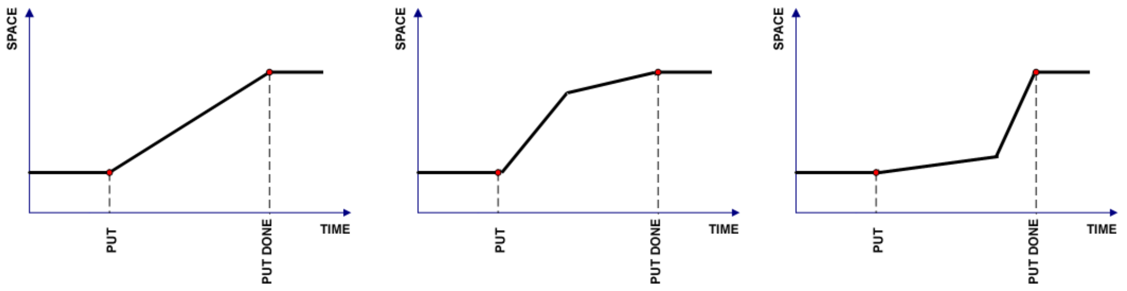


Figure 4.4: Time and network dependent way in which a file might be uploaded on a SE

On the other hand, as we can see in Figure 4.4, we know only the moment when the *PUT* and the *PUT DONE* are recorded. No information can be retrieved in the way the file is transferred. As a consequence, the storage utilized during this interval of time, can not be precisely determined. For this reason a good approximation could be to assume that, the speed of the copy of a file to a storage system is constant like the one shown in the left part of Figure 4.4. This would be the only way to account for the space utilized during this process. The center and right part of 4.4 show what could happen if the transfer speed is variable. The storage utilization between the *PUT*, *PUT DONE* and the *DELETE*, using SAGE's definition, is the disk energy: 'the energy consumed by a file saved on a storage device is given by the integral of its size in time'[36]. This allows to simply calculate the energy utilized by any user, group of users or VOs. Figure 4.5 shows an example of the disk energy utilized by a file stored for a certain amount of time.

As opposed to CPU accounting where a record that describes the job executed is produced at the end of each job, that has a maximum time allowed defined, usually in the batch system data can stored for an unpredictable amount of time. For practical reasons, it is not possible to wait that the file is deleted to produce a UR. If a file is transferred to a storage device and left there for years, the UR would be produced at the end of this period. In the schema devised we can define three records R1, R2 and R3:

- record that describes the operations on a file is recorded in a data base. The events *PUT DONE*, *READ*, *DELETE* form type R1 record;

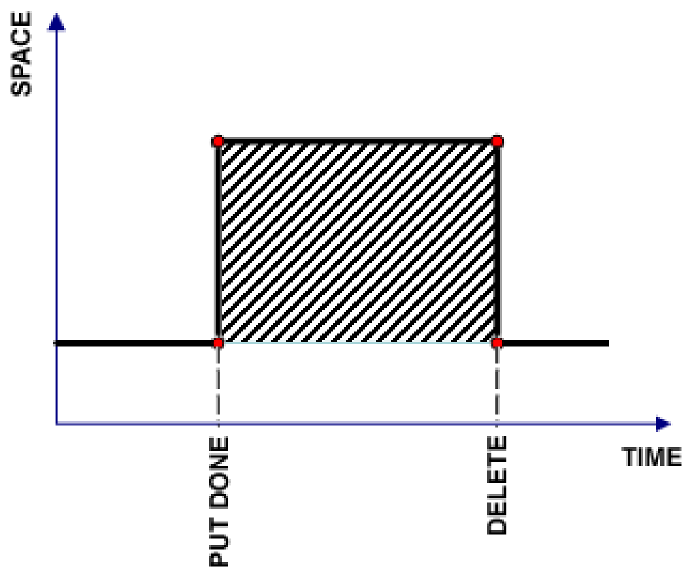


Figure 4.5: The area represents the disk energy required to store a file

- every fixed, but configurable amount of time T1 (e.g.: every week or every day), records R1 produced in the previous T1 interval are analyzed and a record that describe every file and its disk energy is produced. We can call this type of record R2;
- every fixed, but configurable amount of time T2 (e.g.: every month), records R2 are analyzed and summarized in a new record R3 that can have the same information of R2 but for a longer interval of time. In this phase, the information related to the single file can be lost if the aggregation is made also on the files of a single user or over all the files of a single VO. In this case, only a per user or per VO information will be present in the new records R3;
- records R2 are kept for a configurable amount of time after being processed. This would allow the site managers to reprocess data for previous periods or to check the correctness of the collected information if necessary. Record R1 should never be deleted but archived only after a *DELETE* event is present for a corresponding *PUT DONE*. If necessary *READ* operations can be archived even if a corresponding *DELETE* for that file is not present;
- If a file is still present, R1 records related to the time of creation of this file should be kept for all the life of the file. Keeping those records allows to easily generate records R2 and consequently R3.

Keeping different level of information will allow the site managers and the users to track their own records with a decreasing level of detail when moving to older data. It will be useful to use this approach because old records are not useful after a certain amount of time and they should be kept to a minimum to avoid, or at least reduce the possibility that a huge amount of records, corresponding to each operation on the storage system fills the database e.g.: a file being written or deleted. The billing of the storage utilized could be made every month. This would require at least one month of old record R2 to show precisely the files that the user is being charged for. To maintain historical information, record R2 can be deleted after three month and the only information left is record R3. This system could be tuned depending on the necessity of the site. If a site does not produce a lot of record R1 it could decide to keep every thing along with record R2 and R3. As described before files copied to the storage remain until they are deleted. There is an exception: files can have an expiration date. After the expiration date is reached, the file is not guaranteed to be available. The accounting system should consider the expiration date as the end of the billing period for this file, if it is not deleted before. The user could still access it but there are no guarantees. In this case, records R1 with a *READ* event should be produced. It consists of two agents, one collects the SRM published information and one stores them in the accounting database. The front-end of the system is a web interface, capable of dynamically generate the plots required in each view. As for the CPU accounting the three steps necessary for the accounting process are UR generation, UR distribution and presentation. In the following section I will describe in more detail the solution proposed to extend the usage record and collect the required information from the SEs.

### 4.3.1 UR Generation

In the computing scenario a job is submitted, executed and terminated. Its life cycle is well defined and bounded in time thus a UR for computing purpose should give information on the job life cycle, the time taken to execute, etc. In terms of storage resources, things are more complicated: a file is stored somewhere by someone in a certain instant, can be retrieved multiple times and can remain there for undetermined amount of time. We can describe its life cycle, but we can not wait for its termination before making any related UR. This implies that we have to account continuously in time while the file still exists and intercept all activities on that file. For those reasons we can define two different types of accounting information:

- accounting of user access to files;
- accounting of disk space.

The main source of information to make accounting of disk space is the information published by the SE in the IS, following the GLUE Schema described in Section 5.2, Chapter 5. To address this type of accounting, a specific SE agent, running on the site HLR, should be used:

- **InfoAgent**: the InfoAgent is in charge of querying the Grid Information System about the information of SEs registered in each of the GRID sites. This information is then stored in the local accounting database. With the information collected by the agent, the disk space occupied at each site, for each SA and VO can be calculated. All GLUE attribute that will be described in section 5.2 need to be collected. At the moment we consider the *reserved space* as busy and therefore accounted to the VO.

To make accounting of the user activities, we have to intercept the file related events like creation, deletion and reading. Those kind of operations can be made both in a Grid environment and locally. Generally, a mixed scenario is used. For this kind of accounting we need two agents: one agent will be specific to the SE architecture used, the second is needed to send the UR generated to the site HLR.

- **SpecificSEAgent**: The SpecificSEAgent is in charge of creating the R1 EventRecord previously described. This agent should run on the SE and it is highly dependent of the SE technology used. It collects all the information needed by parsing log files or querying SE specific database and merging the requested data in one file for each *event* which represents a R1 record. If a file has a life time set, the time of presence accounted on the storage is the lifetime.

Record for this storage accounting is represented in Table 4.2.

### 4.3.2 Distribution

The command *SendRecord* is used to distribute R1 records to the site HLR. On the HLR side, the getRecordEngine instantiated by the HLR server provides a predefined storage records table, which is suitable for User and VO based accounting.

### 4.3.3 Presentation

A web interface is necessary for the presentation of the collected data. It should be able to show different views of the storage accounting data according to the role of the users.

### 4.3.4 Working description

The storage accounting system should be able to:

- collect information on the usage of the storage space for a given SE;
- provide information to users on the status of their files and how much they are accessing and using that resource;

| **Attribute name** | **Type** | **Primary key** | **Can be NULL** | **Note** |
|---|---|---|---|---|
| RecordIdentity | VARCHAR(64) | Yes | No | a unique identifier for the UR |
| GlobalFileId | VARCHAR (512) | No | Yes | LFC file ID |
| LocalFileId | VARCHAR (512) | No | Yes | full path and file name on the storage or SURL |
| GlobalGroup | VARCHAR(64) | No | Yes | As reported in the VOMS extension, or it should be deduced in other way by the specific SE agent in GRID or another grouping mechanism in other cases |
| GlobalUsername | VARCHAR(128) | No | Yes | If the user DN is missing (local account), it contain the userId |
| LocalUserId | VARCHAR(64) | No | Yes | Local user ID |
| Charge | FLOAT | No | Yes | the charge associated with the storage utilization |
| Status | char(64) | No | Yes | status of a request to the storage system |
| Host | VARCHAR(64) | No | Yes | GlueSEName |
| SubmitHost | VARCHAR(64) | No | Yes | submitting host |
| ProjectName | VARCHAR(64) | No | Yes | It can be the user guid (local account) or the group of the VO as reported in the VOMS extensions |
| ProjectPartition | VARCHAR(64) | No | Yes | GlueSAName |
| StorageType | VARCHAR(64) | No | Yes | StoRM, dCache, etc. |
| ProtocolType | VARCHAR(64) | No | Yes | The protocol used in the event: PutDone, Delete, Read, etc. |
| OperationType | VARCHAR(64) | No | Yes | The operation performed |
| Network | INT(10) | No | Yes | The network utilization for the operation |

| | | | | |
|---|---|---|---|---|
| Disk | INT(10) | No | Yes | CREATE: total file size; DELETE: size of the deleted file; READ: the quantity of file read; for aggregate the storage utilization |
| TimeDuration | INT(10) | No | Yes | the time used for the operation of for aggregate the amount of time that the storage has been used |
| TimeInstant | INT(10) | No | Yes | the timestamp of the event |
| ServiceLevel | VARCHAR(64) | No | Yes | persistent, volatile, etc. |

Table 4.2: Record attributes description

- let site administrators, VO managers, Regional Operation Center (ROC) managers monitor disk space usage.

When the system starts, it can account for user activities and disk space asynchronously. If the accounting system starts on an empty SE, the amount of disk space used collected by the SpecificSEAgents for the user activities and the one published in the IS should be the same. Otherwise, the disk space used, published in the IS might be greater than the one collected by monitoring the user activities. At the moment, the user activities done before the starting of the accounting system is an open issue. Probably it will not possible to collect old activities for all SE architectures. Accounting of user activities and accounting of disk space will be permanently stored in the local site HLR repository. A web interface, like HLRMon, will be used to show different views of the storage accounting data according with the role of the users.

### 4.3.5 Extension of the Usage Record

The UR - Format Recommendation (UR-FR) contains tags that can be used, with the same meaning or with small modifications, in a UR for the storage accounting. Following is a proposal for the definition of the new properties needed for a UR for storage accounting.

#### Property ***RecordIdentity***

A record identity uniquely defines a record in the set of all usage record for the grid implementation.

- This property SHOULD be referred as recordidentity.
- This property MUST contain data of type string.
- This property MUST exist.
- This property MUST be unique.
- Meta-properties:
  - Create time of the record MUST be specified.

#### Property ***GlobalFileId***

The global file identifier as assigned by a system such as LFC.

- This property SHOULD be referred as globalfileid.
- This property MUST have data of type string.
- This property is optional (to account the time a file has been stored in a SE this propriety MUST be specified to have an association between the PUT END and the DELETE. For other operation, like a GET or a DELETE this is not necessary).
- Meta-properties:
  - Create time of the record MUST be specified.
  - Description MAY be specified.

#### Property ***LocalFileId***

The local file identifier as assigned by the local storage system.

- This property SHOULD be referred as localfileid.
- This property MUST have data of type string.
- This property is optional.
- Meta-properties:
  - Description MAY be specified.

### Property *GlobalGroup*

The global group identifies a way to group, e.g. different URs from different storage systems that are produced in the same site. It can be reported in the VOMS extension, or it cab be deduced in other way or configured in the sensors.

- This property SHOULD be referred as globalgroup.
- This property MUST have data of type string.
- This property is optional.

### Property *LocalUserId*

The local identity of the user associated with the resource consumption reported in the UR. This user is often referred to as the requesting user, e.g.: the value might be the user's login name corresponding to the user's uid in */etc/passwd* file in Unix systems.

- This property SHOULD be referred as localuserid.
- This property MUST have data of type string.
- This property is optional.

### Property *GlobalUsername*

The global identity of the user associated with the resource consumption reported in this UR. For example, the value may be the Distinguished Name (DN) from the user's x.509 certificate.

- This property SHOULD be referred as globaluserid.
- This property MUST have data of type string.
- This property is optional.

### Property *Charge*

This property represent the total charge of the job in the system's allocation unit. The meaning of this charge will be site dependent. The value for this property MAY include premiums or discounts assessed on the actual usage represented within this record. Therefore, the reported charge might not be directly reconstructable from the specific usage reported. Note that "Charge" does not necessarily refer to a currency-based unit unless that is what members on the Grid VO agree to as definition. If charge denotes a value in currency, standard currency codes should be used to indicate the currency being reported.

- This property SHOULD be referred as charge.
- This property MUST have data of type float.
- This property is optional.
- Meta-properties:
  - Units MAY be specified.
  - Description MAY be specified.
  - Formula MAY be specified that describes how the charge was arrived at. There is no requirement format for the formula.

#### Property *Status*

As defined in the UR-FR document, it may represent *the exit status of an interactive running process or the exit status from the batch queuing system's accounting record.* In the case of Storage Accounting this definition should be extended to include *the status of a request to the Storage system.*

- This property SHOULD be referred as status.
- This property MUST contain data of type string.
- This property MUST exist.
- As in the UR-FR *Status* MUST support at least the following value with the corresponding meaning:
  - aborted - A policy or human intervention caused the operation (PUT or READ);
  - completed - The operation completed correctly;
  - failed - Operation failed without external intervention;
  - started - The required operation started on the specified moment.
- This property MAY support other values, as agreed upon within the implementation context. The UR-FR reccomendation already list other values for this property.

#### Property *Host*

A descriptive name of the SE on which the file has been stored. This should be a unique identifier. In Grid environment it could represent the Fully Qualified Domain Name (FQDN) of the SE.

- This property SHOULD be referred as machine name.

- This property MUST contain data of type domain name.
- This property MUST exist (in the UR-FR Draft this it is defined as optional but, in the case of storage accounting, this information is necessary).
- Meta-properties:
  - Description MAY be specified.

#### Property *SubmitHost*

The system hostname from which the file has been submitted, retrieved or deleted.

- This property SHOULD be referred as submithost.
- This property MUST contain data of type of domain name.
- This property SHOULD exist (in the UR-FR Draft this it is defined as optional but, in the case of storage accounting, this information should be present at least when a per file accounting is used; for aggregate records it can be omitted).
- Meta-properties:
  - Description MAY be specified.

#### Property *ProjectName*

The project associated with the resource usage reported with this record. In Grid environment it could be the VO of the user consuming the resources.

- This property SHOULD be referred as projectname.
- This property MAY contain data of type string.
- This property is optional.
- Meta-properties:
  - Description MAY be specified.

### Property *ProjectPartition*

The project's partition associated with the resource usage reported with this record. In Grid environment could be the user SA associated to the VO of the user consuming the resources.

- This property SHOULD be referred as projectpartition.
- This property MAY contain data of type string.
- This property is optional.
- Meta-properties:
  - Description MAY be specified.

### Property *StorageType*

Is the technology used for the storage system. In Grid it could be, e.g.: StoRM, dCache, etc.

- This property SHOULD be referred as storagetype.
- This property MUST have data of type string.
- This property is optional.

### Property *ProtocolType*

It is the protocol used for the single operation.

- This property SHOULD be referred as protocoltype.
- This property MUST have data of type string.
- This property is optional.

### Property *OperationType*

It is the operation performed for the single event: PutDone, Delete, Read, etc.

- This property SHOULD be referred as operationtype.
- This property MUST have data of type string.
- This property is optional.

**Property *Network***

The amount of transferred data, protocol, network characteristics used by the job.

- This property SHOULD be referred as network.
- This property MUST contain data of type positive integer.
- This property is optional.
- Meta-properties:
  - Units SHOULD be specified.
  - Metric MAY be specified.
  - If metric is used, the metrics that MUST be supported are:
    - average - the average flow rate over the entire usage window.
    - total - volume of data transferred in the specific unit. This is the default.
    - min - minimum flow rate in the specific units.
    - max - maximum flow rate in the specific units.

**Property *Disk***

Disk storage used.

- This property SHOULD be referred as disk.
- This property MUST contain data of type positive integer.
- This property is optional.
- Meta-properties:
  - Units MUST be specified.
  - Description MAY be specified.
  - Type MAY be specified. The types that MUST be supported are:
    - scratch
    - temp
  - Metric MAY be specified. The metric that MUST be supported are:
    - average
    - total
    - min
    - max

#### Property *TimeDuration*

This property identifies any additional time duration associated with the resource consumption, e.g.: it might report the connection time within the start of a file transfer and its end.

- This property SHOULD be referred as timeduration.
- This property MUST contain data of type duration.
- This property is optional.

#### Property *TimeInstant*

This property identifies any additionally identified discreet timestamp associated with the resource consumption, e.g. it may represent the moment in which a transfer is scheduled in the queue or the moment in which the transfer is started.

- This property SHOULD be referred as timeinstant.
- This property MUST contain data of type duration.
- This property is optional.

#### Property *ServiceLevel*

This property identifies the quality of service associated with the resource consumption. For example, service level may represent if the file can be persistent or volatile.

