

An Accounting System Prototype for the Grid of the ATLAS Experiment at CERN

Raquel Pezoa^{1,2,*}; Julia Andreeva²,
Benjamin Gaidioz², Gilbert Poulard², Ricardo Rocha², Luis Salinas^{1,3}

¹ Informatics Department, Universidad Técnica Federico Santa María, Valparaíso, Chile

² European Organization for Nuclear Research, CERN, Geneva, Switzerland

³ Center for Technological Innovation on High Performance Computing,
Universidad Técnica Federico Santa María, Valparaíso, Chile

Raquel.Pezoa@inf.utfsm.cl Julia.Andreeva@cern.ch Benjamin.Gaidioz@cern.ch
Gilbert.Poulard@cern.ch Ricardo.Rocha@cern.ch Luis.Salinas@usm.cl

Abstract

This paper describes an Accounting System for the grid of the ATLAS Experiment developed by the first author during her stays at CERN, Switzerland, in 2007 and 2008.

ATLAS is the largest high energy physics experiment ever undertaken in the history of Science and is supposed to enter into production on September 10th, 2008. It will produce huge volumes of data, of the order of 10 PB per year. ATLAS uses grid technology to distribute, store and analyze these immenses amounts of data.

The ATLAS grid is composed by three different sub-grid infrastructures. Thus, it is a *multigrid environment*. Each sub-grid contains its own Accounting System to give account of the usage of the component grid resources, such as CPU times, wall clock times, number of jobs, storages, etc.

The Accounting System described in this paper provides a global and updated vision of the usage of the grid resources, retrieving data from the different accounting sources of the ATLAS grid.

1 CERN and the LHC

CERN, the **European Organization for Nuclear Research**,¹ is one of the world's largest and most renowned centres for scientific research in high energy physics. Its business is fundamental physics, finding out what the Universe is made of and how it works. At CERN, the world's largest and most complex scientific instruments are used to study the very basic constituents of matter – the fundamental particles. By studying what happens when particles collide at very large energy levels, physicists learn about the most fundamental laws of nature [2].

The Large Hadron Collider (LHC) [3] is a gigantic scientific instrument located near Geneva, Switzerland. The LHC is currently in its last construction and testing phases, as by August 2008, just before

*Corresponding author. Postal address: Raquel Pezoa, Informatics Department, UTFSM, Avenida Espaa 1620, Casilla 110-V, Valparaso, Chile. The present paper contains parts of the thesis research of Raquel Pezoa to be submitted to the Universidad Técnica Federico Santa María, Valparaso, Chile, in partial fulfillment of the requirements for the M.Sc. degree in Informatics.

¹The name is derived from the French acronym for “Conseil Européen pour la Recherche Nucléaire”, or European Council for Nuclear Research, a provisional body founded in 1952 with the mandate of establishing a world-class fundamental physics research organization in Europe. At that time, particle physics research concentrated on understanding the inside of the nucleus of the atom, hence the word “nuclear”.

entering into operation in September 2008. It has the form of a large ring of 27 Km length and lies some 100 meters underground, below the Franco-Swiss border to the west of Geneva, at the foot of the Jura mountains, in front of the Alps. The LHC is a particle accelerator which brings protons and ions into head-on collisions at energies much higher than ever achieved before. This will allow scientists to penetrate still further into the structure of matter and recreate the conditions prevailing in the early universe, just after the "Big Bang". The LHC hosts four experiments: ALICE, ATLAS, CMS and LHCb.

Each experiment has its own system of detectors. Both the LHC and the detectors have been developed and constructed through a world wide cooperation between research groups from around 35 countries. Proton-proton collisions should attain energy levels of 14 TeV. This energy level will make the LHC the most powerful particle accelerator in the world for several years. The LHC will create conditions equivalent to those prevailing in the early universe, some 14 to 10 seconds after the "Big-Bang". The energy of 14 TeV developed by the proton-proton collisions is high enough to produce elementary reactions between quarks and gluons. The LHC can be operated with heavy ions as well, attaining energy levels as large as 1150 TeV per collision. The resulting experimental conditions will allow reactions similar to those produced some 10 to 6 seconds after the "Big-Bang", that is, short before the so called QCD phase transition.

