Summer Student Work: Accounting on Grid-Computing

Walter Bender

Supervisors: Yves Kemp/Andreas Gellrich/Christoph Wissing

September 18, 2007

Abstract

The task of this work was to develop a graphical tool for accounting the usage of the Grid at DESY.

Contents

1 Introduction to the GRID

In the last years there was a growing need for computing and storage resources in high energy physics. So there was the decision to build up a distributed network for computer-resources called Grid. It is a global network of clusters all over the world and a modern way to use computing resources. The whole physics community can send jobs to the grid network and let them calculate on the sites that support Grid-Computing. In the end the result gets back the to sender. There is also the possibility to save data on the grid. This data St accessible throw the jobs sent to the grid. All resources and the users are organized in Virtual Organizations(VO). On the one hand there are experiments like cms,atlas,hone,zeus etc.. But there is also the possibility to get a special role for a Grid-User (for example to do software-installation or do Monte-Carlo-productions with special privileges).

1.1 How it worked basically

Once the a job is edited at the User Interface, one can add a .jdl[1] - file and send it to the Grid. In dependence of your Virtual Organization your job arrives at the Resource Broker(RB). The RB knows the status of all the sites, that might process your job and send it to the queue of a Computing Element(CE). When a job arrives at the CE, a Batch Server looks at the available Worker Nodes(WN) and runs it on a WN in dependence of the owner of the job[2]. All look of this work is done by the glite software-middleware.

1.2 Personal Research

For me as a summer student it was a great thing to get in contact with the grid and build up my own jobs and let them run on the grid. Apart from this it was possible to get a deeper understanding of Computing Elements, the Batch System and the Worker Nodes.

2 General Accounting

Essential one of the efforts is to monitor the usage of the grid. This information is essential for debugging and management of resources. One part is the accounting of jobs that were executed on the grid-cluster. Each job has entities like owner, workernode, starttime, stoptime and a lot of more. All executed jobs gets accounted in a database. At the local database there is the possibility to account the data per Virtual Organization(VO), per role in the grid or even per user. This information can be used for billing the grid-usage in future times or also be used for controlling the fair-share. (fair-share means the control which user gets what resources in dependence of his VO. Maui is a tool to realize it). Another thing is the representation for international competition. All sites want to achieve high loads and less failures. There are websites that collect the accounting data and offer possibilities to compare different clusters. Accounting is a way to evaluate the performance of the Grid resources.

[1] Job Description Language
[2] fair-share

3 APEL accounting

The DESY Grid infrastructure is operated in the context of the EU-Project EGEE[3], so accounting at DESY is done via APEL[4]. It uses three types of log-files. They are all located on CE. On the one hand there are the general log-files of the Computing Elements located at /var/log/messages. They announce the point in time when a mapping from a Grid-User-Certificate to the local Unix-Account is made. All the detailed informations about the format and the actual Grid-User-Certificate are stored in the Gatekeeper-Logs. At least APEL uses the logs of PBS-Batch-System, which contains all relevant informations about the processed job. If you don't want to loose Information, you have to save the log-files daily. At DESY this is done by an cronjob that saves on a Network-File-System. The APEL Database uses three tables on its database to store all this informations. There is one database for each CE. After processing, APEL combines the tables to a new table one called LcgRecords before the data gets published to the R-GMA-Service[5], which is one central of EGEE.

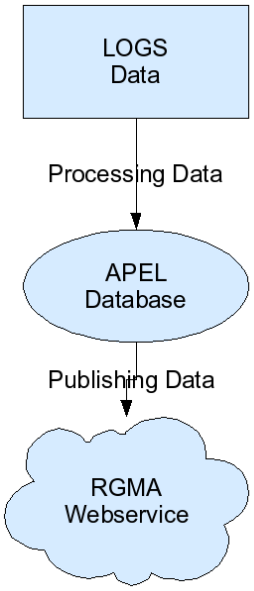

Figure 1: APEL - scheme

3.1 Processing

Processing means the import of the LOG-Files to tables and store them in the database. The relevant tables are EventRecords, GkRecords, MessageRecords.