- This property SHOULD be referred as servicelevel.
- This property MUST contain data of type string.
- Metric MAY be specified. The metric that MUST be supported are:
  - persistent
  - volatile

#### Property *Extension*

For sites that may want to exchange data not defined in the UR, the Extension property can be used to encode any type of usage information. The sites can agree on the meta properties supported for each extension.

- This property SHOULD be referred as extension.
- This property MUST contain data of type string.

- This property is optional.
- Meta-properties:
  - Units may be supported.
  - Metric may be supported.
  - Name may be supported.
- The meta-property must have data of type string.

## 4.4 Features available on current SEs

There are different SE types used on the Grid and each one has its own distinctive characteristics. For each one a short description of what they provide is given in the following sections.

### 4.4.1 dCache

dCache is a SE developed at Deutsches Elektronen-Synchrotron (DESY) and Fermi National Accelerator Laboratory (FNAL). The goal of this project is to provide a system for storing and retrieving huge amounts of data distributed among a large number of heterogeneous server nodes under a single virtual file system tree with a variety of standard access methods. Depending on the persistency model, dCache provides methods for exchanging data with backend (tertiary) storage systems as well as space management, pool attraction, dataset replication, hot spot determination and recovery from disk or node failures. Connected to a tertiary storage system, the cache simulates unlimited direct access storage space. Data exchanges to and from the underlying Hierarchical Storage Management (HSM) are performed automatically and invisibly to the user[44].

### 4.4.2 dCache Logging

The raw information about all dCache activities can be found in:
`/opt/d-cache/billing/<YYYY>/<MM>/billing-<YYYY.MM.DD>.`

A typical output line is:

```
05.31 22:35:16 [pool:<pool-name>:transfer] [000100000000000000001320,24675] \
     myStore:STRING@osm 24675 474 true {GFtp-1.0 <client-host-fqn> 37592} {0:""}
```

The first brackets contain the pool name and the second the perfectly Normal File System[45] (pNFS) id and the size of the file which is transferred. The other values represent the storage class, the actual amount of bytes transferred and the number of milliseconds the transfer took. The next entry is true if the transfer was a wrote data to the pool. In the second line, the first brackets contain the protocol, client Fully Qualified Name (FQN), and the client host data transfer listen port. The final brackets contain the return status and a possible error message[46].

#### Billing Database

In addition to writing out information to the billing logs, usually in the directory:

```
/opt/d-cache/billing
```

dCache can also write the information into a PostgreSQL[47] database from version 1.6.6 onwards[48].

### 4.4.3 DPM

DPM is a light weight solution for disk storage management. If offers the required SRM interfaces and it has been developed at CERN. The information are stored in the DPM database to allow the system to understand what data is being transferred, which users are accessing them, what protocols are being used, which client hosts are initiating the transfers, how many transfer errors are reported and which pools are these errors occurring on[49].

#### DPM Logging

The log files are spread over *dpns*, *dpm*, *srmv1* and *gridftp* logs. In addition gridftp transactions on pool nodes are logged on that host, not on the admin node. It is then far from easy to work out where one should look. DNs get recorded in multiple places by multiple daemons. Usually the information appears more than once. This can be a bit confusing. The answer to the basic security question to whom put file X on the system is much easier if SRM, as it is normally, is used, e.g.: in a SRM v.2.2 interface the log file is:

```
/var/log/srmv2.2/
```

and the important part of the logs are:

- for incoming files look for a PrepareToPut request;
- for outgoing files look for a PrepareToGet request;
- for deletes look for an rm request.

There are other SRM v.2.2 operations with obvious meanings: Ls, MkDir, RmDir. If more information are required, depending on the protocol utilized a different set of log files should be read[50].

#### Monitoring Packages

The monitoring is packaged up in an RPM, the *GridppDpmMonitor*, and uses Brian Bockleman's GraphTool package[51]. This provides a framework for querying a database, plotting the results and displaying them on a webpage. The *GridppDpm-Monitor* package is strongly influenced by Brian's own dCache billing graph package. Essentially, *GridppDpmMonitor* is a repackage version, with appropriately constructed queries for DPM's MySQL database[52].

### 4.4.4 Castor

The Rutherford Appleton Laboratory (RAL) Tier1 CASTOR accounting is realized in the following way: SEs publishes information with the GLUE schema. They publish which SAs are available, and how much space is available in each SA, how much is used, etc.[53]. The data available from the information system can then be used to monitor the space utilization in a similar way as for GridPP.

### 4.4.5 StoRM

Storage Resource Manager (StoRM) is developed at INFN-CNAF and is a light, scalable, flexible, high-performance, file system independent SRM for generic disk based storage system, compliant with the standard SRM interface version 2.2[54]. Developers have planned a new feature that will allow the development of new sensors that can analyze those file and produce the records R1 described. At the moment those events do not leave traces on the system as there is no log file or historical database of the operations.

# Chapter 5

# Storage accounting with DGAS and HLRmon

*"If enough data is collected, anything may be proven by statistical methods."*
Williams and Holland's Law

## 5.1 Storage accounting workflow with DGAS

As described in Chapter 2, DGAS has been used as the accounting system for computing resources on all the Italian sites and also in other countries like Germany and Greece. Recent versions already in production have introduced user definable tables. Using those tables, it is possible to create a database that contains URs for storage accounting with fields that adhere to the definitions given in Chapter 4. In Figure 5.1 is shown a schema of the workflow with the storage accounting that works independently from the computing accounting but that can share the same HLR.

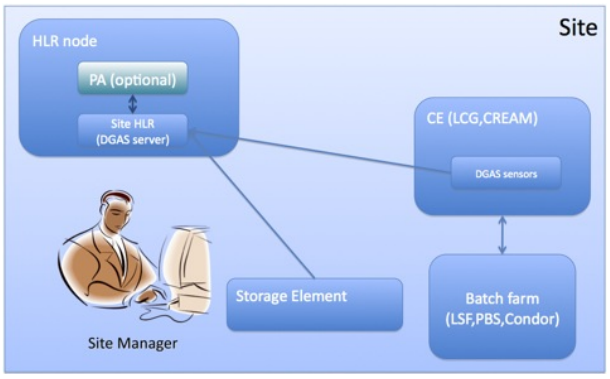


Figure 5.1: Accounting workflow[55]

The URs for the storage accounting are produced on each SEs and sent to the site HLR. At the same time, URs for computing resources are produced on the CEs and sent to the same HLR. If necessary, a dedicated HLR for storage accounting could also be configured. Like in the case of the accounting for computational resources, a specific sensor is necessary for each type of SE. As described in Chapter 4, HLRmon collects storage accounting information from sensors on the SEs that are sent directly to HLRmon with an XML file attached to an email. Instead of using a specific sensor for each SE the strategy temporarily adopted is to query the top level BDII and extract the necessary information to build the needed URs. This is not the optimal solution but it is a first step in the implementation of a complete system that uses standard URs. The next step is to develop sensors that can collect the same information and gradually phase out the UR generation used now. The procedure implemented uses the standard DGAS client. In this way we can obtain the workflow shown on Figure 5.1 where the DGAS standard client, installed with the DGAS packages and described in more detail in Appendix B, sends the URs collected by the sensors directly to the HLR. The DGAS client sends the records to the HLR where the *getRecordEngine* instantiated by the HLR listener insert them in the HLR database. HLRmon will then query the HLR and produce various reports. Figure 5.2 describes the storage accounting toolkit implemented within DGAS and exploited to develop this prototype. Green boxes represents the storage related parts.

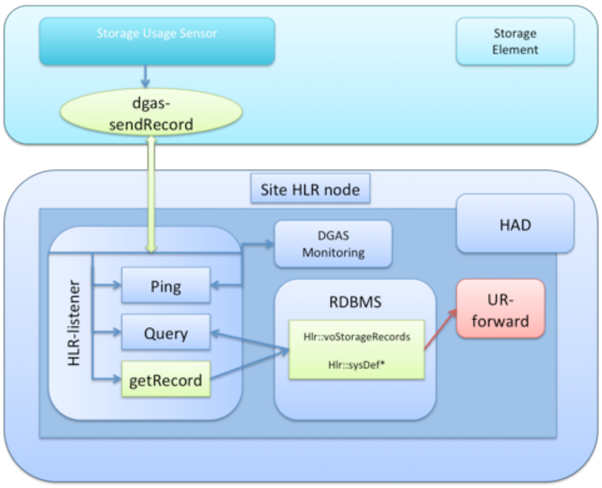


Figure 5.2: Storage accounting toolkit

Blue boxes represent the components, which are in common with the CPU accounting. The red box is the component used to forward the URs to the second level HLR. It is then possible to extend the system capabilities[56], as we have seen, by defining new tables. An implementation that exploit those characteristics has been developed during the thesis work and will be described in more details in the next sections.

## 5.2 Usage Record

Since specific sensors for the specific SE types are not yet available, URs have been created based on the data available on the IS. URs are created interrogating the top level BDII. The information presented in this section, follows the GLUE schema, a common information schema used for the description of Grid resources and services. It has been implemented by every middleware component and it is essential for the inter operability of the Grid infrastructure. The GLUE information schema was originally introduced back in 2002 as a joint project between the European DataGrid (EDG), the Data TransAtlantic Grid (DataTAG) projects and the US International Virtual Data Grid Laboratory (iVDGL). The schema has major components to describe CEs and SEs Elements and also generic service and site information. It has been used extensively in the LHC Computing Grid (LCG) and EGEE Grid for job submission, data management, service discovery and monitoring. The schema itself is an abstract description of information. The primary use cases for the schema is for a snapshot of the current state, although, for performance reasons, there will typically be various caches so information propagation will have some latency. Monitoring use cases may also require historical information to be stored in databases. The information itself is collected by information providers, pieces of code which must be tailored to each implementation of the various GRID services. However, it is desirable to use common code where possible to preserve consistency. In 'Usage of GLUE Schema v1.3 for Worlwide LCG (WLCG) Installed Capacity information'[57] document, it is described how to use the GLUE Schema version v.1.3 in order to publish information to provide the WLCG management with a view of the total installed capacity and resource used by the different VOs in the on each sites. This information can also be used by VO operations and management in order to monitor the VO usage of the resources and, as we have seen, it is possible to extract information for the accounting of storage resources. Version 2.0 of the schema is already available but, as of today, all the implementations of the Grid services utilizes the v.1.3 definition. For this reason the implementation discussed uses the older v.1.3 of the GLUE schema. It is foreseeable that v.2.0 will be implemented in all the middleware and be available in about one year. By this time specialized sensors for the different SEs should be written. More details on the meaning of several GLUE Classes concerning storage resources that can be useful for accounting purpose will now be shown. The **SE Class** describes storage resources at a site. There MAY be more than one SE at one given site. A SE represents a convenient partition of the storage resources of one or more storage systems as a single Grid entity. The attributes of the SE Class described in section 4.1 of 'GLUE Schema Specification v.1.3'[33] have this definition:

- *SizeTotal, SizeFree, TotalOnlineSize, UsedOnlineSize, TotalNearlineSize and UsedNearlineSize.* They SHOULD be aggregated from what is in the SAs and they SHOULD summarize the space in the entire SE. For SEs not supporting tapes, TotalNearlineSize and UsedNearlineSize MUST be published with a value of 0.

The description of the *GlueSA* and *GlueVOInfo* classes are the most relevant for the publication of storage resource installation and usage. The *GlueSA* class describes a logical view of a portion of physical space that can include disks and tape resources. SAs MAY overlap. Shared portions of storage MUST be represented with a single *GlueSA* object, with multiple *GlueSAAccessControlBaseRule* attributes and optionally with multiple *VOInfo* objects pointing to it. Normally a SA is used to represent SRM reserved space. It is RECOMMENDED that a SA object is published for portions of storage configured but not yet reserved. In this case the SA MUST publish ReservedOnlineSize=0 and ReservedNearlineSize=0. A special capability attribute must be used to describe the total installed capacity which is the InstalledCapacity attribute and will be described later. To be noted that for overlapping published SA, in order to avoid double counting of resources, only some of them MUST publish a value for the *InstalledCapacity* attribute greater than 0. The sum of all *InstalledCapacity* attributes over all SA of an SE MUST be equal to the physical storage capacity for that SE.

- *Reserved[Online|Nearline]Size* MUST be published. It is a portion of available storage physically allocated to a VO or to a set of VOs. The value of this attribute MUST be 0 for a SA representing an unreserved space. The *Reserved[Online|Nearline]Size* MUST NOT be negative. For WLCG usage Online refers to space on disk while Nearline refers to space on tape. For tapes, sizes MUST be reported publishing the actual size on tape after compression;

- *Total[Online|Nearline]Size* is the total Online or Nearline space available at a given moment. It SHOULD not include broken disk servers, draining pools, etc. This attribute MUST be published. In the absence of unavailable pools the *TotalSize* is equal to the *ReservedSize* if the *ReservedSize* is greater than 0. The *Total[Online|Nearline]Size* MUST NOT be a negative number and MUST NOT exceed the *Reserved[Online|Nearline]Size* if the *Reserved[Online|Nearline]Size* is greater than 0;

- *Used[Online|Nearline]Size* is the space occupied by available and accessible files that are not candidates for garbage collection. This attribute MUST be published. For CASTOR, since all files in T1D0 are candidates for garbage collection, it has been agreed that in this case *UsedOnlineSize* is equal to *GlueSATotalOnlineSize*. For T0D1 classes of storage this is the space occupied by valid files. Size MUST NOT be a negative number and MUST NOT exceed the *Total[Online|Nearline]Size*;

- *Free[Online|Nearline]Size* MUST be published and MUST be equal to *Total[Online|Nearline]Size-Used[Online|Nearline]Size*.

The value to assign to the *ServiceLevel* attribute can be deduced from the *storage class* assigned to a certain SA. Those storage classes are meant to distinguish among different data management policies. The different storage classes available are:

- *Disk0-Tape1* **D0T1**: data migrated to tape and deleted from disk when the staging area is full;
- *Disk1-Tape0* **D1T0**: data always available on disk, never migrated to tape, never deleted by the system;
- *Disk1-Tape1* **D1T1**: large buffer on disk with a tape back-end[58].

## 5.3 Storage accounting implementation

A table that has the fields listed in Table 5.1 has been created on the HLR. The name of the table is free but since it is user defined, it must start with *sysDef*. For this reason the name *sysDefStorageAccounting* has been chosen.

| **Column name** | **Type** | **Primary key** | **Can be NULL** |
|---|---|---|---|
| RecordIdentity | char(64) | Yes | No |
| GlobalFileId | char(512) | No | Yes |
| LocalFileId | char(512) | No | Yes |
| GlobalGroup | char(64) | No | Yes |
| GlobalUsername | char(128) | No | Yes |
| LocalUserId | char(64) | No | Yes |
| Charge | float | No | Yes |
| Status | char(64) | No | No |
| Host | char(64) | No | No |
| SubmitHost | char(64) | No | No |
| ProjectName | char(64) | No | Yes |
| ProjectPartition | char(64) | No | Yes |
| StorageType | char(64) | No | Yes |
| ProtocolType | char(64) | No | Yes |
| OperationType | char(64) | No | Yes |
| Network | int(10) | No | Yes |
| Disk | int(10) | No | Yes |
| TimeDuration | int(10) | No | Yes |
| TimeInstant | int(10) | No | Yes |
| ServiceLevel | char(64) | No | Yes |

Table 5.1: UR used in the DGAS Storage Accounting

The records are then created on the SE and sent, to the HLR, with the command (described in more detail in Appendix B):

```
glite-dgas-send-record <OPTIONS> [USAGE RECORD LIST]
```

According to the definition given in Table 4.2 in Chapter 4 a new table has been created on a test HLR installed at INFN-CNAF. The fields of this table are those shown in Table 5.1. In this implementation the attribute *SubmitHost* is always empty because the script that analyzes the SE gives simply an aggregate of the usage utilization. In this case *SubmitHost* can not assume any value. More specifically, also the *GlobalFileId*, *LocalFileId*, *GlobalUsername*, *LocalUserId* attributes are never utilized for the same reason and their content is always empty but, for further development, it is useful to have the table already able to accept all those attributes. Also the attribute *Charge* has not been used at this time as the measurement of the utilized storage space was the only parameter taken in consideration. In this table the field *Extension* has not been used because not necessary in the current implementation.

## 5.4 URs collection

In this prototype, the creation of the URs is demanded to a Python script (Appendix B). It reads the storage information by contacting the top level BDII and creates the URs for all the configured sites. This script creates records that have the attributes discussed. This is a simplification and a temporary solution as the records generated do not have the accuracy of a true accounting system and they do not have precise information about the single event e.g.: when the single file was copied in the storage and when it was deleted. However, they can be considered as aggregate records. For those reasons it is also not possible to have any information on the accessed files and on the number of times those files are accessed. The storage measured in the current implementation is those used mostly by the LHC experiments. This means that the files are written once and read many times thus this can be considered an acceptable approximation. Using this temporary solution it is possible to create and test the infrastructure and in future, when specialized sensors for each SE will be available, to collect the URs from those sensors. The specialized sensors will produce URs directly on the site hosting the storage by parsing the logs of the SEs or the database that contains the operation performed on the storage. The URs created during the test which are sent to a the test HLR, a virtual machine with the latest production version of the DGAS packages (v.3.4.0-23) installed on a Scientific Linux[1] (SL) 4, 32 bit Linux distribution. The DGAS client is used to send the records to

[1]SL is a Linux release put together by Fermilab, CERN, and various other labs and universities around the world. Its primary purpose is to reduce duplicated effort of the labs, and to have a common install base for the various experimenters. The base SL distribution is basically Enterprise Linux, recompiled from source[59].

the HLR. It accept a list of attributes and values separated by a blank space. The *TimeDuration* attribute can not be calculated from the actual events on the files but is measured as the time since the previous measure of the space consumed on the SE. In this implementation, the *RecordIdentity* attribute is calculated using an hashing function applied on the a combination of the *GlobalGroup*, *ProjectName* and *TimeInstant* attributes. The corresponding command used to send an usage record takes this form:

```
/opt/glite/libexec/glite-dgas-sendRecord -v 3 -s dgas-test-vm01.cnaf.infn.it -t sysDefStorageAccounting "ID=''"
"RecordIdentity='fa4e77d2286fd8f3d32dcd7cf9080e28019ac584'" "GlobalFileId=''" "LocalFileId=''" "GlobalGroup='INFN-T1""
"GlobalUsername=''" "LocalUserId=''" "Charge=''" "Status=''" "Host='storm-fe-archive.cr.cnaf.infn.it'" "SubmitHost=''"
"ProjectName='magic'" "ProjectPartition='magic:custodial:nearline'" "StorageType=''" "ProtocolType=''" OperationType=''"
"Network=''" "Disk='0'" "TimeDuration='33'" "TimeInstant='1300062872'" "ServiceLevel=''"
```

The options specified sets the verbosity to level 3, the DGAS's HLR where the records are sent to *dgas-test-vm01.cnaf.infn.it* and the table used to store the records to *sysDefStorageAccounting*. An extra attribute is also present: *ID*. This is not an attribute related to the UR but it is just needed for the particular implementation of the DGAS table on the HLR. The script responsible for the generation of the URs is available on Appendix B as well the script that take cares of the execution of the DGAS's client. This script checks for the presence of URs in a configurable place (UR directory). For each UR available it tries to send it to the HLR. In case of failure it moves it in an error directory (UR error directory) or it removes it otherwise. Errors could arise if the network connection between the client and the HLR server is not present or e.g.: the HLR server is not accepting records. After having removed the source of error it is sufficient to move the UR from the UR error directory to the UR directory to have them reprocessed on the next execution of the script.