The detectors will produce huge amounts of data related to the myriads of particles –known and unknown– resulting from the collisions. This immense volume of data must be handled by the computing and information systems integrated into the LHC.

In this paper we are concerned with the grid around the ATLAS experiment and the grid system designed for the analysis, distribution and storage of the experimental data. This grid is called ATLAS grid in the sequel.

2 ATLAS Distributed Computing

ATLAS (**A** Toroidal **LHC** Apparatu**S**) is a particle physics experiment that will explore the fundamental nature of matter and the basic forces that shape our universe.

ATLAS is one of the largest collaborative efforts ever attempted in the physical sciences. This experiment involves some 1900 physicists (including some 400 students) from more than 164 universities and laboratories in around 35 countries. [4]

The search for new fundamental particles in the ATLAS experiment is expected to produce about 10 Petabytes (10×10^{15} Bytes) of data per year. This huge amount of data must be handled both during online data acquisition and during off-line processing of the events. The experiment will start producing significant data already during the last weeks of August 2008.

In order to handle such a huge amount of data, the ATLAS computing model [1] has decisively contributed to the development of the *computer grid paradigm* and of hierarchical computing with high degree of decentralization and sharing of resources. Grid computing technology is fundamental in order to be able to analyze the immense amounts of data to be produced.

The ATLAS computing model is based on a worldwide computing grid infrastructure that uses a set of hierarchical tiers: Tier-0, Tier-1, Tier-2 and Tier-3. The sites belonging to these tiers are also grouped into clouds. Thus, a rather complex network emerges, which is usually referred to as the ATLAS *topology*. The ATLAS grid has around 177 sites providing 29856 CPU's and an storage capacity of 11368 TB [19].

The ATLAS grid must have an Accounting System in order to provide the administrators with a global view of the usage of the computational resources, so that they are at any moment aware of the consumption of the resources. This allows a proper and efficient working of the ATLAS grid and also provides useful information to perform accounting and auditing tasks.

3 Grid Accounting

In a grid environment, the computing resources, the application data and the users belonging to a Virtual Organization (VO) are distributed. Jobs submitted by the users may be sent to computing resources close to the data or may go to remote resources with available job slots, thus reducing queue times. As a consequence, jobs that run on a grid environment must be properly accounted for. The usage of resources (CPU time, wall-clock time and memory) and the resources provided by sites, all as a function of time, can then be easily determined as well.[11]

The data of the usage of the grid resources, which includes CPU times, wall clock times, number of jobs, storages, etc., is generically called *accounting data*.

The ATLAS grid is composed by three different sub-grid infrastructures: EGEE [5], OSG [6] and NDGF.[13] Thus, the ATLAS grid is a *multigrid environment*. Each sub-grid contains its own middleware and hence, its own Accounting System, which are described as follows:

APEL [12, 8] (**A**ccounting **P**rocessor for **E**vents **L**ogs) is the accounting tool used by EGEE, which filters the accounting information from the log files generated by a site and build accounting records. The accounting data stored in APEL uses a fine-grained schema and is used to describe accounting information at the job level i.e. cpu time, wall clock time, memory, dn, vo, etc.

The collection of accounting records is done through R-GMA[15], an implemtation of the Grid Monitoring Architecture (GMA) proposed by the GGF (Global Grid Forum) [16]. GMA models has a set of *consumers*, which request the information, *producers*, which provides the information, and a *registry*, which allows the communication between producers and consumers. The WLCG/EGEE sites publishes their own accounting data using a *producer* and its locally assigned R-GMA server. To collect data from all participating sites, the data is streamed to a centralized database via a secondary producer.

APEL processes gatekeeper and batch system logs to produce accounting records. Currently supports PBS and LSF batch systems but can be easily extended to other batch systems as well. APEL systems allows the visualization of the accounting data in a web application and generates summaries of resource usage of all the EGEE sites.