[3]Enabling Grids for E-SciencE
[4]*http : //goc.grid.sinica.edu.tw/gocwiki/ApelHome*
[5]*http : //www3.egee.cesga.es/gridsite/accounting/CESGA/egee_view.html*

3.2 Publishing

Publishing means to connect the tables to generate LcgRecords and publish the data to the R-GMA-Service

3.3 Extensions for APEL

Besides APEL is a tool that is used on the most sites for accounting, there is a lack of a graphical interface to analyze the data and a lack of data-fields like the time when the job went into the queue or the even the exit-status of an job. A new tool can close the gap to do local accounting in an advanced and easier way !

4 DIMT Accounting

DIMT is an acronym for DESY Interactive Monitoring Tool. DIMT was programmed during the summer student programme. To get in contact with a new programming language *python*[6] has been chosen to develop the software and *root*[7] is used to display the data as charts and to contact the *mysql*[8]-database. To build up an user interface I used *web.py*[9] and *cheetah*[10] to build up an dynamic server site web-application. The main task of the software is to visualize the duration of executed jobs, when it was send to the queue, how long it has been in the queue - all in dependence of the Virtual Organization and/or the specific User-Role. It is also possible to get information which organization/institute or even user used the DESY Grid resource how often and on which level of CPU demand. DIMT searches throw the database interactively and gives back the result as a table, a bitmap (pie, chart) and as a .root-file for further analysis.

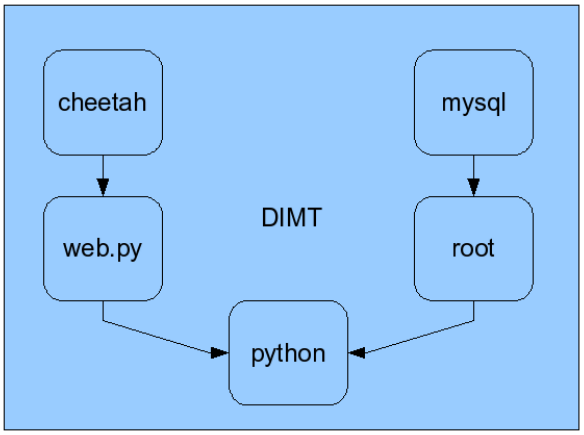

Figure 2: used packets

[6] http://www.python.org/
[7] http://root.cern.ch/
[8] http://www.mysql.com/
[9] http://webpy.org/
[10] http://www.cheetahtemplate.org/

To get the extra functionalities like additional info's about jobs, DIMT needs an extra database that is generated by an additional script. The script searched through the APEL database and uses the PBS-LOGS for additional informations. It will be executed every night with a cron job.

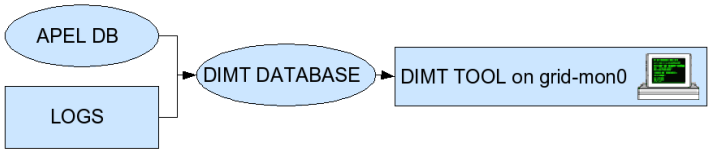

Figure 3: working scheme

4.1 Database

The Accounting database scheme has an additional field for the Queue, ExitStatus or QueueTimeEpoch etc.. The table is needed to to additional accounting and to try out fairsharing.

RecordIdentity	varchar(255)
BatchSystem	varchar(10)
ExecutingSite	varchar(50)
ExecutingCE	varchar(50)
LocalJobID	varchar(50)
LCGUserDN	varchar(100)
LocalGroupID	varchar(50)
LocalUserID	varchar(50)
LCGUserVO	varchar(50)
LocalWN	varchar(50)
Queue	varchar(50)
ExitStatus	int(11)
ElapsedTimeSeconds	int(11)
BaseCpuTimeSeconds	int(11)
QueueTimeEpoch	int(11)
StartTimeEpoch	int(11)
StopTimeEpoch	int(11)
MemoryReal	int(11)
MemoryVirtual	int(11)
SpecIntWN	int(11)
ResourceListCPU	int(11)
ResourceListWALL	int(11)
InsertDate	timestamp
APELEventDate	date
APELEventTime	time
APELMeasurementDate	date
APELMeasurementTime	time