## 5.5 HLRmon storage interface

HLRmon interface that already shows storage data collected by means of the simple method described in Chapter 4, has been modified to overcome the email method used for collecting UR by querying the HLR and populate its internal database. Figure 5.3 shows a screen shot of the storage web page. For each site, VO and SA, a graph is produced using the URs provided by the HLR. The information on the claimed usage in each graph is recorded in a separate table.

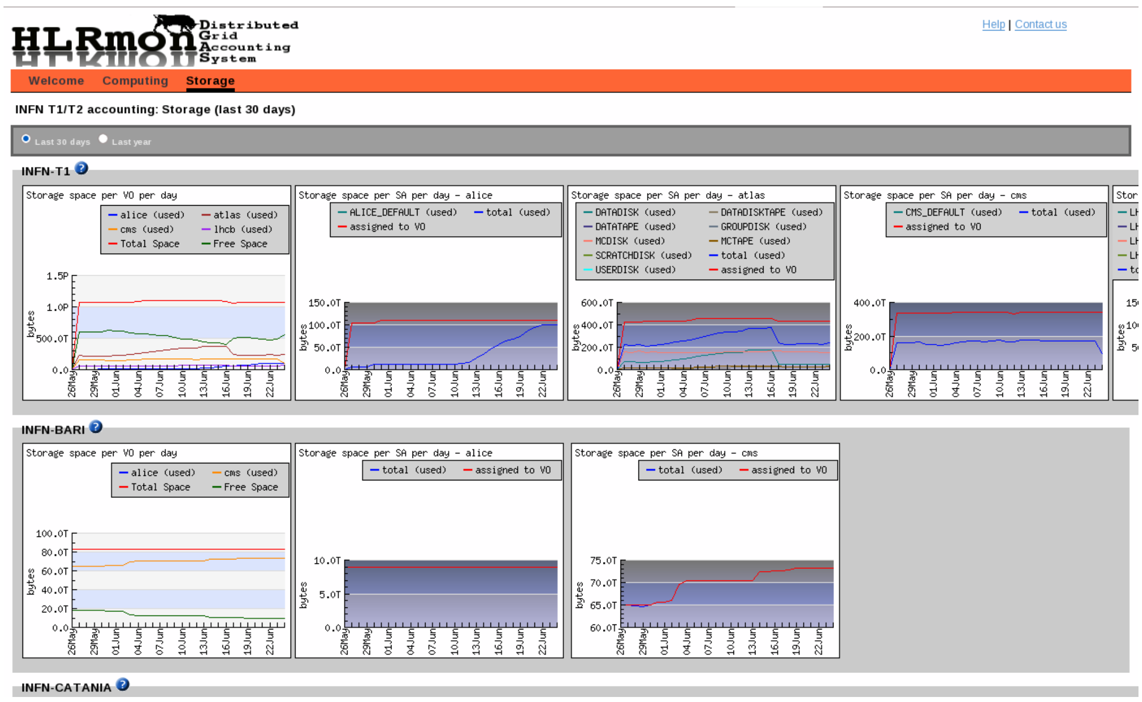

HLRmon Distributed Grid Accounting System
Help | Contact us
Welcome
Computing
Storage
INFN T1/T2 accounting: Storage (last 30 days)
Last 30 days
Last year
INFN-T1
Storage space per VO per day
Storage space per SA per day - alice
Storage space per SA per day - atlas
Storage space per SA per day - cms
INFN-BARI
INFN-CATANIA

Figure 5.3: HLRmon storage accounting page

# Chapter 6

# Tests of DGAS implementation of ActiveMQ

*"Before software should be reusable, it should be usable."*
Ralph Johnson.

## 6.1 ActiveMQ

ActiveMQ is an open source (Apache 2.0 licensed) message broker which fully implements the Java Message Service (JMS) 1.1. It provides "Enterprise Features" like clustering, multiple message stores, and the ability to use any database as a JMS persistence provider[60]. Its success is also due to the support to a large number of programming languages: Java, C, C++, C#, Ruby, Perl, Python and PHP. It consists of a host called broker that collects the messages produced by a series of clients called producers. Those messages are grouped in different channels divided with a logic of some kind decided by the organization or by the system administrators. Another group of computers called consumers reads those messages and might take decisions or produce some kind of output based on the content of the consumed messages. In the last three years, the message system described has been adopted as the standard transport protocol in the EGI-InSPIRE project. One of the implementation of this transport mechanism is for the monitoring of the services. In Figure 6.1, a schema of the messaging distribution for the Nagios monitoring system is shown. The result of the tests that each regional instance of Nagios publishes on the broker are then consumed by different consumers. In the example, we can see the ticketing interface, data warehouse and the alarm system. The same host that acts as a consumer can be at the same time a producer and publish messages on another channel. Because of its success it has been evaluated by different groups and among them, those dealing with the accounting. Both APEL and DGAS have planned the migration to the ActiveMQ messaging system. The currently used DGAS's proprietary system has been proved stable and performant but it can not provide the interoperability needed with other accounting tools. The adoption of ActiveMQ in conjunction with the use of a standard UR will allow the mix of different sensors, accounting system and the exchange of UR between them.

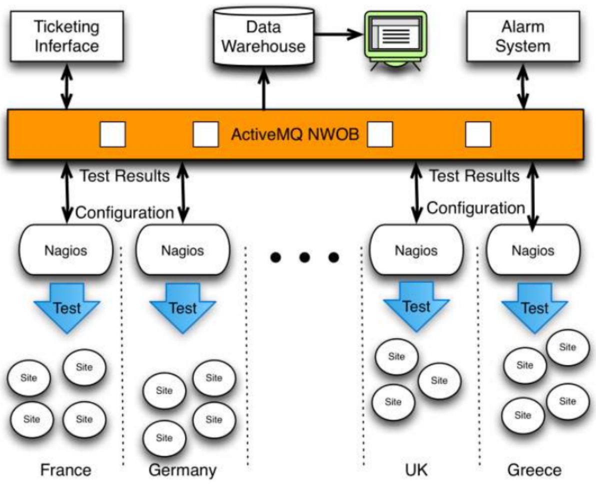


Figure 6.1: ActiveMQ implementation on Nagios[61]

## 6.2 DGAS implementation of ActiveMQ

The DGAS production team has decided to adopt ActiveMQ as the communication mechanism for the different components of the accounting system. The first step is the implementation of ActiveMQ on the channel of communication between the CEs and the HLR. In the following sections, tests are made in this new method of communication between CEs and HLR will be presented. The next step will be to implement and test the ActiveMQ communication between HLRs of different levels and from the HLR to the central repository at Cesga. This is the first implementation step and it is the most important because it should prove the feasibility of an accounting system that relies on ActiveMQ as the transport layer. The migration to ActiveMQ will be gradual and the proprietary transport mechanism will be supported until necessary. The tests on the ActiveMQ implementation and the the coexistence of both communication methods (proprietary and ActiveMQ) has been undertaken in a dedicated testbed. The results of those tests have been compared to ensure the reliability of the new transport mechanism. The proprietary communication protocol has been already widely tested and represent a good index of performance. After a short description of the testbed, the tests that are made and their results will be presented.

## 6.3 Testbed setup

To test the functionality of the new transport mechanism and the new version of the HLR server, a dedicated testbed has been setup. Its configuration is shown of Table 6.1. The CE, the 2 WNs and the 2 HLRs have been installed at INFN-CNAF while the two ActiveMQ servers were already installed one at INFN in Turin, specifically for this tests, and one used as a general purpose and for tests at CERN. The hosts installed at INFN-CNAF were all Virtual Machines (VMs) installed on this hardware:

- Dual quad core Intel(R) Xeon(R) CPU E5520;
- 24 GB RAM;
- 2 hard disk SAS, 10.000 RPM, 300 GB configured with Redundant Array of Independent Disks (RAID) 1.

| # | **Role** | **OS** |
|---|---|---|
| 1 | CE with ActiveMQ enabled sensors | SL 5, 64 bit |
| 2 | WN | SL 5, 64 bit |
| 1 | HLR with the latest production version, v.3.4.0-23 (with legacy transport protocol) | SL 5, 32 bit |
| 1 | HLR prototype with ActiveMQ as transport protocol v.3.4.1-0 | SL 5, 32 bit |
| 1 | dedicated ActiveMQ Broker installed at INFN-Torino | |
| 1 | general purpose ActiveMQ installed at CERN | |

Table 6.1: Testbed configuration

The Operating System (OS) utilized for the computer hosting the VMs was SL 5, 64 bit. Each VM, has been virtualized using Kernel-based Virtual Machine[1] (KVM). With the use of KVM the need of real servers has been dramatically reduced and the entire testbed, with the exception of the two brokers already active, has been possible using only the computer described. Each VM has been configured with the following features:

- 1 Virtual CPU core;
- 1 GB RAM;
- 1 10 GB hard disk[2].

[1]KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions for Intel VT or AMD-V. It consists of a loadable kernel module, *kvm.ko*, that provides the core virtualization infrastructure and a processor specific module, *kvm-intel.ko* or *kvm-amd.ko*[62].

[2]The virtual hard drive has been created using an LVM partition and, for better performances,

## 6.4 Test of ActiveMQ implementation

After the installation of the operating system on the real machine, the setup of the virtualization software and the preparation of the virtual machines with the OS specified in Table 6.1, the middleware necessary for the tests has been installed and configured. For the CE and the 2 WNs, the installation and configuration was accomplished using the standard YAIM profiles for gLite 3.2[64]. After this procedure, only an upgrade of the following packages, that includes the producer for ActiveMQ, was necessary on the CE:

- *glite-dgas-common-3.4.1-0.centos5.x86_64.rpm*
- *glite-dgas-hlr-clients-3.4.1-0.centos5.x86_64.rpm*

For the two HLR, on the other hand, a specific profile was not available. For this reason a manual installation of the RPM packages and a manual setup of the configuration files has been required. The software packages installed in the standard HLR were:

- *glite-dgas-pa-clients-3.4.0-3.centos5*
- *glite-dgas-hlr-clients-3.4.0-12.centos5*
- *glite-dgas-hlr-service-3.4.0-23.centos5*
- *glite-dgas-common-3.4.0-4.centos5*

While those for the HLR wiht ActiveMQ capability enabled were:

- *glite-dgas-pa-clients-3.4.1-0.centos5*
- *glite-dgas-hlr-clients-3.4.1-0.centos5*
- *glite-dgas-hlr-service-3.4.1-0.centos5*
- *glite-dgas-common-3.4.1-0.centos5*

In addition some other packages and their dependencies have been installed and configured on the CE and on the ActiveMQ enabled HLR:

- *autoconf, automake, libtool, cppunit-devel, cppunit, gcc-c++, gcc, e2fsprogs-devel, expat-devel*

*virtio* has been utilized both for disk access and for network emulation. Virtio is a Linux standard for network and disk device drivers where just the guest's device driver "knows" it is running in a virtual environment, and cooperates with the *hypervisor*. This enables guests to get high performance network and disk operations, and gives most of the performance benefits of paravirtualization[63].

This allowed the compilation and installation of the ActiveMQ libraries necessary for the producer on the CE and the consumer on the HLR to run. Those steps are described in more detail in Appendix C. The tests were divided into several rounds to verify all the different aspects and compare the two methods. Different sets of test to validate the communication, integrity, reliability and throughput evaluation have been carried out. Plain text first and then Secure Socket Layer (SSL) enabled transmission which provides the encryption with asymmetric keys to the single UR transmitted have also been performed for all rounds.

### 6.4.1 CE - HLR ActiveMQ test

The purpose of the initial tests is to check the communication between the CE and the ActiveMQ enabled HLR. The first thing to do is to ensure that the messages are correctly produced on the CE and consumed on the HLR. To accomplish this, only the consumer has been started on the HLR. The procedure starts all the standard process necessary to the HLR and some that are new for the ActiveMQ enabled HLR. *glite_dgas_AMQRecordManager* is one of them and should be terminated. This allowed us to check that the UR were sent to the broker and then consumed by the HLR. At this point all the accounting process on the CE could be started and some job were submitted to the CE to produce some UR. Having terminated the *glite_dgas_AMQRecordManager* process on the HLR, the accounting chain is not complete and we verified the content of the UR retrieved from the broker that were present in this folder:

```
/tmp/dgasamq/
```

The files contained in this directory were in plain text or encrypted in accord with the configuration on the CE and on the HLR. This is because it is not the channel that is encrypted but the single UR. This is necessary to reach the required confidentiality for the information transferred to the broker. In fact the broker could be outside the site that is producing those information or even in another country. Encryption of each UR is obtained using the public key of the HLR of destination allowing this HLR to be the only one able to access the content of the UR. To switch between clear text and encrypted is sufficient to switch between the two configurations and restart the services on the CE and the HLR.

### 6.4.2 Communication test with a user defined ActiveMQ topic

The purpose of this test is to verify that the site administrator can freely change the topic used to publish the usage records and at the same time to set another channel on the consumer HLR. This will be the case e.g. when a site administrator is setting up an installation on its site and needs to use the specific channel assigned for his site. This same channel should be set both on the producer and the consumer. The test only required to change, on both configuration file on the CE and on the HLR, of this attribute:

```
dgasAMQTopic = "DGAS.TEST"
```

to another value like this:

```
dgasAMQTopic = "DGAS.ACTIVEMQ"
```

A restart of the accounting services on both machine is the only operation needed. The test has been performed once again with SSL disabled first and then enabled.

### 6.4.3 Integrity check

To perform this test the database needed to be cleaned. To do that the two databases that contains the accounting information on both HLRs have been completely removed and recreated using the scripts provided with the installation packages. The aim of this test is in fact to verify that the content of the records stored on the database, based on the URs produced, correspond exactly to those that are present on the CE that in turn correspond to the number of jobs that are executed on the WNs. The configuration file on the CE has been set in a way to publish the URs on both HLRs. To do that the attribute:

```
res_acct_bank_id
```

*dgas_sensors.conf* file on the CE has been set to point to the legacy HLR and at the same time the configuration for the ActiveMQ publisher has been added. This allowed to publish the same URs on both HLR at the same time and allow a simple comparison of the two databases. The test consisted in the submission of a fixed number of jobs to the CE present on the testbed. In Appendix C, is the script that sends a configurable number of jobs to the CE. The JDL job utilized was the simple execution of the command */usr/bin/whoami*. It was not important to have heavy jobs on the WNs. Instead, it was only required the production of a certain known number of URs. The entire process, submission, processing, execution on the WNs and finally the production of the URs sent to the HLRs required about two days. After this period of time, about 10.000 of records about three quarter without SSL and the rest with SSL enabled were present on both HLR The one running the proprietary communication protocol and the one using ActiveMQ. The test consisted in checking the number of job present in the two HLR and the sum of the CPU time and Wall Clock Time for the two round of execution. In both cases a subset of a dozen of jobs has been successfully checked in detail to ensure that all the information present on the log files of the batch system and the Grid logs agreed with the information present in the correspondent UR recorded on the two HLRs.

### 6.4.4 Service reliability

This test was divided in several parts. Its purpose was to verify the reliability of the services under normal utilization or in the event that one of its components became inactive for any reason. One of the daemons could be dead or simply unavailable for network problems. The tests performed are the followings:

- all the services have been kept active for two days with normal submission of jobs to the CE and consequent production of URs that were then sent to the ActiveMQ HLR. This test is in fact the same as the previous because, as before, the service was kept functional for two days under constant utilization. For this reason it has not been repeated but the result of the previous test have been considered valid also in this case;
- the same test as before has been performed but with the *hlrd* service turned off on both HLRs. This allowed to simulate a temporary freeze or unavailability of this service. After two days the services were restored in both HLRs and the URs started to be processed again. On the ActiveMQ enabled HLR this allowed the consumer to retrieve the URs present on the broker while in the production HLR the normal the URs were sent again and the corresponding file present in the folder *dgasURBox* started to be progressively processed;
- the last part of the test consisted on the simulation of the complete unavailability of the HLRs. This circumstance has been very well tested on the production HLR but was never tested on a ActiveMQ enabled HRL. After other two days of normal activity on the CE and URs sent to the brokers, the services on the new HLR have been restarted and the record were retrieved from the broker.

All the round of this test proved the reliability of the services also in the event of malfunction.