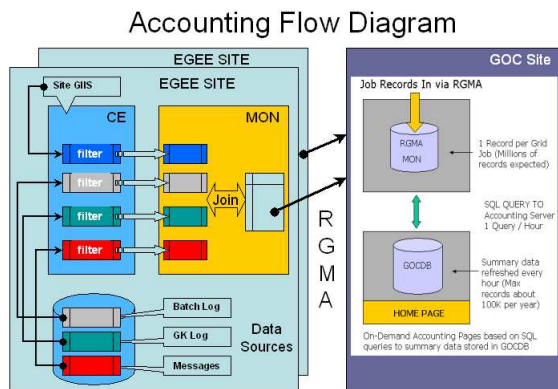


Figure 1: Data collection process done by APEL.

GRATIA [9] is the accounting system used by OSG, which designs and deploys robust, scalable, trustable and dependable grid accounting, publishes an interface to the services and provides a reference implementation.

This accounting system is composed by four main parts:

- Collection of accounting data.
- Interface between collectors and the accounting database.
- Sets of distributed data stores containing the accounting information.
- Interface to publish the accounting information containing in the databases.
- A presentation layer giving access to report and aggregates information.

SGAS [10] is the accounting system used by NDGF, and it considers the following main components: users, VOs, projects, resources, bank services, log and usage services. The last ones hold detailed information about the resources consumed (Usage Records URs), job status, input data, output data, etc., about the jobs submitted by the VO users.

When a user sends a job through the resource broker, the *accounting manager* (AM) controlling a *job manager* (JM) is started to manage the job. The user's delegated credentials as well as the job request are forwarded to the AM, which ensures that the specified VO bank is trusted and before proceeding and forwarding the request to the JM. After the job has completed the AM gathers information about the resources consumed by the user's job and records this in a Usage Record (UR).

4 The ATLAS Accounting System

We have described the three accounting systems belonging to each sub-grid of ATLAS. Currently, APEL publishes accounting data of EGEE and OSG sites in a web application, but still there is the need to have a global view of the usage of the all resources and the visualization should be available in different formats and structured as the ATLAS topology, considering the different tiers and clouds that form the whole ATLAS grid. Furthermore, the existing systems, APEL, GRATIA and SGAS, store accounting data for all LHC experiments. More precisely, they are not restricted exclusively to the ATLAS experiment. In fact, ATLAS did not have its own system for deploying the relevant information as required by the *managers of the experiment*, so that a real need for an ATLAS specific Accounting System existed. The prototype of such an Accounting System, implemented mainly by the first author during her 2007 stay at CERN and described in the present paper, satisfies that need.

The *ATLAS Accounting System* allows to retrieve accounting data from different sources –from the different sub-grids– and exposes the information to the users in different formats, providing a "topological view", and also allows to add new sources of information in an easy way due to the modular implementation of the system [20].

4.1 Dashboard Framework

The ATLAS Accounting System was developed using the Dashboard Framework [7] whose main goal is to collect, store and present to users information coming from the most distinct sources in a well defined way. It focus on the different grid entities (jobs, datasets and services in general) and tries to get from them relevant messages concerning interesting events.

The framework is implemented in Python, uses the HTTP server and its `mod_python` extension (Apache module that embeds the python interpreter within the server) [14]. Fig. 2 shows the different components each focusing on one of the the three actions mentioned above (jobs, datasets and services in general).

The Dashboard framework has different modules which can be built and packaged in an unique and straightforward way. The *arda.dashboard.apel* is the new module developed that implements ATLAS Accounting System application.

The accounting data is retrieved using the *apel.collector* service, which gets the information from an accounting data source, and then it stores it in a local database.

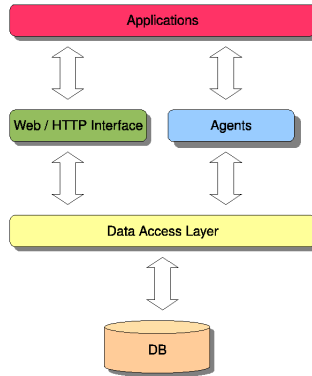


Figure 2: Dashboard Components.

4.2 Database Schema

The schema of the database is simple, it has three tables: *sumCPU*, *site* and *InsertRecord*. The first one stores ATLAS's accounting data, the *site* table contains attributes related with the topology and site of the ATLAS grid, and finally, *InsertRecord* stores the timestamp of each insertion into the *sumCPU* table.

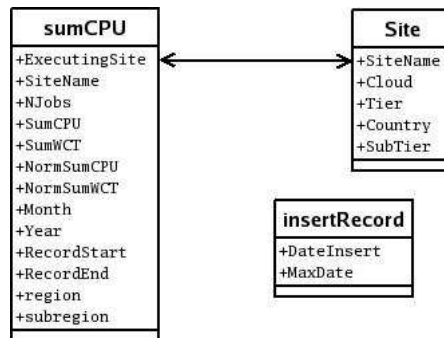


Figure 3: Database schema.

The primary key of the *sumCPU* table has the following fields: *ExecutingSite*, *LCGUserVO*, *Month* and *Year*, and this makes possible to maintain aggregated monthly accounting data for each site (belonging to a *VO*, in this case always ATLAS *VO*).

The accounting data retrieved from the different sources can have different *granularity*. For instance, APEL has a *monthly granularity*. This means that the records with aggregated data to which APEL gives access, have aggregation coverages of one month each. More precisely, APEL filters the log files produced for each site, to build the accounting records and then continuously aggregates the accounting data of each job sent to the grid. For instance, a particular APEL accounting record can represent that in June 2008 site A used, say, 272394 hours of CPU, and in the same month site B used, say, 1055 hours of CPU.

On the other hand, GRATIA allows the access of accounting records in a per *job granularity*. For instance, job X sent by Site A used, say, 2 hours of CPU on June 24th, 2008. The ATLAS Accounting System maintains a monthly granularity (like APEL). Thus, when accounting data is retrieved from GRATIA, it is necessary to process this data in a way that insures that the output granularity is the same as the one defined in the database schema. (Fig. 3).

4.3 *arda.dashboard.apel* Module Structure

The *arda.dashboard.apel* is the dashboard module that implements the ATLAS Accounting System, and it has the following files and directories structure:

| | |
|------------------|---|
| setup.py | Main distutils setup file |
| /config (dir) | Package configuration files. Usually each module will have additional configuration files stored in this area, but at least one directory following the package name should exist |
| /doc (dir) | Holds the package documentation. This includes different sub-directories, namely: guides, for user guides like this one; man, for command line tool man pages |
| /lib (dir) | All the package libraries should go here. |
| /templates (dir) | Optional directory holding the web application static files (web pages, xsl stylesheets, images) |
| /build (dir) | Temporary directory created for build files |

The *setup.py* file is a configuration file that contains the definitions of the packages (located in the */lib* directory) and files related with the application's interface of the *arda.dashboard.apel* module. As follows, we describe the main packages of this module (the complete documentation can be found in [17]).

- *dashboard.dao.oracle.apel*: This package allows the management of the application's database.
- *dashboard.http.actions.apel*: This package contains the entities in charge of request of data playing as the interface between the user (or other application) and the data layer.
- *dashboard.http.views.apel*: It allows the display of accounting data in a particular format. In particular, it can display the accounting information as pie plots.
- *dashboard.collector.apel*. This package allows to retrieve the accounting data from an external sources of information (APEL, GRATIA) and store this data into the own application's database. The collector is constantly retrieving and updating the accounting information. In this application it is necessary to create one collector for each source of information.

4.4 Accounting Data Collection

Currently, the ATLAS Accounting System has implemented two services to collect accounting data in an automatic and periodic manner.