### 6.4.5 Throughput evaluation

The purpose of the last test was to compare the proprietary transport mechanism with the new one. This test requires, necessarily, to check the performances of the HLR with the proprietary communication protocol for comparison. This is due to the limitations of the testbed utilized. Even with good performances provided by the KVM virtualization technology, the VMs used can not be compared to those of a dedicated hardware that is normally used for a service of this kind. For this reason the test using the standard HLR were first performed and then, with the same number of jobs, the new version of DGAS using the two brokers available. On the utilized testbed with only a CE e two WNs was not possible to put under stress the transport layer. For those reasons three months worth of logs copied from a INFN-T1's CE were utilized. The procedure required only a reconfiguration of the testbed CE to use those logs instead of the real ones produced by the software installed. Even if the two CE were of different type, the testbed was PBS while the logs copied from the INFN-T1's CE were produced by LSF, this did not affect the tests because the DGAS sensors only analyze the logs file without checking anything else. At this point it was sufficient to completely clean the database and remove the file corresponding to the attribute in *collectorBufferFileName* on the *dgas_sensors.conf* configuration file on the CE each time a new round of test were started. This allows the sensors on the CE to reprocess all the logs as if they were never processed before and obtain at the same time a reproducible behavior. In all the cases analyzed, the sensors on the CE and all the other process were kept running for 24 hours. The results of the tests are shown in Table 6.2. The test shows that the ActiveMQ throughput is lower than the proprietary messaging system and that the performances are not affected by the SSL encryption.

| **HLR** | **SSL** | **Broker** | **# jobs received by the HLR** |
|---|---|---|---|
| ActiveMQ | no | Turin | 165346 |
| ActiveMQ | yes | Turin | 163289 |
| ActiveMQ | yes | CERN | 168853 |
| Legacy | - | - | 328338 |

Table 6.2: Throughput test summary

## 6.5 Test's result analysis

The main problem during the installation of this new version of DGAS has been the resolution of the packages' dependencies that required a very specific version of the libraries. Wrong version of the packages would make the producer and the consumer not run at all. The correct combination of version has been shown in Section 6.4 in this chapter and in Appendix C. All the tests described, with few exceptions have been carried out using both ActiveMQ brokers, the one at INFN in Turin and the one at CERN that were already installed. At the same time a production HLR installed in the testbed was also used to compare the results of the two systems. All the tests were successful and they showed that ActiveMQ transport mechanism is as reliable as the proprietary transfer protocol. The encryption layer, that is necessary when using a broker outside the site that produce the URs and in general should be used to keep confidentiality on the utilization of the resources, does not affect the performance noticeably. In the test carried out the content of the URs itself was always correct and in accordance with the proprietary system. At the moment the only difference is on the performances that resulted to be noticeably lower with the ActiveMQ method. The number of URs transferred from the CE to the HLR in 24 hours with ActiveMQ is half of the number of URs transferred with the proprietary method in the same period of time as shown if Table 6.2. We expected some differences caused by the overhead due to the use of the new method but this test can not be considered satisfying. However, all the other tests were positive, the advantages in the use of ActiveMQ and the possibility to optimize the current code we are optimistic that future versions will show an increase in performances. It should be also noted that DGAS is an accounting system and not a monitoring system thus even if the URs arrive with some delay this is not a problem. The most important thing is that they are all processed correctly as verified with all the other tests. Another less severe problem that arose during this study was the huge amount of logs written on the CE's local disk by the producer, *pushdAscii.log* and by the consumer on the HLR, *dgas-hlr-amq-manager.log* and *hlr_ qmgrd.log*. Often the size of the logs filled the space available on the VMs in a very short amount of time blocking all the operations. Special cronjobs have been set in place to rotate those logs and restart the services when needed. Without those operations it would have not been possible to continue the tests scheduled. Future version should allow to set the verbosity of those log files or give the choice of enable, for debugging purpose, or disable them them when not needed. The results of those tests will also be presented, in a poster, during the "EGI User Forum 2011" that will take place in Vilniuls, Lithuania from the 11st to the 15th of April 2011[65].

# Conclusions

The utilization of Grid technologies in different field of research has greatly expanded in the last few years and it is foreseeable that it will keep growing in the years to come. In Chapter 1, an overview of some European projects with particular emphasis to EGI-InSPIRE and the *gLite* middleware has been given. Those projects are aimed to promote and develop an innovative and sustainable Grid. Grids are composed of several different services that must interact with each others. One of the key aspects that is getting more and more attention is the accounting of the utilized resources. Computational resources have been studied for some years now and there are different solutions available while accounting for storage resources has just started to receive the deserved attention. In light of those problematic, the work thesis focused, on Chapter 3, on the implementation of a system for the assessment of the computational accounting provided by DGAS. This proved to be a useful activity allowing the INFN-T1 and INFN-CNAF-LHCB sites to validate their accounting data and to detect when problems arose after a misconfiguration. The focus on Chapter 4, moved to the accounting systems for storage where the available tools were shown along with some of their distinctive characteristics. In the same chapter, the work continued with the definition of a standard UR for storage accounting and a description of the features present in the current SE implementations. A series of attributes has also been defined based on the information that are relevant to this type of accounting. In Chapter 5, the work has been focused on the implementation of a new accounting system for the storage using the previously given definitions. Temporarily, this implementation relies on a top level BDII to build the URs. URs generated are then collected using DGAS and visualized in a HLRmon web page. This separation between the URs generation, the storing and the visualization is an essential feature and will allow the implementation of new sensors that gather the information necessary for the production of the URs directly on the SE. New sensors will also provide more precise measurement and information e.g.: the users owning the file, number of times a file has been accessed, etc. In Chapter 6, the final focus of the thesis moved on the testing of the performances and reliability of the latest release of DGAS. It is not a public release yet, but it is a first step toward the migration to a standard communication protocol. Together, with the use of a standard UR both for the storage and computation resources, it will ensure a system that uses standard protocols and interfaces. The work done is an important step toward the definition and implementation of a system of this type. It has been proved the feasibility of a storage accounting system by defining a UR, developing a general basic sensor for the SE and reusing some of the tools already available. More development is required especially for the implementation of specific SE sensors that can provide more detailed UR. Nevertheless, the work done demonstrate the feasibility of this system. Once all the accounting systems will share the same standard protocols for both communication and URs it will be possible to exchange URs and mix different components.

# Glossary

## A

| | |
|---|---|
| **Amazon EC2** | Amazon Elastic Compute Cloud; |
| **Amazon S3** | Amazon Simple Storage Service; |
| **APEL** | Accounting Processor for Event Logs; |
| **API** | Application Programming Interface; |

## B

| | |
|---|---|
| **BDII** | Berkeley Database Information Index; |

## C

| | |
|---|---|
| **CE** | Computing Element; |
| **CERN** | Conseil Européen pour la Recherche Nucléaire (European Laboratory for Particle Physics); |
| **CESGA** | Centro de Supercomputación de Galicia; |
| **CNAF** | Centro Nazionale per la Ricerca e Sviluppo nelle Tecnologie Informatiche e Telematiche; |
| **CPT** | CPU time; |
| **CPU** | Central Processing Unit; |
| **CREAM CE** | Computing Resource Execution And Management CE; |
| **CSV** | Comma Separated Value; |

## D

| | |
|---|---|
| **DataTAG** | Data TransAtlantic Grid; |
| **DESY** | Deutsches Elektronen-Synchrotron; |
| **DGAS** | Distributed Grid Accounting System; |
| **DN** | Distinguished Name; |
| **DPM** | Disk Pool Manager; |

## E

| | |
|---|---|
| **EDG** | European DataGrid; |
| **EDGeS** | Enabling Desktop Grids for e-Science; |
| **EGEE** | Enabling Grids for E-Science in Europe; |
| **EGI-InSPIRE** | European Grid Initiative - Integrated Sustainable Pan-European Infrastructure for Researchers in Europe) ; |

## F

| | |
|---|---|
| **FNAL** | Fermi National Accelerator Laboratory; |
| **FQAN** | Fully Qualified Domain Name; |
| **FQN** | Fully Qualified Name; |
| **FTP** | File Transfer Protocol; |
| **FTS** | File Transfer Service; |

## G

| | |
|---|---|
| **GGUS** | Global Grid User Support; |
| **GLUE** | Grid Laboratory for a Uniform Environment; |
| **GLUE-WG** | GLUE - Working Group; |
| **GPFS** | General Parallel File System; |
| **GUID** | Grid Unique IDentifier; |

## H

| | |
|---|---|
| **HEP** | High Energy Physics; |
| **HLR** | Home Location Register; |
| **HTML** | HyperText Markup Language; |
| **HSM** | Hierarchical Storage Management; |

## I

| | |
|---|---|
| **ICE** | Interface to CREAM Environment; |
| **IGI** | Italian Grid Initiative; |
| **INFN** | Istituto Nazionale di Fisica Nucleare; |
| **INFN CA** | INFN Certification Authority; |
| **IS** | Information System; |
| **iVDGL** | International Virtual Data Grid Laboratory; |

## J

| | |
|---|---|
| **JDL** | Job Description Language; |
| **JMS** | Java Message Service; |

## K

| | |
|---|---|
| **KVM** | Kernel-based Virtual Machine; |

## L

| | |
|---|---|
| **LB** | Logging & Bookkeeping; |
| **LCG** | LHC Computing Grid; |
| **lcgCE** | LHC Computing Grid CE; |
| **LDAP** | Lightweight Directory Access Protocol; |
| **LDIF** | LDAP Data Interchange Format; |
| **LFC** | Local File Catalog; |
| **LFN** | Logical File Name; |
| **LHC** | Large Hadron Collider; |
| **LHCb** | LHC beauty experiment; |
| **LRMS** | Local Resource Management System; |
| **LSF** | Load Sharing Facility; |
| **LVM** | Logical Volume Manager; |

## M

| | |
|---|---|
| **MAC** | Media Access Control; |
| **MoU** | Memorandum of Understanding; |

## N

| | |
|---|---|
| **NGI** | National Grid Initiative; |
| **NMR** | Nuclear Magnetic Resonance; |

## O

| | |
|---|---|
| **OGF** | Open Grid Forum; |
| **OS** | Operating System; |

## P

| | |
|---|---|
| **pNFS** | perfectly Normal File System; |
| **PHP** | PHP Hypertext Preprocessor; |
| **PBS** | Portable Batch System; |

## R

| | |
|---|---|
| **RAL** | Rutherford Appleton Laboratory; |
| **RAID** | Redundant Array of Independent Disks; |
| **REST** | Representational State Transfer; |
| **ROC** | Regional Operation Center; |
| **RPM** | RPM Package Manager; |

## S

| | |
|---|---|
| **SA** | Storage Area; |
| **SAGE** | Storage Accounting for Grid Environments; |
| **SAN** | Storage Area Network; |
| **SGAS** | SweGrid Accounting System; |
| **SGE** | Sun Grid Engine; |
| **SIENA** | Standards and Interoperability for eInfrastructure Implementation Initiative; |
| **SL** | Scientific Linux; |
| **SRM** | Storage Resource Manager; |
| **SSL** | Secure Socket Layer; |
| **StoRM** | Storage Resource Manager; |
| **SQL** | Structured Query Language; |
| **SURL** | Storage URL; |

## T

| | |
|---|---|
| **TURL** | Transport URL; |

## U

| | |
|---|---|
| **UI** | User Interface; |
| **UMD** | Unified Middleware Roadmap; |
| **UNICORE** | Uniform Interface to Computing Resources; |
| **UR-FR** | UR - Format Recommendation; |
| **UR-WG** | Usage Record - Working Group; |
| **URL** | Uniform Resource Locator; |
| **UUID** | Universally Unique IDentifier; |

## V

| | |
|---|---|
| **VM** | Virtual Machine; |
| **VO** | Virtual Organization; |
| **VOMS** | VO Membership Service; |

## W

| | |
|---|---|
| **WeNMR** | Worldwide e-Infrastructure for NMR and structural biology; |
| **WLCG** | Worldwide LCG; |
| **WMS** | Workload Management System; |
| **WTC** | Wall Clock Time; |

## X

| | |
|---|---|
| **XML** | eXtensible Markup Language; |

## Y

| | |
|---|---|
| **YAIM** | YAIM Ain't an Installation Manager; |

# List of Tables

# List of Figures

# Appendix A

# *cross-check* scripts

## A.1 Generic scripts for data retrieval

### populate_db.sh

```
#!/bin/bash

source ./conf_site/populate_db_site.conf
source ./conf/populate_db.conf
source ./conf/update_tables.conf


cd $BASEDIR


case "$1" in
        hlr)
                $QUERY_DIR/populate_hlr.sh 2>&1 $LOGDIR/$LOGFILE_HLR
        ;;

        redeye)
                $QUERY_DIR/populate_redeye.sh 2>&1 $LOGDIR/$LOGFILE_REDEYE
        ;;

        batch)
                $QUERY_DIR/populate_batch.sh 2>&1 $LOGDIR/$LOGFILE_BATCH
        ;;

        all)
                $QUERY_DIR/populate_hlr.sh 2>&1 $LOGDIR/$LOGFILE_HLR
                $QUERY_DIR/populate_redeye.sh 2>&1 $LOGDIR/$LOGFILE_REDEYE
                $QUERY_DIR/populate_batch.sh 2>&1 $LOGDIR/$LOGFILE_BATCH
        ;;

        *)
                echo $"Usage: $0 {hlr|redeye|batch|all}"
        exit 1
esac
```

### update_db.sh

```
#!/bin/bash

source ./conf_site/update_tables_site.conf
source ./conf/update_tables.conf


cd $BASEDIR


case "$1" in
        hlr)
                $QUERY_DIR/update_hlr_table.sh 2>&1 $LOGDIR/$HLR_LOG
        ;;
        batch)
                $QUERY_DIR/update_batch_table.sh 2>&1 $LOGDIR/$BATCH_LOG
        ;;

        redeye)
                $QUERY_DIR/update_redeye_table.sh 2>&1 $LOGDIR/$REDEYE_LOG
        ;;

        all)
                $QUERY_DIR/update_batch_table.sh 2>&1 $LOGDIR/$BATCH_LOG
                $QUERY_DIR/update_hlr_table.sh 2>&1 $LOGDIR/$HLR_LOG
                $QUERY_DIR/update_redeye_table.sh 2>&1 $LOGDIR/$REDEYE_LOG
        ;;

        *)
                echo $"Usage: $0 {hlr|batch|redeye|all}"
        exit 1

esac
```

## A.2 Generic scripts data insert in the database

The *populate_ db.sh* and *update_ tables.sh* are run every night with the cronjob:

```
0 6 * * *       cd /home/cristofori/cross_check_INFN_CNAF_LHCB/ && ./populate_db.sh all && ./update_tables.sh all
```

### update_ db.conf

```
### HLR settings (Default: 10 get, 10 delete) (should be the same)
export INTERVAL_HLR_GET=10
export INTERVAL_HLR_DELETE=10

### HLR local settings
export DIR_HLR=/tmp
export FILENAME_HLR=$DIR_HLR/"hlr_"$SITE"_`date +%F`.csv"
export LOGFILE_HLR=log_query_hlr.log


### REDEYE settings (Default: 10 get, 10 delete)
export INTERVAL_REDEYE_GET=10                           #(should be the same as $INTERVAL_REDEYE_DELETE in update_tables.conf)

### REDEYE local settings
export DIR_REDEYE=/tmp
export FILENAME_REDEYE=$DIR_REDEYE/"redeye_"$SITE"_`date +%F`.csv"
export LOGFILE_REDEYE=log_query_redeye.log



### BATCH local setting
export DIR_BATCH=/tmp
export FILENAME_BATCH=$DIR_BATCH/"batch_"$SITE"_`date +%F`.csv"
export LOGFILE_BATCH=log_query_batch.log


### GLOBAL setting
export QUERY_DIR=$BASEDIR/query
export QUERY_SITE_DIR=$BASEDIR/query_site
```

### update_ tables.conf

```
### BATCH local setting
export BATCH_LOG=update_batch_table.log
export INTERVAL_BATCH_VIEW_DEL=120

### HLR settings
export HLR_LOG=update_hlr_table.log
export INTERVAL_HLR_VIEW_UPDATE=10
export INTERVAL_HLR_VIEW_DEL=7
```

```
### REDEYE settings
export REDEYE_LOG=update_redeye_table.log
export INTERVAL_REDEYE_DELETE=10                        #(should be the same as $INTERVAL_REDEYE_GET in populate_db.conf)

### GLOBAL setting
export YESTERDAY=`date --date="1 day ago" "+%Y-%m-%d"`
export QUERY_DIR=$BASEDIR/query
export QUERY_SITE_DIR=$BASEDIR/query_site
```

## A.3 Configuration files

### populate_db_site.conf

```
#!/bin/bash

### COMMON SETTINGS
export BASEDIR="/home/cristofori/cross_check_INFN_CNAF_LHCB"    # Check This!!!
export LOGDIR=$BASEDIR/log

### Local database
export DBHOST=localhost
export DBNAME=cross_check_INFN_CNAF_LHCB                        # Check This!!!
export DBUSER=cross_check
export DBPASS=ch3ck
export SITE="INFN-CNAF-LHCB"                                    # Check This!!!

### HLR database
export HLRHOST=hlr-t1.cr.cnaf.infn.it
export HLRNAME=hlr
export HLRUSER=*****
export HLRPASS=*****



### REDEYE database
export REDEYEHOST=log3.cnaf.infn.it
export REDEYENAME=accounting
export REDEYEUSER=*****
export REDEYEPASS=*****
```

### update_tables_site.conf

```
### COMMON SETTINGS
export BASEDIR=/home/cristofori/cross_check_INFN_CNAF_LHCB      # Check this!!!
export LOGDIR=$BASEDIR/log

### Local database
export DBHOST=localhost
export DBNAME=cross_check_INFN_CNAF_LHCB                        # Check this!!!
export DBUSER=*****
export DBPASS=*****
export SITE=INFN-CNAF-LHCB                                      # Check this!!!
```

## A.4 Generic database populate scripts

### populate_batch.sh

```
#!/bin/bash


echo "##########################" >>$LOGDIR\/$LOGFILE_BATCH
echo    `date "+%Y-%m-%d %T"`      >>$LOGDIR\/$LOGFILE_BATCH
echo "##########################" >>$LOGDIR\/$LOGFILE_BATCH



#Get the data for the last 10 days from the HLR and save them in $FILENAME_BATCH
```

```
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "LOAD DATA LOCAL INFILE \
'$FILENAME_BATCH' \
IGNORE \
INTO TABLE \
        batch \
        FIELDS TERMINATED BY ';' \
        (endtime, \
        lrmsid, \
        userid, \
        vo, \
        que, \
        starttime, \
        cputime, \
        walltime, \
        subhost); \
show warnings" >> $LOGDIR\/$LOGFILE_BATCH



#Set the SITE name for the new jobs

mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "UPDATE \
        batch \
set \
        site='$SITE' \
where \
        site=''; \
show warnings" >> $LOGDIR\/$LOGFILE_BATCH




#Set walltime to 0 where start time is 0 and the wall time is bigger than 1 month (job cancelled)

mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "update \
        batch \
set \
        walltime=0 \
where \
        starttime=0 and \
        walltime/(3600*24) > 30"  >> $LOGDIR\/$LOGFILE_BATCH



###End Common part
#Execute site-specific query

$QUERY_SITE_DIR/populate_batch_site.sh
```

## populate_hlr.sh

```
#!/bin/bash


echo "###########################" >>$LOGDIR\/$LOGFILE_HLR
echo    `date "+%Y-%m-%d %T"`      >>$LOGDIR\/$LOGFILE_HLR
echo "###########################" >>$LOGDIR\/$LOGFILE_HLR



#Get the data for the last 10 days from the HLR and save them in $FILENAME_HLR

mysql --skip-column-names -B -s --database $HLRNAME -h $HLRHOST -u $HLRUSER  -p$HLRPASS -e "select \
        DATE(enddate), \
        siteName, \
        userVo, \
        count(*) as NJobs, \
        sum(cpuTime)/3600 as cpuTime, \
        sum(wallTime)/3600 as wallTime, \
        voOrigin \
from \
        jobTransSummary \
where \
        DATE(enddate)>=DATE_SUB(CURDATE(),INTERVAL '$INTERVAL_HLR_GET' day ) and \
        DATE(enddate) < DATE(CURDATE()) and \
        siteName='$SITE' \
group by \
        date(enddate), \
        userVo, \
        voOrigin, \
        siteName" > $FILENAME_HLR



#Delete HLR data for last 10 days

mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "delete \
from \
        hlr \
where \
        DATE(date)>=DATE_SUB(CURDATE(), INTERVAL '$INTERVAL_HLR_DELETE' day) and \
        DATE(date) < CURDATE() and\
        SITE='$SITE'; show warnings" >> $LOGDIR\/$LOGFILE_HLR




#Import last 10 days data from HLR

mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "LOAD DATA LOCAL INFILE \
'$FILENAME_HLR' \
INTO TABLE \
        hlr \
        FIELDS TERMINATED BY '\t' \
        LINES TERMINATED BY '\n' \
        (DATE, \
        SITE, \
        VO, \
        NJOBS, \
        CPUTIME, \
        WALLTIME,
        VOORIGIN); \
show warnings" >> $LOGDIR\/$LOGFILE_HLR



###End Common part
#Execute site-specific query

$QUERY_SITE_DIR/populate_hlr_site.sh
```

## populate_redeye.sh

```
#!/bin/bash


echo "##########################" >>$LOGDIR\/$LOGFILE_REDEYE
echo    `date "+%Y-%m-%d %T"`      >>$LOGDIR\/$LOGFILE_REDEYE
echo "##########################" >>$LOGDIR\/$LOGFILE_REDEYE


#Get the data for the last 10 days from REDEYE and save them in $FILENAME_REDEYE

mysql --skip-column-names -B -s --database $REDEYENAME  -h $REDEYEHOST -u $REDEYEUSER -p$REDEYEPASS -e "select \
        DATE(date), \
        queue, \
        njobs, \
        cpt_hours, \
        wct_hours, \
        '$SITE'
from \
        all_jobs \
where \
        date>=DATE_SUB(CURDATE(), INTERVAL '$INTERVAL_REDEYE_GET' day ) and \
        date < CURDATE() and \
        queue not in \
                ('ams', \
                'babar', \
                'babar_build', \
                'babar_objy', \
                'babar_xxl', \
                'babar_test', \
                'lost_and_found', \
                'pps', \
                'quarto', \
                'Tier1', \
                'wnod')" > $FILENAME_REDEYE;
```

```
#Delete REDEYE data for last $INTERVAL_REDEYE_DELETE

mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "delete \
from \
        redeye \
where \
        DATE(date)>=DATE_SUB(CURDATE(), INTERVAL '$INTERVAL_REDEYE_DELETE' day) and \
        DATE(date) < CURDATE(); \
show warnings" >> $LOGDIR\/$LOGFILE_REDEYE




#Import last 10 days data from REDEYE

mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "LOAD DATA LOCAL INFILE \
'$FILENAME_REDEYE' \
INTO TABLE \
        redeye \
        FIELDS TERMINATED BY '\t' \
        LINES TERMINATED BY '\n' \
        (DATE, \
        QUEUE, \
        NJOBS, \
        CPUTIME, \
        WALLTIME, \
        SITE); \
show warnings" >> $LOGDIR\/$LOGFILE_REDEYE


###End Common part
#Execute site-specific query

$QUERY_SITE_DIR/populate_redeye_site.sh
```

## A.5 Generic database update scripts

### update_batch_table.sh

```
#!/bin/bash



echo "###########################" >>$LOGDIR\/$BATCH_LOG
echo    `date "+%Y-%m-%d %T"`      >>$LOGDIR\/$BATCH_LOG
echo "###########################" >>$LOGDIR\/$BATCH_LOG


export YESTERDAY=`date --date="1 day ago" "+%Y-%m-%d"`



#Delete the last days informations
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "UPDATE \
        batch_grouped \
        SET
                njobs=0, \
                cputime=0, \
                walltime=0 \
        where \
                date>=DATE_SUB(CURDATE(),INTERVAL '$INTERVAL_BATCH_VIEW_DEL' day ) and \
                site='$SITE';
show warnings" >> $LOGDIR\/$BATCH_LOG




#Update the batch_grouped table with the aggregate values
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "REPLACE INTO \
        batch_grouped \
                (date, \
                vo, \
                queue, \
                njobs, \
                cputime, \
                walltime, \
                site) \
        SELECT \
                date(from_unixtime(endtime)) as date, \
                vo, \
                que as queue, \
                count(*) as njobs, \
                sum(cputime)/3600 as cputime, \
                sum(walltime)/3600 as walltime, \
                site \
        FROM \
                batch \
        WHERE \
                que<>'pps' and \
                vo<>'unknown' \
        group by \
                date, \
                vo, \
                que, \
                site; \
show warnings" >> $LOGDIR\/$BATCH_LOG


###End Common part
#Execute site-specific table query

$QUERY_SITE_DIR/update_batch_table_site.sh
```

## update_hlr_table.sh

```
#!/bin/bash


echo "###########################" >>$LOGDIR\/$HLR_LOG
echo    `date "+%Y-%m-%d %T"`      >>$LOGDIR\/$HLR_LOG
echo "###########################" >>$LOGDIR\/$HLR_LOG



#Delete hlr_local_view table record for the last 7
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "DELETE \
        FROM \
                hlr_local_view \
        where
                date>=DATE_SUB(CURDATE(),INTERVAL '$INTERVAL_HLR_VIEW_DEL' day ); \
show warnings" >> $LOGDIR\/$HLR_LOG


#Update and insert hlr_local_view record for the last 10 days with the aggregate values
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "REPLACE INTO \
        hlr_local_view \
                (date, \
                vo, \
                njobs, \
                cputime, \
                walltime, \
                site) \
        SELECT \
                date, \
                vo, \
                sum(njobs) as njobs, \
                sum(cputime) as cputime, \
                sum(walltime) as walltime, \
                site \
        FROM \
                hlr \
        WHERE \
                voorigin='map' and\
                date>=DATE_SUB(CURDATE(),INTERVAL '$INTERVAL_HLR_VIEW_UPDATE' day )
        group by \
                date, \
                vo, \
                site; \
show warnings" >> $LOGDIR\/$HLR_LOG







#Delete hlr_grid_view table record for the last 7
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "DELETE \
        FROM \
                hlr_grid_view \
        where
                date>=DATE_SUB(CURDATE(),INTERVAL '$INTERVAL_HLR_VIEW_DEL' day ); \
show warnings" >> $LOGDIR\/$HLR_LOG


#Update and insert hlr_grid_view record for the last 10 days with the aggregate values
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "REPLACE INTO \
        hlr_grid_view \
                (date, \
                vo, \
                njobs, \
                cputime, \
                walltime, \
                site) \
        SELECT \
                date, \
                vo, \
                sum(njobs) as njobs, \
                sum(cputime) as cputime, \
                sum(walltime) as walltime, \
                site \
        FROM \
                hlr \
        WHERE \
                (voorigin='fqan' or voorigin='pool') and\
                date>=DATE_SUB(CURDATE(),INTERVAL '$INTERVAL_HLR_VIEW_UPDATE' day )
        group by \
                date, \
                vo, \
                site; \
show warnings" >> $LOGDIR\/$HLR_LOG






#Delete hlr_all_view table record for the last 7
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "DELETE \
        FROM \
                hlr_all_view \
        where
                date>=DATE_SUB(CURDATE(),INTERVAL '$INTERVAL_HLR_VIEW_DEL' day ); \
show warnings" >> $LOGDIR\/$HLR_LOG


#Update and insert hlr_all_view record for the last 10 days with the aggregate values
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "REPLACE INTO \
        hlr_all_view \
                (date, \
                vo, \
                njobs, \
                cputime, \
                walltime, \
                site) \
        SELECT \
                date, \
                vo, \
                sum(njobs) as njobs, \
                sum(cputime) as cputime, \
                sum(walltime) as walltime, \
                site \
        FROM \
                hlr \
        WHERE \
                date>=DATE_SUB(CURDATE(),INTERVAL '$INTERVAL_HLR_VIEW_UPDATE' day )
        group by \
                date, \
                vo, \
                site; \
show warnings" >> $LOGDIR\/$HLR_LOG



###End Common part
#Execute site-specific table query

$QUERY_SITE_DIR/update_hlr_table_site.sh
```

## update_redeye_table.sh

```
#!/bin/bash
```

```
echo "###########################" >>$LOGDIR\/$REDEYE_LOG
echo    `date "+%Y-%m-%d %T"`      >>$LOGDIR\/$REDEYE_LOG
echo "###########################" >>$LOGDIR\/$REDEYE_LOG



###End Common part
#Execute site-specific table query

$QUERY_SITE_DIR/update_redeye_table_site.sh
```

## A.6 Site specific database populate scripts

### populate_batch_site.sh

```
#Site-specific query for the batch record
```

### populate_hlr_site.sh

```
#Site-specific query for the hlr record

#Rename vo pillhcb in lhcb
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "UPDATE \
                hlr \
        SET \
                VO='lhcb' \
        WHERE \
                VO='pillhcb'; \
show warnings" >> $LOGDIR\/$LOGFILE_HLR
```

### populate_redeye_site.sh

```
#Site-specific query for the redeye record
```

## A.7 Site specific database update scripts

### update_batch_table_site.sh

```
#Site-specific query for the batch table


#Create empty lines for YESTERDAY
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "INSERT IGNORE INTO \
        batch_grouped \
                (date, \
                vo, \
                queue, \
                njobs, \
                cputime, \
                walltime, \
                SITE) \
        VALUES \
                ('$YESTERDAY', \"lhcb\", '', 0, 0, 0, \"$SITE\"); \
show warnings" >> $LOGDIR\/$BATCH_LOG


#Rename queue dteam in ops
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "UPDATE \
                batch_grouped \
        SET \
                VO='ops' \
        WHERE \
                VO='dteam'; \
show warnings" >> $LOGDIR\/$BATCH_LOG
```

### update_hlr_table_site.sh

```
#Site-specific query for the hlr table


#Rename queue dteam in ops
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "UPDATE \
                hlr \
        SET \
                VO='ops' \
        WHERE \
                VO='dteam'; \
show warnings" >> $LOGDIR\/$REDEYE_LOG
```

### update_redeye_table_site.sh

```
#Site-specific query for the redeye table



#Delete the records non belonging to INFN-CNAF-LHCB

mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "delete from\
        redeye \
where \
        DATE(date)>=DATE_SUB(CURDATE(), INTERVAL '$INTERVAL_REDEYE_DELETE' day) and \
        queue not in\
                ('lhcb_tier2', \
                'cert_t2')" >> $LOGDIR\/$REDEYE_LOG


#Rename queue lhcb_tier2 in lhcb
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "UPDATE \
                redeye \
        SET \
                QUEUE='lhcb' \
        WHERE \
                QUEUE='lhcb_tier2'; \
show warnings" >> $LOGDIR\/$REDEYE_LOG


#Rename queue cert_t2 in cert
mysql -vv --database $DBNAME -h $DBHOST -u $DBUSER -p$DBPASS -e "UPDATE \
                redeye \
        SET \
                QUEUE='ops' \
        WHERE \
                QUEUE='cert_t2'; \
show warnings" >> $LOGDIR\/$REDEYE_LOG
```

## A.8 CE scripts

### convert-batch_log.sh

```
#!/bin/bash

export SITE=INFN-T1                                                                #Check this!!!
export BASEDIR=/root/lsf-analysis/INFN-T1                                          #Check this!!!

export BATCH_BASEDIR=/usr/share/lsf/work/tier1-lsf/dgas/Tier1/                     #Check this!!!
export FILENAME=batch.acct                                                         #Check this!!!

export YESTERDAY=`date --date='2 days ago' +%Y-%m-%d`
export TODAY=`date +%Y-%m-%d`

export BATCH_SCRIPT=lsfCheck.pl                                                    #Check this!!!
export LOG_FILE=convert-batch.log

cd $BASEDIR


##### INFN-T1


cat ${BATCH_BASEDIR}/lsb.acct.1 ${BATCH_BASEDIR}/lsb.acct.2 > ${BASEDIR}/${FILENAME}                  #Check this!!!

${BASEDIR}/${BATCH_SCRIPT} ${FILENAME} \'${YESTERDAY}\' \'${TODAY}\' >> ${BASEDIR}/${LOG_FILE}

scp ${BASEDIR}/all_lsb_acct.txt root@gstore.cnaf.infn.it:/tmp/batch_${SITE}_`date +%F`.csv >> ${BASEDIR}/${LOG_FILE}           #Check this!!!
```

### lsfCheck.pl

This script extract the UR information from LSF batch system log files. The initial development of the script has been made by Riccardo Brunetti[67].

```
#!/usr/bin/perl

use Text::ParseWords;
use Time::Local;
#use strict;

my $i = 0;
my $cpuTime = 0;
my $wallTime = 0;
my $line;

($yyyy, $mm, $dd) = $ARGV[1] =~ /(\d+)-(\d+)-(\d+)/;
# calculate epoch seconds at midnight on that day in this timezone
$epoch_seconds = timelocal(0, 0, 0, $dd, $mm-1, $yyyy);
my $begints=$epoch_seconds;
print "$begints\n";

($yyyy, $mm, $dd) = $ARGV[2] =~ /(\d+)-(\d+)-(\d+)/;
# calculate epoch seconds at midnight on that day in this timezone
$epoch_seconds = timelocal(0, 0, 0, $dd, $mm-1, $yyyy);
my $endts=$epoch_seconds;
print "$endts\n";

$mindate = $endts;
$maxdate = $begints;

open(OUTPUT, "> all_lsb_acct.txt");
open(FILE, $ARGV[0]) || &error("Error: Cannot open configuration file: $ARGV[0]\n");
        while($line = <FILE>)
        {
                &parseUR_lsf($line,$i);
        }

while (($key,$value) = each(%njobsVO)) {
        print "Number of jobs of $key is $value\n";
}
while (($key,$value) = each(%cpuTimeVO)) {
        print "Total cpuTime of $key is $value\n"
}
while (($key,$value) = each(%wallTimeVO)) {
        print "Total wallTime of $key is $value\n"
}
while (($key,$value) = each(%njobsVOzero)) {
        print "Number of jobs with zero time for $key is $value\n"
}
while (($key,$value) = each(%njobsVOnotQueue)) {
        print "Number of jobs in different queue for $key is $value\n"
}
print "Mindate = $mindate; Maxdate = $maxdate\n";

sub parseUR_lsf
{

        my @ARRAY = split(" " , $_[0] );
        if ( (scalar(@ARRAY) > 3) && ($ARRAY[0] eq "\"JOB_FINISH\"") ) {
        #print "$_[1]:";
                my $URString = $_[0];
                my @new = quotewords(" ", 0, $URString);
                if ($new[2] >= $begints && $new[2] < $endts) {
                        $mindate = $new[2] if ($new[2] <= $mindate);
                        $maxdate = $new[2] if ($new[2] >= $maxdate);
                        my $shift1 = $new[22];
                        my $shift2 = $new[23+$shift1];
                #$urAcctlogInfo{server}=$new[16].".".$domainName;
                # attention: LSF log might already contain a complete hostname
                # (with domain name) ...
                # cross-check, instead of just adding the CE's domain name:
                        my $user=$new[11];
                        my $queue=$new[12];
                        $VO = findVO($user);
                        $njobsVO{$VO}++;
                        if ($queue ne $VO) {
                                $njobsVOnotQueue{$VO}++;
                        }
                # we now know the user's UNIX login, try to find out the group, that
                # isn't available in the LSF log file:
                        my $lrmsId=$new[3];
                        if ($new[28+$shift2] == -1) {
                # indicates that the value is not available!
                                $new[28+$shift2]=0;
                        }
                        if ($new[29+$shift2] == -1) {
                # indicates that the value is not available!
                                $new[29+$shift2]=0;
                        }
                        if ($new[10] == 0 && $new[28+$shift2] == 0 && $new[29+$shift2] == 0) {
                # job startTime is zero; happens if job was cancelled before executing?
                # in this case we consider wallTime = 0 and cpuTime = 0 is it correct?
                                $njobsVOzero{$VO}++;
                                $cpuTime = 0;
                                $wallTime = 0;
                        } else {
                                $cpuTime=int($new[28+$shift2])+int($new[29+$shift2]);
                                $wallTime = $new[2]-$new[10];     # are we sure?
                        }
                        $cpuTimeVO{$VO} += $cpuTime;
                        $wallTimeVO{$VO} += $wallTime;
                        my $start=$new[10];
                        my $end=$new[2];
                        #if ($VO eq "ops") {
                        printf OUTPUT "$new[2];$lrmsId;$user;$VO;$queue;$start;$cpuTime;$wallTime;$new[16]\n";
                        #}
                        $_[1]++;
                }
        }

}

sub findVO {

# Finding the VO from the username of the job submitter.
# The following combinations are admitted:
# 1) ^(.*)sgm$            ex: alicesgm
# 2) ^(.*)prd$            ex: aliceprd
# 3) ^(.*)sgm\d{3}$        ex: alicesgm001
# 4) ^(.*)prd\d{3}$        ex: aliceprd001
# 5) ^pil(.*)\d{3}$        ex: pilatlas001
# 6) ^pil(.*)$             ex: pilatlas
# 7) ^sgm(.*)\d{3}$        ex: sgmalice001
# 8) ^prd(.*)\d{3}$        ex: prdalice001
# 9) ^sgm(.*)$            ex: sgmalice
# 10) ^prd(.*)$            ex: prdalice
# 11) ^(.*)\d{3}$            ex: alice001

   my $VO = "unknown";
   my $user = $_[0];
   if ( $user =~ /^(.*)sgm$/ || $user =~ /^(.*)prd$/
               || $user =~ /^(.*)sgm\d{3}$/ || $user =~ /^(.*)prd\d{3}$/
               || $user =~ /^sgm(.*)\d{3}$/ || $user =~ /^prd(.*)\d{3}$/
               || $user =~ /^pil(.*)\d{3}$/ || $user =~ /^pil(.*)$/
               || $user =~ /^sgm(.*)$/ || $user =~ /^prd(.*)$/
               || $user =~ /^(.*)\d{3}$/) {
   $VO = $1;
   }
   return ($VO);
}
```