APEL Collector This agent gathers accounting data from APEL. The agent request the data using the *MySQLdb* interface (see Fig. 4), asking for the data directly to the APEL database. Once the data is retrieved, the APEL collector must process this data to obtain the proper format, and later the data is stored into the local database. This process is done periodically and each time that the collector retrieves the data from APEL, checks the timestamps of the previous collection, in order to retrieve only the new accounting data, thus avoiding the retrieval of the complete APEL accounting data (which is a huge volume of data).

The following code is an extract of the class (written in Python language) that implements the APEL collector:

```
def run(self):
    while self.status() is not None: (1)
        ctx = DAOContext.getDAOContext() (2)
        dao = DAOFactory.getDAOFactory().createApelDAO(ctx)
        dao.fillAccountingTable(accData) (3)
        self._logger.info("sumcpu table has been filled")
        new_sites=dao.fillSiteTable('sumcpu') (4)
```

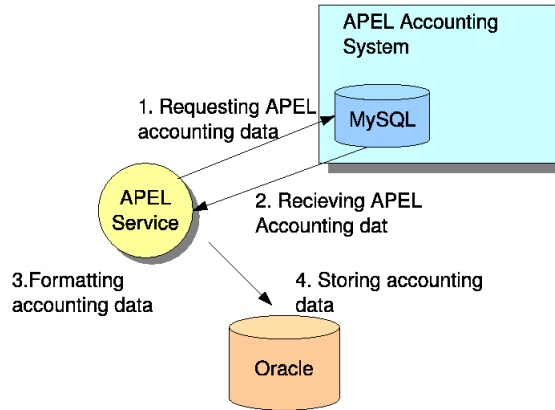


Figure 4: APEL Collector.

```

self._logger.info("New sites into site table:%s " % new_sites )
dao.fillRecordInsertTable('sumcpu','insertrecord')
self._logger.info("InsertRecord table has been filled")
ctx.commit()
ctx.destroy()
sleep(self.runInterval)
... ..

```

The run method (1) allows to loop until a stop is requested, making possible to get the data in a periodic way. Later, by means of (2), a DAO context is created, that is, an object is created in order to establish communication with the data layer. In (3), the *fillAccountingTable* method retrieves accounting data from APEL and then inserts into the local database (sumCPU table, see Fig. 3). The *fillSiteTable* adds the new sites to the *site* table and finally the *fillRecordInsertTable* stores the timestamp of each insertion into the sumCPU table.

Gratia Collector This agent gathers accounting data from the GRATIA. The request is done via an HTTP request, using the *URLClient* object, provided by the Dashboard framework. In contrast to APEL Collector, where we have direct access to the APEL database, here the GRATIA agent establishes the connection with an specified URL, and through a service provided by GRATIA, it is possible to retrieve the accounting data in a CSV format, for which, is necessary to format the data to insert it later in the local database of the ATLAS Information System. Fig. 5 depicts the mechanism to collect the GRATIA accounting data.

```

def csvQueryLoop(self):
    while (firstDate <= lastDate:):
        year=firstDate.year
        month=firstDate.month
        query='select sum(a.WallDuration) as sumwct, sum(a.CpuUserDuration) as
sumcpuuser, sum(a.CpuSystemDuration) as sumcpu,
sum(a.Njobs) as njobs, a.Status as status,
b.ReportedSiteName as ExecutingSite, min(a.StartTime) as
recordstart,max(a.EndTime) as recordend from JobUsageRecord a,
JobUsageRecord_Meta b where a.dbid=b.dbid and a.VOName like "%atlas%" and
a.EndTime >= "' +str(year)+'-'+str(month)+'-'+str('01')+' "'
and a.EndTime <= "' +str(year)+'-'+str(month)+'-'+str('31')+' "'
group by b.ReportedSiteName, a.Status'
self.getCSVFile(query)

def run(self):
    while self.status() is not None:
        self.csvQueryLoop()
        sleep(self.runInterval)