It is run every night with the following cronjob:

```
0 1 * * *     /root/lsf-analysis/INFN-T1/convert-batch_log.sh >> /root/lsf-analysis/INFN-T1/convert-lsf_log.log
```

### pbsCheck.pl

This script extract the UR information from PBS batch system log files. The initial development of the script has been made by Riccardo Brunetti[67].

```
#!/usr/bin/perl

use Text::ParseWords;
use Time::Local;
#use strict;

my $line;

($yyyy, $mm, $dd) = $ARGV[1] =~ /(\d+)-(\d+)-(\d+)/;
# calculate epoch seconds at midnight on that day in this timezone
$epoch_seconds = timelocal(0, 0, 0, $dd, $mm-1, $yyyy);
my $begints=$epoch_seconds;
print "$begints\n";

($yyyy, $mm, $dd) = $ARGV[2] =~ /(\d+)-(\d+)-(\d+)/;
# calculate epoch seconds at midnight on that day in this timezone
$epoch_seconds = timelocal(0, 0, 0, $dd, $mm-1, $yyyy);
my $endts=$epoch_seconds;
print "$endts\n";

$mindate = $endts;
$maxdate = $begints;

open(OUTPUT, "> all_pbs_acct.txt");
open(FILE, $ARGV[0]) || &error("Error: Cannot open configuration file: $ARGV[0]\n");
        while($line = <FILE>)
        {
                &parseUR_pbs($line,$i);
        }

while (($key,$value) = each(%njobsVO)) {
        print "Number of jobs of $key is $value\n";
}
while (($key,$value) = each(%cpuTimeVO)) {
        print "Total cpuTime of $key is $value\n"
}
while (($key,$value) = each(%wallTimeVO)) {
        print "Total wallTime of $key is $value\n"
}
while (($key,$value) = each(%njobsVOzero)) {
        print "Number of jobs with zero time for $key is $value\n"
}
while (($key,$value) = each(%njobsVOnotQueue)) {
        print "Number of jobs in different queue for $key is $value\n"
}
print "Mindate = $mindate; Maxdate = $maxdate\n";

sub parseUR_pbs
{
        my $start=0;
        my $cpuTime = 0;
        my $wallTime = 0;
        my $lrmsId = 0;
        my $user = "";
        my $queue = "";
        my $VO = "";
        my $subhost = "";
        my %elements = ();

        my @inarray = split(";" , $_[0] );
        if ($inarray[1] eq "E") {
            $strange = 0;
            $lrmsId=$inarray[2];
            my @inarraydata = split(" ", $inarray[3]);
            foreach $value (@inarraydata) {
                my @tmp = split("=",$value);
                $elements{$tmp[0]} = $tmp[1];
            }
            $start = $elements{"start"};
            $end = $elements{"end"};
            $user = $elements{"user"};
            $queue = $elements{"queue"};
            $VO = findVO($user);
            $njobsVO{$VO}++;
            if ($queue ne $VO) {
                $njobsVOnotQueue{$VO}++;
            }
            if (exists($elements{"resources_used.cput"})) {
                my @inarraycputime = split(":",$elements{"resources_used.cput"});
                $cpuTime = $inarraycputime[0]*3600 + $inarraycputime[1]*60 + $inarraycputime[2];
            } else {
                $strange = 1;
            }
            if (exists($elements{"resources_used.walltime"})) {
                my @inarraywalltime = split(":",$elements{"resources_used.walltime"});
                $wallTime = $inarraywalltime[0]*3600 + $inarraywalltime[1]*60 + $inarraywalltime[2];
            } else {
                $strange = 1;
            }

            $subhost = $elements{"exec_host"};

            if ($start == 0) {
                $njobsVOzero{$VO}++;
                $cpuTime = 0;
                $wallTime = 0;
            }
            if ($end >= $begints && $end < $endts && $strange==0) {
                $mindate = $end if ($end <= $mindate);
                $maxdate = $end if ($end >= $maxdate);
                $cpuTimeVO{$VO} += $cpuTime;
                $wallTimeVO{$VO} += $wallTime;
                printf OUTPUT "$end;$lrmsId;$user;$VO;$queue;$start;$cpuTime;$wallTime;$subhost\n";
            }
        }
}

sub findVO {

# Finding the VO from the username of the job submitter.
# The following combinations are admitted:
# 1) ^(.*)sgm$                  ex: alicesgm
# 2) ^(.*)prd$                  ex: aliceprd
# 3) ^(.*)sgm\d{3}$             ex: alicesgm001
# 4) ^(.*)prd\d{3}$             ex: aliceprd001
# 5) ^sgm(.*)\d{3}$             ex: sgmalice001
# 6) ^prd(.*)\d{3}$             ex: prdalice001
# 7) ^sgm(.*)$                  ex: sgmalice
# 8) ^prd(.*)$                  ex: prdalice
# 9) ^(.*)\d{3}$                        ex: alice001

        my $VO = "unknown";
        my $user = $_[0];
        if ( $user =~ /^(.*)sgm$/ || $user =~ /^(.*)prd$/
                || $user =~ /^(.*)sgm\d{3}$/ || $user =~ /^(.*)prd\d{3}$/
                || $user =~ /^sgm(.*)\d{3}$/ || $user =~ /^prd(.*)\d{3}$/
                || $user =~ /^sgm(.*)$/ || $user =~ /^prd(.*)$/
                || $user =~ /^(.*)\d{3}$/) {
        $VO = $1;
        }
        return ($VO);
}
```

## A.9 cross-check populate tables

Following are the commands used to create the different tables used to store the records present in the CSV files.

### hlr

This table contains the records retrieved from the HLR server with the *populate_hlr.sh* script and then updated with the *populate_hlr_site.sh* script. The information is already aggregated by day and VO.

```
CREATE TABLE hlr (
        DATE date not null,
        SITE char(50),
        VO char (100),
        NJOBS integer,
        CPUTIME float,
        WALLTIME float
);
```

### redeye

This table contains the records retrieved from the Redeye server with the *populate_redeye.sh* script and then updated with the *populate_redeye_site.sh* script. The information is already aggregated by day and VO and is directly used for the information shown on the web interface.

```
CREATE TABLE redeye (
        DATE date not null,
        QUEUE char(50),
        NJOBS integer,
        CPUTIME float,
        WALLTIME float,
        SITE char(50),
        primary key (DATE,QUEUE,SITE)
);
```

### batch

This table contains the records retrived from the Redeye server with the *populate_redeye.sh* script and then updated with *populate_redeye_site.sh*. Each record correspond to a single job accounted.

```
CREATE TABLE batch (
        endtime integer(32) NOT NULL,
        lrmsid integer NOT NULL,
        userid varchar(255) NOT NULL default '',
        vo varchar(255) NOT NULL default '',
        que varchar(255) NOT NULL default '',
        starttime integer(32) NOT NULL,
        cputime integer NOT NULL,
        walltime integer NOT NULL,
        subhost varchar(255) NOT NULL default '',
        site varchar(50) NOT NULL default '',
        primary key (lrmsid, starttime, vo)
) ENGINE=MyISAM;
```

## A.10 cross-check update tables

Following are the commands used to create the different tables and views used to store and access the records resulting from the processing of the previous shown tables. The last view, *daily_view_rounded*, is the one used by the web interface.

### hlr_local_view

This table contains the daily and VO aggregated records for DGAS accounting system for the local resources.

```
CREATE TABLE hlr_local_view (
        DATE date not null,
        SITE char(50),
        VO char (100),
        NJOBS integer,
        CPUTIME float,
        WALLTIME float,
        primary key (DATE,VO,SITE)
);
```

### hlr_grid_view

This table contains the daily and VO aggregated records for DGAS accounting system for the Grid resources.

```
CREATE TABLE hlr_grid_view (
        DATE date not null,
        SITE char(50),
        VO char (100),
        NJOBS integer,
        CPUTIME float,
        WALLTIME float,
        primary key (DATE,VO,SITE)
);
```

### hlr_all_view

This table contains the daily and VO aggregated records for DGAS accounting system for the Grid and local resources aggregated.

```
CREATE TABLE hlr_all_view (
        DATE date not null,
        SITE char(50),
        VO char (100),
        NJOBS integer,
        CPUTIME float,
        WALLTIME float,
        primary key (DATE,VO,SITE)
);
```

### batch_grouped

This table contains the daily and VO aggregated records for the batch log analysis.

```
CREATE TABLE batch_grouped (
        date DATE,
        VO char(255),
        queue char(255),
        njobs integer,
        cputime float,
        walltime float,
        site varchar(50)
) ENGINE=MyISAM;
```

### batch_view

This view makes a proper aggregation of the information contained in the *batch_grouped* table.

```
CREATE VIEW
        batch_view (
                date,
                vo,
                njobs,
                cputime,
                walltime,
                site
        ) as
        SELECT
                date,
                vo,
                sum(njobs),
                sum(cputime),
                sum(walltime),
                site
        FROM
```

```
                batch_grouped
        GROUP BY
                date,
                vo,
                site;
```

### daily_view_rounded

This view aggregates records coming from *hlr_ local_ view*, *hlr_ grid_ view*, *hlr_ all_ view*, *redeye* and *batch_ view*. It is the only view used by the web interface and contains all the necessary data including the calculated differences between the three systems.

```
CREATE VIEW
        daily_view_rounded (
                DATE,
                VO,
                HLR_local_jobs,
                HLR_grid_jobs,
                HLR_all_jobs,
                REDEYE_jobs,
                BATCH_jobs,
                DIFF_njobs_hlrall_redeye,
                DIFF_njobs_hlrall_batch,
                DIFF_njobs_redeye_batch,
                HLR_local_walltime,
                HLR_grid_walltime,
                HLR_all_walltime,
                REDEYE_walltime,
                BATCH_walltime,
                DIFF_walltime_hlrall_redeye,
                DIFF_walltime_hlrall_batch,
                DIFF_walltime_redeye_batch,
                HLR_local_cputime,
                HLR_grid_cputime,
                HLR_all_cputime,
                REDEYE_cputime,
                BATCH_cputime,
                DIFF_cputime_hlrall_redeye,
                DIFF_cputime_hlrall_batch,
                DIFF_cputime_redeye_batch,
                SITE
        ) as
        select
                batch_view.date as date,
                batch_view.vo as vo,
                hlr_local_view.njobs as HLR_local_jobs,
                hlr_grid_view.njobs as HLR_grid_jobs,
                hlr_all_view.njobs as HLR_all_jobs,
                redeye.njobs as REDEYE_jobs,
                batch_view.njobs as BATCH_jobs,
                round(((hlr_all_view.njobs-redeye.njobs)/redeye.njobs)*100, 2) as DIFF_njobs_hlrall_redeye,
                round(((hlr_grid_view.njobs-batch_view.njobs)/batch_view.njobs)*100, 2) as DIFF_njobs_hlrall_batch,
                round(((redeye.njobs-batch_view.njobs)/batch_view.njobs)*100, 2) as DIFF_njobs_redeye_batch,
                round(hlr_local_view.walltime, 2) as HLR_local_walltime,
                round(hlr_grid_view.walltime, 2) as HLR_grid_walltime,
                round(hlr_all_view.walltime, 2) as HLR_all_walltime,
                round(redeye.walltime, 2) as REDEYE_walltime,
                round(batch_view.walltime, 2) as BATCH_walltime,
                round(((hlr_all_view.walltime-redeye.walltime)/redeye.walltime)*100, 2) as DIFF_walltime_hlrall_redeye,
                round(((hlr_grid_view.walltime-batch_view.walltime)/batch_view.walltime)*100, 2) as DIFF_walltime_hlrall_batch,
                round(((redeye.walltime-batch_view.walltime)/batch_view.walltime)*100, 2) as DIFF_walltime_redeye_batch,
                round(hlr_local_view.cputime, 2) as HLR_local_cputime,
                round(hlr_grid_view.cputime, 2) as HLR_grid_cputime,
                round(hlr_all_view.cputime,2 ) as HLR_all_cputime,
                round(redeye.cputime, 2) as REDEYE_cputime,
                round(batch_view.cputime, 2) as BATCH_cputime,
                round(((hlr_all_view.cputime-redeye.cputime)/redeye.cputime)*100, 2) as DIFF_cputime_hlrall_redeye,
                round(((hlr_grid_view.cputime-batch_view.cputime)/batch_view.cputime)*100, 2) as DIFF_cputime_hlrall_batch,
                round(((redeye.cputime-batch_view.cputime)/batch_view.cputime)*100, 2) as DIFF_cputime_redeye_batch,
                batch_view.site
        from
                (((batch_view
                        LEFT JOIN
                hlr_local_view on (hlr_local_view.vo=batch_view.vo and hlr_local_view.date=batch_view.date and hlr_local_view.site=batch_view.site))
                        LEFT JOIN
                hlr_grid_view on (hlr_grid_view.vo=batch_view.vo and hlr_grid_view.date=batch_view.date and hlr_grid_view.site=batch_view.site))
                        LEFT JOIN
                hlr_all_view on (hlr_all_view.vo=batch_view.vo and hlr_all_view.date=batch_view.date and hlr_all_view.site=batch_view.site))
                        LEFT JOIN
                redeye on (redeye.queue=batch_view.vo and redeye.date=batch_view.date and batch_view.site=redeye.site)
        order by
        date,
        vo;
```

# Appendix B

# Scripts for Storage Accounting

## B.1 DGAS Send Record client

This is the DGAS Send Record client provided with the DGAS packages and used to send the custom storage URs to the HLR:

```
${GLITE_LOCATION}/libexec/glite-dgas-sendRecord
```

which has the following syntax:

```
DGAS Send Record client
Author: Andrea Guarise <andrea.guarise@to.infn.it>
Version:0.13.2.0
Usage:

glite-dgas-send-record <OPTIONS> [USAGE RECORD LIST]
Where options are:
-h  --help                      Display this help and exit.
-v  --verbosity <verbosity>     (0-3) default 3 maximum verbosity
-s  --server <HLR contact>  Contact string of the Site HLR.
-t  --table <HLR table>  table on the HLR.

The HLR an PA contact strings have the form: "host:port:host_cert_subject".

[voStorageRecords USAGE RECORD LIST]:
["timestamp=<int>"] "vo=<string>" "voDefSubClass=<string>"
"storage=<string>" "storageSubClass=<string>" "usedBytes=<int>"
"totalBytes=<int>"
```

## B.2 *voStorageAccounting* table

This table, created on a test HLR, contains the custom URs discussed on Chapter 5:

```
CREATE TABLE sysDefStorageAccounting (
        ID               bigint(20) not null auto_increment, key(ID),
        RecordIdentity   char(64) not null, primary key(RecordIdentity),
        GlobalFileId     char(512),
        LocalFileId      char(512),
        GlobalGroup      char(64),
        GlobalUsername   char(128),
        LocalUserId      char(64),
        Charge           float,
        Status           char(64) not null,
        Host             char(64) not null,
        SubmitHost       char(64),
        ProjectName      char(64),
        ProjectPartition char(64),
        StorageType      char(64),
        ProtocolType     char(64),
        OperationType    char(64),
        Network          int(10),
        Disk             int(10),
        TimeDuration     int(10),
        TimeInstant      int(10),
        ServiceLevel     char(64)
);
```