```

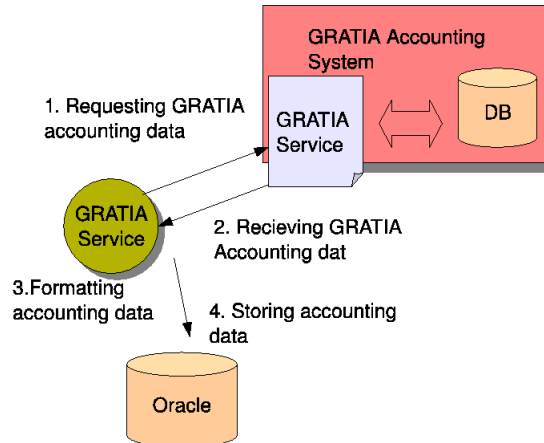


Figure 5: GRATIA Collector.

The above code is a fragment of the GRATIA collector implementation. In the same way as with the APEL collector, the run method (1) makes possible to get the data in a periodic way. Later, the *cvsQueryLoop* method (2) performs the data retrieval, building the query to obtain aggregated data and calling the *getCSVFile* method, which finally connects with GRATIA to retrieve the data.

We have described two collectors that allow the retrieval of accounting data from the APEL and GRATIA systems. To collect data from new sources of information, it is necessary to create a new collector, which has to retrieve the information and then process it in such a way that the data can be stored in the our database, using the schema described in Fig. 3. The important point here is to preserve the *monthly granularity* of the accounting data, this is the records of accounting data stored in the local database must maintain the primary key: ExecutingSite, VO, Month and Year.

4.5 Results

The ATLAS Accounting System is a prototype that retrieves accounting data from different sources, allowing to have a global vision of the usage of the resources of the ATLAS grid.

To visualize the accounting data the ATLAS Accounting System has a web application [18], which allows to accounting data of the ATLAS experiment.

The ATLAS Accounting System can display accounting data such as: CPU time, normalized CPU time, wall clock time, normalized wall clock time, number of jobs and CPU power, and all these can be visualized per site, country, tier, cloud and region –ATLAS grid topology concepts.

Furthermore, the data can be retrieved in different formats (XML and CSV) because of the modularity of the system (given by the Dashboard Framework).

5 Conclusions and Future Work

In this paper we red have presented an accounting system prototype for the grid of the ATLAS experiment, at CERN. ATLAS is one of the most important particle physics experiments in the history of Science and will start to produce huge volumes of data during the first weeks of September 2008. ATLAS uses the grid technology to analyze and store this data. The ATLAS grid is composed by three different sub-grids (EGEE, OSG and NDGF), and each one has its own accounting system. APEL, the accounting system of the EGEE sub-grid, has a model with the following main componetes: *producer*, *consumer*, *registry*, where a site publish its accounting data using a producer and its locally assigned R-GMA server.

Thereafter, the data is collected for a specialized producer, which stores this data into a centralized database.

GRATIA, the accounting system of the OSG sub-grid, has a similar mechanism to collect and publish the accounting data as APEL. In SGAS, the accounting system of NDGF sub-grid, the *accounting manager* (AM) is the entity in charge of gather information about the resources consumed and then of build the Usage Records (UR).

The ATLAS Accounting System works one level above the previously described accounting systems, because the main idea of this system, is to provide a global view of the accounting data of the ATLAS grid (not separated per sub-grid) and also to provide the information in different formats and specific views (considering the ATLAS topology) in order to be used mainly by the *managers* of ATLAS grid, but because of the flexibility of the system, this may be used also by other users or other applications of the ATLAS grid.

Currently, the ATLAS Accounting System is a prototype, which performs the most needed functionalities such as collection of data from the main sources of information, providing this information in different formats (XML, CSV, images) and also, a web application that displays the accounting data in the ATLAS grid topology manner, considering the sites, tiers, clouds, countries and regions that forms the whole ATLAS grid.

This report represents a work in progress, therefore, there are still some additional tasks to be done, such as to improve the interface of the web application (mainly the plots), and add new sources of accounting data, but the latest can be done in an easy way, because of the modularity of the system (MVC architecture). However, the main architectural design of the ATLAS Accounting System is already concluded.