## B.3 retrieve_info_sys_new.py

This script generates the URs for the system tested in Chapter 5.

```
#!/usr/bin/env python

##########################################################
#
# Python client to query an ldap server (gLite top bdii)
# and retrieve info about the GlueSubClusterLogicalCPUs
# attribute value.
# Originally written by:
# Giuseppe Misurelli
# giuseppe.misurelli@cnaf.infn.it
# Peter Solagna
# peter.solagna@pd.infn.it
#
# Last modification: Oct 14th, 2009
#
#
# Adapted for the creation of UR for Storage Accounting by:
# Enrico Fattibene
# enrico.fattibene@cnaf.infn.it
# Andrea Cristofori
# andrea.cristofori@cnaf.infn.it
#
# Last modification: Mar 04th, 2011
#
##########################################################

import ldap, os, sys, time, hashlib


#
#PARSING CONF FILE
#

fconf='./retrieve_info_sys_new.conf' #temporary solution
f=open(fconf,'r')
content=f.readlines()
f.close()
string_to_exec = ""

for l in content:
    if l[0]=='#' or l.find('=') <= 0: continue
    string_to_exec += l #exec l.rstrip('\n')
exec string_to_exec # In this way it's possible to write configuration directives in more than one line. Peter.


# Main function that execute all the functions defined
def main():

    # Server bdii to bind
    server_uri = top_bdii
    global l
    l = ldap.initialize(server_uri)

    try:
        # Functions that will be invoked and executed
        out_list = infoSys_query(l)
        commands = generate_ur(out_list)
    except ldap.LDAPError, e:
        print "LDAP ERROR:"
        print e

#    for command in commands:
#        print command


def daystamp (today_yesterday):
    if today_yesterday == 1:
        unixtime = int(time.time() - 60*60*24)
    else:
        unixtime = int(time.time())
    return unixtime


# Searching for accounting information published by sites
def infoSys_query(l):

    res_dict = {}

    output_list = []
    for site_name_search in sites:
        # Querying info system for a single site.
        search_scope = ldap.SCOPE_SUBTREE
        timeout = 15
        site_used_space       = 0
        site_vo         = 0
        site_sa     = 0
        site_host     = 0
        try:
            result_id = l.search("mds-vo-name=" + site_name_search + ",mds-vo-name=local,o=grid", search_scope, search_filter, search_attribute)
            result_set = []

#           print result_id

            site_dict = {}
            n = 0
            while 1:
                site_dict[n] = {}
                result_type, result_data = l.result(result_id, 0, timeout)
                if result_data == []:
                    break
                else:
                    if result_type == ldap.RES_SEARCH_ENTRY:
                        become_list = list(result_data[0])
                        # The list contains: a first element that is the search parameters, and the second that is a dictionary with the\
                               output values.
                        grab_dict = become_list[1]
                        # We need to convert in number the strings got from ldap:
                        try:
                            int_usedspace = int(grab_dict['GlueSAUsedOnlineSize'][0])
                        except ValueError:
                            int_usedspace = 0
                        try:
                            str_vo = str(grab_dict['GlueSAAccessControlBaseRule'][0])
                        except ValueError:
                            str_vo = ''
                        try:
                            str_sa = str(grab_dict['GlueSALocalID'][0])
                        except ValueError:
                            str_sa = ''
                        try:
                            str_host = str(grab_dict['GlueChunkKey'][0])
                        except ValueError:
                            str_host = ''

                        # The site storage information:
                        site_used_space = int_usedspace
                        site_vo=str.replace(str_vo, 'VO:', '')
                        site_sa=str_sa
                        site_host=str.replace(str_host, 'GlueSEUniqueID=', '')

                        # Build a dictionary for this site:
                        site_dict[n]['Site'] = site_name_search
                        site_dict[n]['Host'] = str(site_host)
                        site_dict[n]['VO'] = str(site_vo)
                        site_dict[n]['SA'] = str(site_sa)
                        site_dict[n]['UsedSpace'] = str(site_used_space)
                        site_dict[n]['RecordIdentity'] = hashlib.sha1(site_dict[n]['Site']+site_dict[n]['VO']+str(daystamp(0))).hexdigest()
                        output_list.append(site_dict[n])
                        n += 1
        except ldap.LDAPError, e:
            print "LDAP ERROR: "
            print e

    return output_list


# Generating the usage record files
def generate_ur(input_list):
    today_stamp     = daystamp(0)
    yesterday_stamp = daystamp(1)

    try:
        time_file = open(buffer_time_instant_file, 'r')
        instant = time_file.read()
        time_file.close()
    except IOError, e:
        time_duration=0
```

```
            print "READ ERROR: "
            print e
    else:
        try:
            time_duration=today_stamp-int(instant)
        except ValueError, e:
            time_duration=0
            print "VALUE ERROR: "
            print e

    ur_line_list = []
    n = 0
    for single_dict in input_list:
        ur_line="RecordIdentity=%s GlobalFileId= LocalFileId= GlobalGroup=%s GlobalUsername= LocalUserId= Charge='' Status= Host=%s SubmitHost=\
      ProjectName=%s ProjectPartition=%s StorageType= ProtocolType= OperationType= Network='' Disk='%s' TimeDuration='%s' TimeInstant='%s'\
      ServiceLevel="%(single_dict['RecordIdentity'], single_dict['Site'], single_dict['Host'], single_dict['VO'],\
      single_dict['SA'], single_dict['UsedSpace'], time_duration, today_stamp)

        output = open(output_file_path + prefix_name + str(today_stamp) + "-" + single_dict['RecordIdentity'], 'w')
        output.write(ur_line)
        output.close()
        ur_line_list.append(ur_line)
        n += 1

    try:
        time_file = open(buffer_time_instant_file, 'w')
        time_file.write(str(today_stamp))
        time_file.close()
    except IOError, e:
        print "WRITE ERROR: "
        print e

    return ur_line_list

if __name__=='__main__':
    main()
```

Its configuration file, *retrieve_info_sys_new.conf*:

```
# Empty lines or starting withs # are skipped
# Multiple lines are allowed, respecting python syntax.


sites = ["INFN-MILANO-ATLASC","INFN-BARI"]
top_bdii = "ldap://gridit-bdii-01.cnaf.infn.it:2170"

search_filter = "objectClass=GlueSA"
search_attribute = ["GlueSAUsedOnlineSize","GlueSAAccessControlBaseRule",\ "GlueSALocalID","GlueChunkKey","GlueSAName"]

buffer_time_instant_file = "/home/GUEST/cristofori/testStorage/bufferTimeInstant"
output_file_path = "/home/GUEST/cristofori/testStorage/"
prefix_name = "testStorage-"
```

It needs a Python interpreter and the following additional libraries if not installed already:

```
python-ldap
python-hashlib
```

## B.4 push_sa-ur.sh

This script is responsible for the execution of the DGAS client and the error handling:

```
#!/bin/bash

source ./push_sa-ur.conf

/opt/globus/bin/grid-proxy-init -cert /etc/grid-security/hostcert.pem -key /etc/grid-security/hostkey.pem

for i in $SAUR_DIR/$PREFIX*; do

        echo $i

        ATTRIBUTES=`cat $i`

        $PUSH_COMMAND -v $VERBOSITY -s $HOST_NAME -t $TABLE_NAME "ID=''" $ATTRIBUTES >> $LOG_FILE
        EXIT_STATUS=$?
        if [ $EXIT_STATUS != 0 ]; then
                echo `date +"%F %T"` $EXIT_STATUS - ERROR, moving $i to $SAUR_ERRDIR dir >> $LOG_FILE_ERR
                mv $i $SAUR_ERRDIR
        else
                echo `date +"%F %T"` $EXIT_STATUS - OK, removing $i >> $LOG_FILE
                rm $i
        fi
done;
```

Its configuration file, *push_sa-ur.conf*:

```
export PUSH_COMMAND=/opt/glite/libexec/glite-dgas-sendRecord

export SAUR_DIR=/home/GUEST/cristofori/testStorage                      # Check this
export SAUR_ERRDIR=$SAUR_DIR/ERR
export PREFIX="testStorage-"                                            # Check this

export VERBOSITY=3
export HOST_NAME=dgas-test-vm01.cnaf.infn.it
export TABLE_NAME=sysDefStorageAccounting

export LOG_DIR=$SAUR_DIR/log
export LOG_FILE=$LOG_DIR/push_sa-ur.log
export LOG_FILE_ERR=$LOG_DIR/push_sa-ur_ERR.log
```

# Appendix C

# ActiveMQ test scripts

## C.1 Compilation and installation

### C.1.1 APR

APR: *apr-1.3.12.tar.gz*

- tar zxvf apr-1.3.12.tar.gz
- cd apr-1.3.12
- ./configure
- make
- make install

### C.1.2 APR-util

APR-util: *apr-util-1.3.9.tar.gz*

- tar zxvf apr-util-1.3.9.tar.gz
- cd apr-util-1.3.9
- ./configure
- make
- make install

### C.1.3 ACTIVEMQ

ACTIVEMQ: *activemq-cpp-library-3.1.0-src.tar.gz*

- tar zxvf activemq-cpp-library-3.1.0-src.tar.gz
- cd activemq-cpp-library-3.1.0
- On the CE: ./configure –prefix=/opt/activemq/lib/
- On the HLR AMQ: ./configure –prefix=/usr/local/lib/
- make
- make install
- ldconfig

## C.2 CE configurations

On the CE, the configuration file *$GLITE_LOCATION/etc/dgas_sensors.conf* has been set with those extra parameters:

```
printAsciiLog = "yes"
asciiLogFilePath = "/opt/glite/var/log/pushdAscii.log"
amqBrokerUri = "URI del broker AMQ"
transportLayer = "legacy;amq"
recordComposer = "/opt/glite/libexec/glite_dgas_recordComposer"
amqProducer = "/opt/glite/libexec/glite_dgas_hlrProducer"
dgasAMQTopic = "DGAS.TEST"
```

to enable SSL the parameter *amqProducer* has been changed with:

```
amqProducer = "openssl smime -binary -encrypt HLR_public_certificate.pem | /opt/glite/libexec/glite_dgas_hlrProducer"
```

## C.3 HLR configurations

On the HLR the configuration file *$GLITE_LOCATION/dgas-activemq-consumer.conf* has been edited with the following parameters:

```
lockFileName = "/opt/glite/var/dgas-hlr-amq-consumer.lock"
logFileName = "/opt/glite/var/dgas-hlr-amq-consumer.log"
amqBrokerUri = "URI del broker AMQ"
recordsDir = "/tmp/dgasamq/"
dgasAMQTopic = "DGAS.TEST"
hlr_user = "root"
dgas_var_dir = "/opt/glite/var/"

managerLockFileName = "/opt/glite/var/dgas-hlr-amq-manager.lock"
messageParsingCommand = "cat MESSAGEFILE"
AMQRecordsDir = "/tmp/dgasamq/"

managerLogFileName = "/opt/glite/var/dgas-hlr-amq-manager.log"

#Those parameters are the same present in dgas_hlr.conf
hlr_sql_server = "localhost"
hlr_sql_user = "root"
hlr_sql_password = "database password"
hlr_sql_dbname = "hlr database"
hlr_tmp_sql_dbname = "hlr_tmp database"
```

to enable SSL the parameter *messageParsingCommand* has been changed as following:

```
messageParsingCommand = "cat MESSAGEFILE |
  openssl smime -decrypt -inkey /etc/grid-security/hostkey.pem -keyform PEM -recip /etc/grid-security/HLR_private_key.pem"
```

## C.4 Job submission script

```
#!/bin/bash

COUNTER=0
while [  $COUNTER -lt $1 ]; do
        echo Sending the job number $COUNTER
        let COUNTER=COUNTER+1

        glite-wms-job-submit -o jobs.txt -c wms_grid_lab.conf -r dgas-test-vm03.cnaf.infn.it:8443/cream-pbs-cert -a test.jdl
        sleep 12
done
```

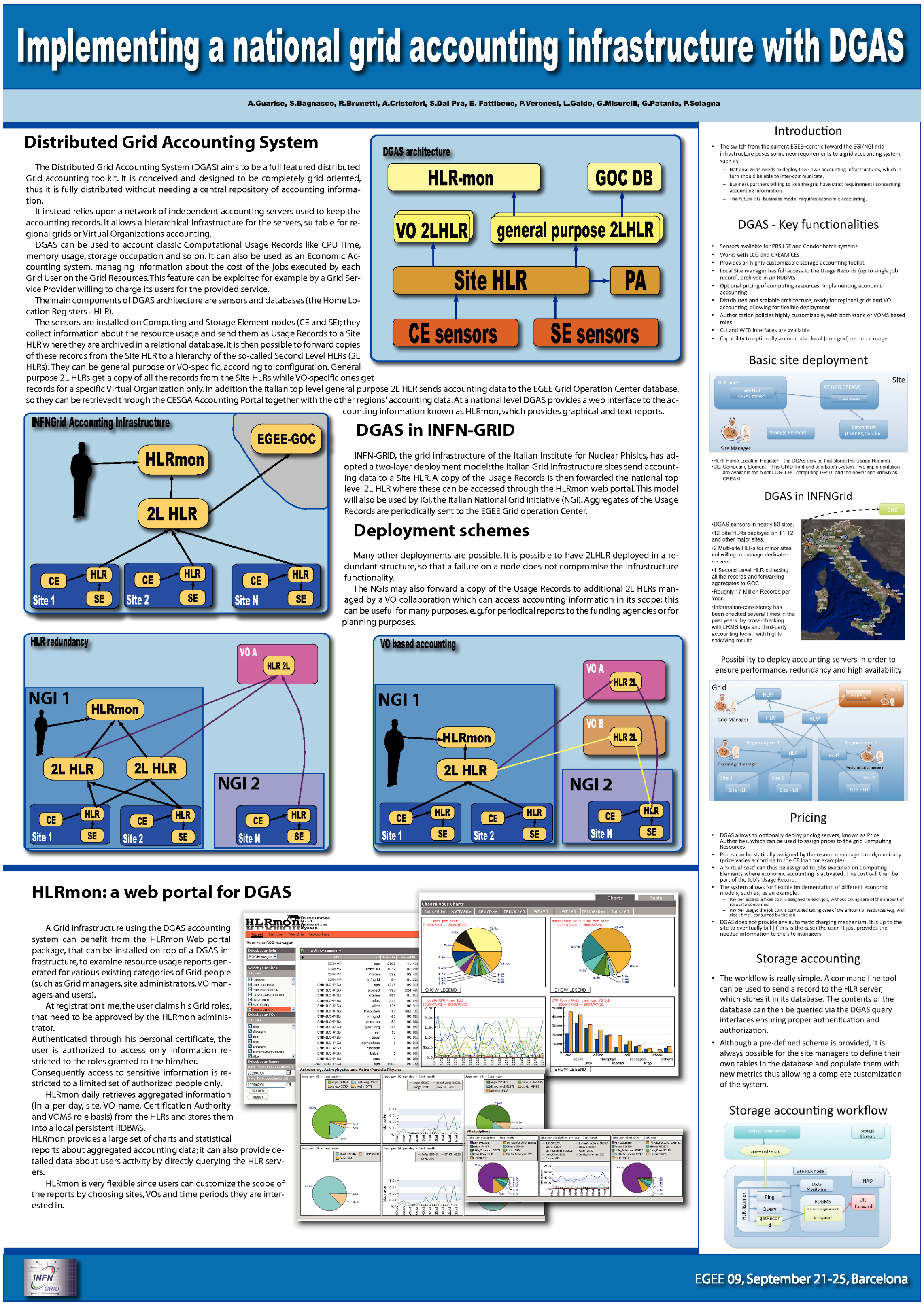
## C.5 JDL for the test job

```
[
Executable = "/usr/bin/whoami";
StdOutput = "hnam.out";
StdError = "hnam.err";
OutputSandBox = {"hnam.out", "hnam.err"};
OutputSandbox = {"stderr.log", "stdout.log"};
LBAddress = "lb007.cnaf.infn.it:9000";
]
```

# Appendix D

# Publications

Following is a collection of posters related to the accounting and, in particular, on DGAS, that have been presented in different occasions. During the last three years, in which I participated both with my personal contribution of contents and on the preparation of the poster itself.



# Implementing a national grid accounting infrastructure with DGAS

A.Guarise, S.Bagnasco, R.Brunetti, A.Cristofori, S.Dal Pra, E. Fattibene, P.Veronesi, L.Gaido, G.Misurelli, G.Patania, P.Solagna

## Distributed Grid Accounting System

The Distributed Grid Accounting System (DGAS) aims to be a full featured distributed Grid accounting toolkit. It is conceived and designed to be completely grid oriented, thus it is fully distributed without needing a central repository of accounting information.

It instead relies upon a network of independent accounting servers used to keep the accounting records. It allows a hierarchical infrastructure for the servers, suitable for regional grids or Virtual Organizations accounting.

DGAS can be used to account classic Computational Usage Records like CPU Time, memory usage, storage occupation and so on. It can also be used as an Economic Accounting system, managing information about the cost of the jobs executed by each Grid User on the Grid Resources. This feature can be exploited for example by a Grid Service Provider willing to charge its users for the provided service.

The main components of DGAS architecture are sensors and databases (the Home Location Registers - HLR).

The sensors are installed on Computing and Storage Element nodes (CE and SE); they collect information about the resource usage and send them as Usage Records to a Site HLR where they are archived in a relational database. It is then possible to forward copies of these records from the Site HLR to a hierarchy of the so-called Second Level HLRs (2L HLRs). They can be general purpose or VO-specific, according to configuration. General purpose 2L HLRs get a copy of all the records from the Site HLRs while VO-specific ones get records for a specific Virtual Organization only. In addition the italian top level general purpose 2L HLR sends accounting data to the EGEE Grid Operation Center database, so they can be retrieved through the CESGA Accounting Portal together with the other regions' accounting data. At a national level DGAS provides a web interface to the accounting information known as HLRmon, which provides graphical and text reports.

## DGAS in INFN-GRID

INFN-GRID, the grid infrastructure of the Italian Institute for Nuclear Phisics, has adopted a two-layer deployment model: the Italian Grid infrastructure sites send accounting data to a Site HLR. A copy of the Usage Records is then fowarded the national top level 2L HLR where these can be accessed through the HLRmon web portal. This model will also be used by IGI, the Italian National Grid Initiative (NGI). Aggregates of the Usage Records are periodically sent to the EGEE Grid operation Center.

## Deployment schemes

Many other deployments are possible. It is possible to have 2LHLR deployed in a redundant structure, so that a failure on a node does not compromise the infrustructure functionality.

The NGIs may also forward a copy of the Usage Records to additional 2L HLRs managed by a VO collaboration which can access accounting information in its scope; this can be useful for many purposes, e. g. for periodical reports to the funding agencies or for planning purposes.

## HLRmon: a web portal for DGAS

A Grid infrastructure using the DGAS accounting system can benefit from the HLRmon Web portal package, that can be installed on top of a DGAS infrastructure, to examine resource usage reports generated for various existing categories of Grid people (such as Grid managers, site administrators, VO managers and users).

At registration time, the user claims his Grid roles, that need to be approved by the HLRmon administrator.

Authenticated through his personal certificate, the user is authorized to access only information restricted to the roles granted to the him/her.

Consequently access to sensitive information is restricted to a limited set of authorized people only.

HLRmon daily retrieves aggregated information (in a per day, site, VO name, Certification Authority and VOMS role basis) from the HLRs and stores them into a local persistent RDBMS.

HLRmon provides a large set of charts and statistical reports about aggregated accounting data; it can also provide detailed data about users activity by directly querying the HLR servers.

HLRmon is very flexible since users can customize the scope of the reports by choosing sites, VOs and time periods they are interested in.