The main advantage of the prototype presented in this paper lies in its *global* capability, allowing a tracking and analysis of the resources's usage for the whole ATLAS grid. In particular, the *managers* of the experiment can use the ATLAS Accounting System to find out how the resources provided for all the sites belonging to the ATLAS grid are being used. They can check whether the resources are being used as it was agreed at the beginning of the Project (Memorandum of Understanding) and can check if the resources are being used efficiently in order to have an efficient work in the whole ATLAS grid.

As it was mentioned before, the ATLAS Accounting System allows data visualization considering the ATLAS topology, but during the development of the ATLAS Accounting System, a serious lack of a single system that provides information related to the resources, services and topology of the ATLAS grid, was detected. In a grid environment, in particular in the ATLAS grid, it is fundamental to have an Information System that gives information of the whole grid, and not only by each sub-grid (which is the current situation). That Information System will be the subject of forthcoming papers.

6 Acknowledgements

Raquel Pezoa deeply appreciates the financial support received from two HELEN fellowships that allowed her to complete two long-term and very pleasant and productive stays (April-October, 2007, and May-July, 2008) at CERN, Geneva, Switzerland, in order to develop researches on grid computing. She also warmly thanks the support of the Universidad Técnica Federico Santa María (UTFSM), Valparaíso, Chile, obtained through UTFSM-DGIP-research grant No. 24.08.62. Last but not least, she would like to express her deep gratitude to Massimo Lamanna for his hospitality and for the opportunity of working in his research team at CERN.

Luis Salinas acknowledges the support from the Center for Technological Innovation on High Performance Computing of the UTFSM and from UTFSM-DGIP-research grant No. 24.08.62.

References

- [1] The ATLAS Computing Model. http://www.gridpp.ac.uk/eb/ComputingModels/atlas_computing_model.pdf
- [2] <http://www.cern.ch>
- [3] <http://public.web.cern.ch/Public/en/LHC/LHC-en.html>
- [4] <http://atlas.ch/>
- [5] <http://lcg.web.cern.ch/LCG/>
- [6] <http://www.opensciencegrid.org/>
- [7] <http://dashboard.cern.ch/>
- [8] <http://goc.grid.sinica.edu.tw/gocwiki/ApelHome>
- [9] <https://twiki.grid.iu.edu/twiki/bin/view/Accounting/WebHome>
- [10] <http://www.sgas.se/>
- [11] LHC Grid accounting package clocks up 1 million job records. CERN Computer Newsletter. <http://cerncourier.com/cws/article/cnl/22401>
- [12] APEL: An implementation of Grid accounting using R-GMA. R. Byrom et al., UK e-Science All Hands Conference, Nottingham, September 2005.
- [13] Nordic Data Grid Facility. <http://www.ndgf.org/ndgfweb/home.html>
- [14] Dashboard for the LHC Experiments. Julia Andreeva. International Conference on Computing in High Energy Physics, CHEP 2007
- [15] R-GMA: Relational Grid Monitoring Architecture. <http://www.r-gma.org/>
- [16] <http://www.ogf.org/>
- [17] ATLAS Accounting System Documentation. <http://dashb-build.cern.ch/build/nightly/doc/guides/apel/html/dev/index.html>
- [18] ATLAS Accounting System <http://dashb-atlas-job.cern.ch/dashboard/request.py/accounting2>
- [19] ATLAS and the WLCG Project. Presentation of Gilbert Poulard in Latin American Software and Computing Workshop. 10 - 13 March 2008, Buenos Aires, Argentina. http://www.df.uba.ar/~aia/atlas_workshop/index.html
- [20] ATLAS Accounting Data. Presentation of Raquel Pezoa in Latin American Software and Computing Workshop. 10 - 13 March 2008, Buenos Aires, Argentina. http://www.df.uba.ar/~aia/atlas_workshop/index.html