## Introduction

- The switch from the current EGEE–centric toward the EGI/NGI grid infrastructure poses some new requirements to a grid accounting system, such as:
  - National grids needs to deploy their own accounting infrastructures, which in turn should be able to inter-communicate.
  - Business partners willing to join the grid have strict requirements concerning accounting information.
  - The future EGI business model requires economic accounting.

## DGAS - Key functionalities

- Sensors available for PBS,LSF and Condor batch systems
- Works with LCG and CREAM CEs
- Provides an highly customizable storage accounting toolkit
- Local Site manager has full access to the Usage Records (up to single job record), archived in an RDBMS
- Optional pricing of computing resources. Implementing economic accounting
- Distributed and scalable architecture, ready for regional grids and VO accounting, allowing for flexible deployment
- Authorization policies highly customizable, with both static or VOMS based roles
- CLI and WEB interfaces are available
- Capability to optionally account also local (non-grid) resource usage

## Basic site deployment

•HLR: Home Location Register – the DGAS service that stores the Usage Records.
•CE: Computing Element – The GRID front-end to a batch system. Two implementation are available the older LCG, LHC computing GRID, and the newer one known as CREAM.

## DGAS in INFNGrid

•DGAS sensors in nearly 50 sites.
•12 Site HLRs deployed on T1,T2 and other major sites.
•2 Multi-site HLRs for minor sites not willing to manage dedicated servers.
•1 Second Level HLR collecting all the records and forwarding aggregates to GOC.
•Roughly 17 Million Records per Year.
•Information-consistency has been checked several times in the past years, by cross-checking with LRMS logs and third-party accounting tools, with highly satisfying results.

Possibility to deploy accounting servers in order to ensure performance, redundancy and high availability

## Pricing

- DGAS allows to optionally deploy pricing servers, known as Price Authorities, which can be used to assign prices to the grid Computing Resources.
- Prices can be statically assigned by the resource managers or dynamically (price varies according to the CE load for example).
- A 'virtual cost' can thus be assigned to jobs executed on Computing Elements where economic accounting is activated. This cost will then be part of the Job's Usage Record.
- The system allows for flexible implementation of different economic models, such as, as an example:
  - Pay per access: a fixed cost is assigned to each job, without taking care of the amount of resource consumed.
  - Pay per usage: the job cost is computed taking care of the amount of resources (e.g. wall clock time ) consumed by the job.
- DGAS does not provide any automatic charging mechanism. It is up to the site to eventually bill (if this is the case) the user. It just provides the needed information to the site managers.

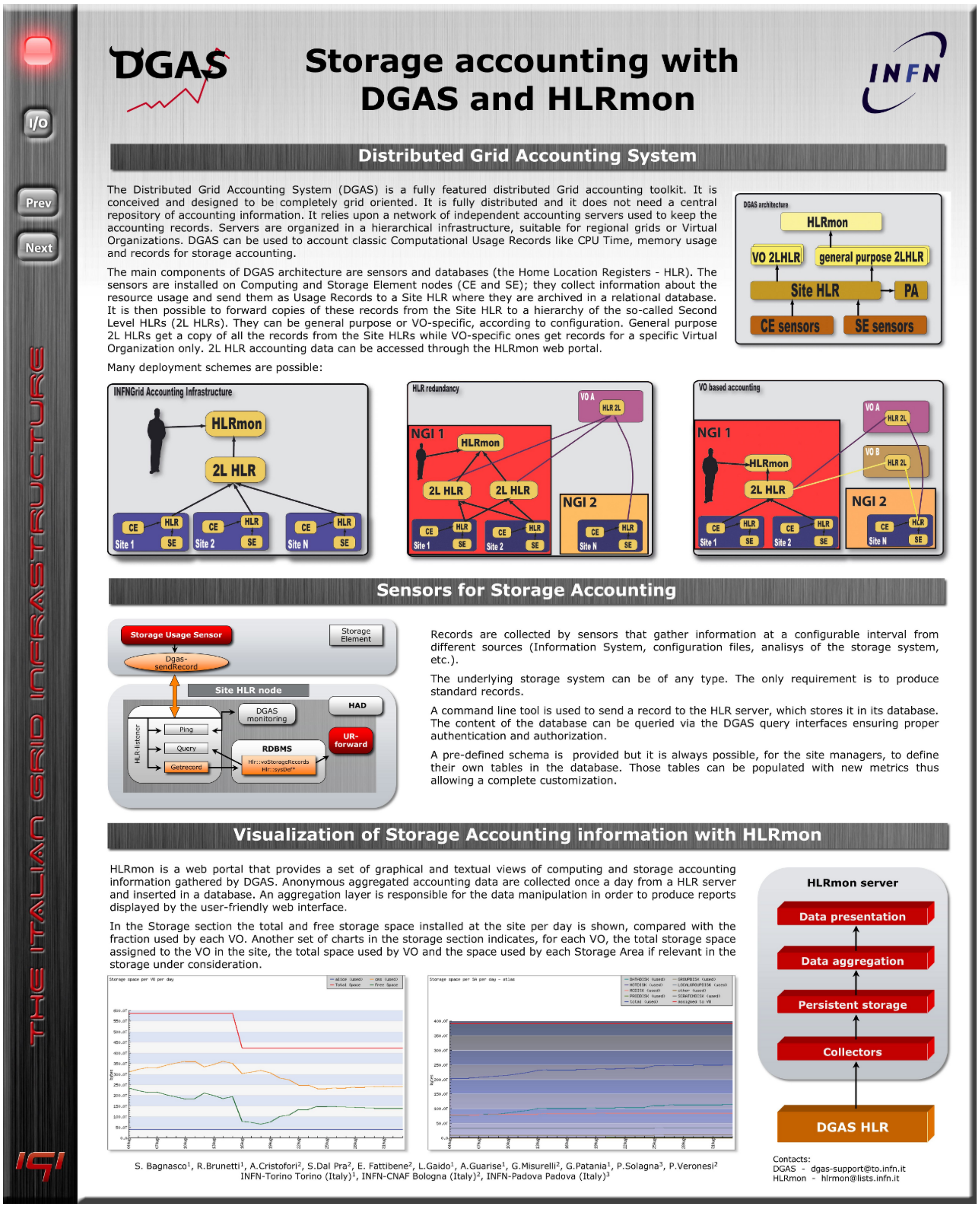
## Storage accounting

- The workflow is really simple. A command line tool can be used to send a record to the HLR server, which stores it in its database. The contents of the database can then be queried via the DGAS query interfaces ensuring proper authentication and authorization.
- Although a pre-defined schema is provided, it is always possible for the site managers to define their own tables in the database and populate them with new metrics thus allowing a complete customization of the system.

## Storage accounting workflow

EGEE 09, September 21-25, Barcelona

Figure D.1: Implementing a National grid accounting infrastructure with DGAS[26]



I/O
Prev
Next
THE ITALIAN GRID INFRASTRUCTURE
DGAS
INFN

## Storage accounting with DGAS and HLRmon

### Distributed Grid Accounting System

The Distributed Grid Accounting System (DGAS) is a fully featured distributed Grid accounting toolkit. It is conceived and designed to be completely grid oriented. It is fully distributed and it does not need a central repository of accounting information. It relies upon a network of independent accounting servers used to keep the accounting records. Servers are organized in a hierarchical infrastructure, suitable for regional grids or Virtual Organizations. DGAS can be used to account classic Computational Usage Records like CPU Time, memory usage and records for storage accounting.

The main components of DGAS architecture are sensors and databases (the Home Location Registers - HLR). The sensors are installed on Computing and Storage Element nodes (CE and SE); they collect information about the resource usage and send them as Usage Records to a Site HLR where they are archived in a relational database. It is then possible to forward copies of these records from the Site HLR to a hierarchy of the so-called Second Level HLRs (2L HLRs). They can be general purpose or VO-specific, according to configuration. General purpose 2L HLRs get a copy of all the records from the Site HLRs while VO-specific ones get records for a specific Virtual Organization only. 2L HLR accounting data can be accessed through the HLRmon web portal.

Many deployment schemes are possible:

### Sensors for Storage Accounting

Records are collected by sensors that gather information at a configurable interval from different sources (Information System, configuration files, analisys of the storage system, etc.).

The underlying storage system can be of any type. The only requirement is to produce standard records.

A command line tool is used to send a record to the HLR server, which stores it in its database. The content of the database can be queried via the DGAS query interfaces ensuring proper authentication and authorization.

A pre-defined schema is provided but it is always possible, for the site managers, to define their own tables in the database. Those tables can be populated with new metrics thus allowing a complete customization.

### Visualization of Storage Accounting information with HLRmon

HLRmon is a web portal that provides a set of graphical and textual views of computing and storage accounting information gathered by DGAS. Anonymous aggregated accounting data are collected once a day from a HLR server and inserted in a database. An aggregation layer is responsible for the data manipulation in order to produce reports displayed by the user-friendly web interface.

In the Storage section the total and free storage space installed at the site per day is shown, compared with the fraction used by each VO. Another set of charts in the storage section indicates, for each VO, the total storage space assigned to the VO in the site, the total space used by VO and the space used by each Storage Area if relevant in the storage under consideration.

S. Bagnasco[1], R.Brunetti[1], A.Cristofori[2], S.Dal Pra[2], E. Fattibene[2], L.Gaido[1], A.Guarise[1], G.Misurelli[2], G.Patania[1], P.Solagna[3], P.Veronesi[2]
INFN-Torino Torino (Italy)[1], INFN-CNAF Bologna (Italy)[2], INFN-Padova Padova (Italy)[3]

Contacts:
DGAS - dgas-support@to.infn.it
HLRmon - hlrmon@lists.infn.it

Figure D.2: Storage accounting with DGAS and HLRmon[66]

# Bibliography

[1] The nist definition of cloud computing. Website. "http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc".

[2] George D. Greenwade. Egee-iii project publishable summary. Website, 2010. "http://www.eu-egee.org/fileadmin/documents/publications/EGEEIII_Publishable_summary.pdf".

[3] Ibm solutions grid for business partners: Helping ibm business partners to grid-enable applications for the next phase of e-business on demand., 2002. U.S.A.

[4] I. Foster. What is the grid? a three point checklist, July 2002.

[5] What is grid computing? "http://www.gridcafe.org/version1/whatisgrid/whatis.html".

[6] Enabling desktop grids for e-science. "http://www.edges-grid.eu/".

[7] Unicore uniform interface to computing resources. "http://www.unicore.eu/".

[8] European grid initiative design study. Website. "http://web.eu-egi.eu/".

[9] What is egi? Website. "http://web.eu-egi.eu/about/about/".

[10] Registered egee vos. Website. "http://cic.gridops.org/index.php?section=home&page=volist".

[11] Production total number of countries connected to egi per date. Website. "http://www3.egee.cesga.es/gridsite/metrics/CESGA/tree_rcs.php".

[12] Egi accountin portal. Website. "http://www3.egee.cesga.es/gridsite/accounting/CESGA/egee_view.php".

[13] Umd roadmap. Website. "https://documents.egi.eu/secure/ShowDocument?docid=272".

[14] Virtual organization membership service. Website. "https://twiki.cnaf.infn.it/twiki/bin/view/VOMS/WebHome".

[15] Genius portal - overview. Website. "http://egee.cesnet.cz/cms/export/sites/egee/en/user/genius-guide.pdf".

[16] glite 3.1 user guide. Website. "https://edms.cern.ch/file/722398/1.3/gLite-3-UserGuide.html".

[17] Berkeley database information index. Website. "https://twiki.cern.ch/twiki/bin/view/EGEE/BDII".

[18] Welcome to the globus toolkit homepage. Website. "http://www.globus.org/toolkit/".

[19] Cream ce home page. Website. "http://grid.pd.infn.it/cream/".

[20] T. Sandholm, P. Gardfjaell, E. Elmroth, L. Johnsson, and O. Mulmo. An OGSA-based accounting system for allocation enforcement across HPC centers. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 279–288. ACM, 2004.

[21] R. Byrom, R. Cordenonsi, L. Cornwall, M. Craig, A. Djaoui, A. Duncan, S. Fisher, J. Gordon, S. Hicks, D. Kant, et al. APEL: An implementation of Grid accounting using R-GMA. In *UK e-Science All Hands Conference*, 2005.

[22] About us | deploying and unifying the nmr e-infrastructure in system biology. Website. "http://www.e-nmr.eu/eNMR-about-us".

[23] Dgas distributed grid accounting system. Website. "http://www.to.infn.it/dgas/".

[24] Yaim home. Website. "http://www.yaim.info/".

[25] Dgas guide. Website. "http://www.to.infn.it/grid/INFNGRID/TESTING/TESTING/files/Documentation/DGAS-guied_0_6_1.htm".

[26] A. Guarise; S. Bagnasco; R. Brunetti; A. Cristofori; S. Dal Pra; E. Fattibene; P. Veronesi; L. Gaido; G. Misurelli; G. Patania; P. Solagna. Implementing a national grid accounting infrastructure with dgas. Proceeding of EGEE'09 Conference; Barcellona 21st to 25th September 2010.

[27] SD Pra, E. Fattibene, G. Misurelli, F. Pescarmona, and L. Gaido. HLRmon: a role-based grid accounting report web tool. In *Journal of Physics: Conference Series*, volume 119, page 052012. IOP Publishing, 2008.

[28] Mysql. Website. "http://dev.mysql.com/".

[29] Hlrmon home page. Website. "https://dgas.cnaf.infn.it/hlrmon/".

[30] Monitoring@cnaf. Website. `"http://tier1.cnaf.infn.it/monitor/"`.

[31] Check incrociato dgas - redeye - batch. Website. `"https://cert-wms-05.cnaf.infn.it:8443/cross_check_INFN_CNAF_LHCB/"`.

[32] Service status details for service group dgas. Website. `"https://gstore.cnaf.infn.it/nagios/cgi-bin/status.cgi?servicegroup=DGAS&style=detail"`.

[33] S. Andreozzi, S. Burke, L. Field, S. Fisher, B. Kónya, M. Mambelli, JM Schopf, M. Viljoen, A. Wilson, and R. Zappi. GLUE Schema Specification–version 1.3. *OGF: https://forge. gridforum. org/sf/go/doc14185*, 2005.

[34] Storage accounting - gridppwiki. Website. `"http://www.gridpp.ac.uk/wiki/StorageAccounting"`.

[35] Hlrmon home page. Website. `"https://dgas.cnaf.infn.it/hlrmon/"`.

[36] F. Scibilia. Accounting of Storage Resources in gLite Based Infrastructures. *16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007), Paris*, pages 273–278.

[37] Bittorrent.org. Website. `"http://www.bittorrent.org/beps/bep_0003.html"`.

[38] Amazon elastic compute cloud (amazon ec2). Website. `"http://aws.amazon.com/ec2/"`.

[39] Amazon s3 pricing. Website. `"http://aws.amazon.com/s3/pricing/"`.

[40] M. Cao, T.Y. Tso, B. Pulavarty, S. Bhattacharya, A. Dilger, and A. Tomas. State of the art: Where we are with the ext3 filesystem. In *Proc. of the Linux Symposium*. Citeseer.

[41] Xfs: A high-performance journaling filesystem. Website. `"http://oss.sgi.com/projects/xfs/"`.

[42] General parallel file system. Website. `"http://www-03.ibm.com/systems/software/gpfs/"`.

[43] Lustre. Website. `"http://www.lustre.org/"`.

[44] dcache - main page. Website. `"http://www.dcache.org/"`.

[45] The perfectly normal file system. Website. `"http://www-pnfs.desy.de/"`.

[46] Chapter 25. accounting. Website. `"http://www.dcache.org/manuals/Book-1.9.10/cookbook/cb-accounting.shtml"`.

[47] Postgresql: The world's most advanced open source database. Website. `"http://www.postgresql.org/"`.

[48] Chapter 22. postgresql and dcache. Website. "http://www.dcache.org/manuals/Book-1.9.5/cookbook/cb-postgres.shtml".

[49] Dpm - grid data management. Website. "https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm".

[50] Dpm log file tracing. Website. "http://www.dcache.org/manuals/Book/start/in-additional.shtml".

[51] Graphtool overview. Website. "http://t2.unl.edu/documentation/graphtool".

[52] Dpm monitoring. Website. "http://www.gridpp.ac.uk/wiki/DPM_Monitoring".

[53] Ral tier1 castor accounting. Website. "http://www.gridpp.ac.uk/wiki/RAL_Tier1_CASTOR_Accounting".

[54] home [storm]. Website. "http://storm.forge.cnaf.infn.it/home".

[55] Features. Website. "http://www.to.infn.it/dgas/features/features.html".

[56] Dgas guide. Website. "http://www.to.infn.it/grid/INFNGRID/TESTING/TESTING/files/Documentation/DGAS-guied_0_6_1.htm".

[57] Usage of glue schema v1.3 for wlcg installed capacity information. Memo. "https://twiki.cern.ch/twiki/pub/LCG/WLCGCommonComputingReadinessChallenges/WLCG_GlueSchemaUsage-1.8.pdf".

[58] Storage. Website. "http://web.infn.it/argo/index.php/produzione-/storage".

[59] Welcome to scientific linux (sl). Website. "https://www.scientificlinux.org/".

[60] Apache activemq. Website. "http://activemq.apache.org/".

[61] Using activemq at cern. "http://fusesource.com/collateral/download/82/".

[62] Main page - kvm. Website. "http://www.linux-kvm.org/".

[63] libvirt virtualization api. Website. "http://wiki.libvirt.org/page/Virtio".

[64] Infngrid installation and configuration guide for glite 3.2 sl5 x86_64. Website. "http://igrelease.forge.cnaf.infn.it/doku.php?id=doc:guides:install-3_2".

[65] M. Bencivenni; R. Brunetti; A. Cristofori; E. Fattibene; L. Gaido; A.Guarise; G. Patania. Dgas implementation of activemq transport mechanism. EGI User Forum 2011, Vilniuls, Lithuania from the 11st to 15th April 2011.

[66] S. Bagnasco; R. Brunetti; A. Cristofori; S. Dal Pra; E. Fattibene; L. Gaido; A. Guarise; G. Misurelli; G. Patania; P. Solagna; P. Veronesi. Storage accounting with dgas and hlrmon.

[67] Riccardo Brunetti. Infn torino.