Table 1: database scheme

4.2 Web-interface

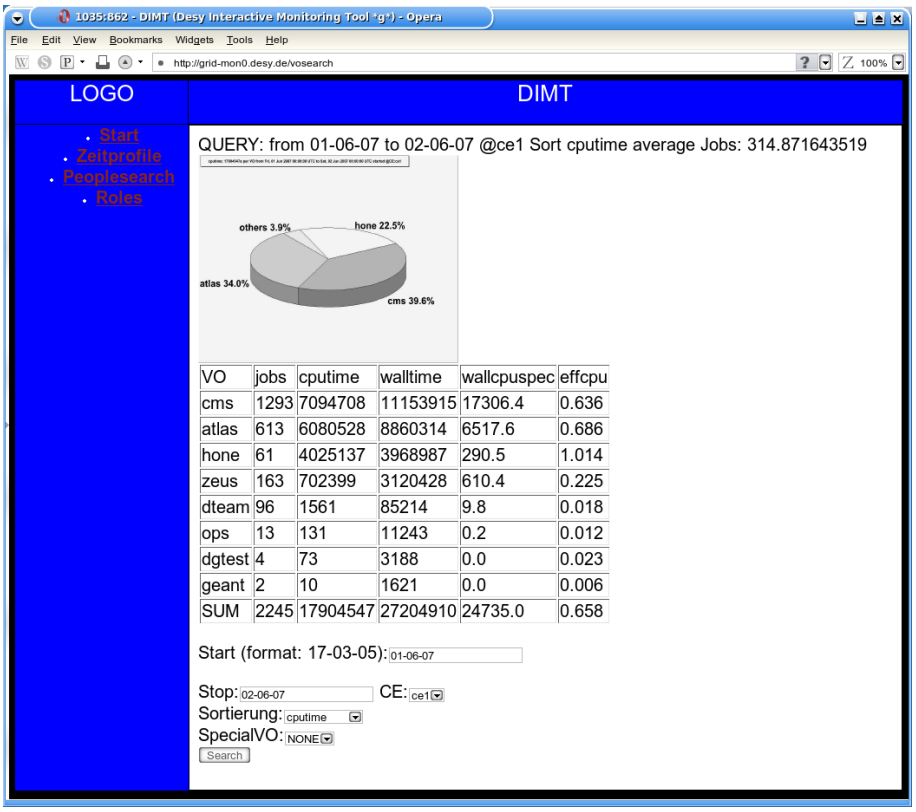

Figure 4: the web application

4.3 Implementation

The code-volume grows by time so that there is no complete code-design, but in general I used a code-design to separate the datamodell of the jobs from the an to display it in charts, pies and tables.

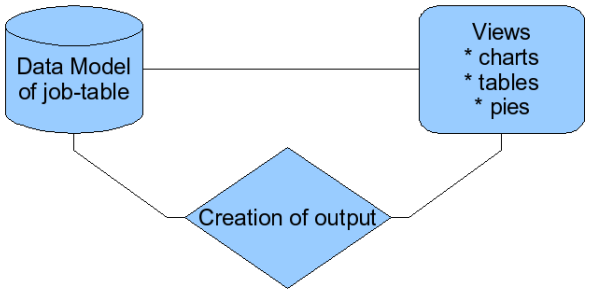

Figure 5: code pattern

4.4 Reports of DIMT

4.5 Analysis per User-Role

DIMT can be used to look at the durations of jobs done at DESY. In fig.6 you can see the duration of jobs that have been executed on all CEs of DESY. It stacks all the different roles to the total number of jobs with the given length. (USR stand for User, SGM for Software manager and PRD for MC-Production). As you can see, the jobs of software managers are only about one minute, while in general the jobs of users and producers are much longer.

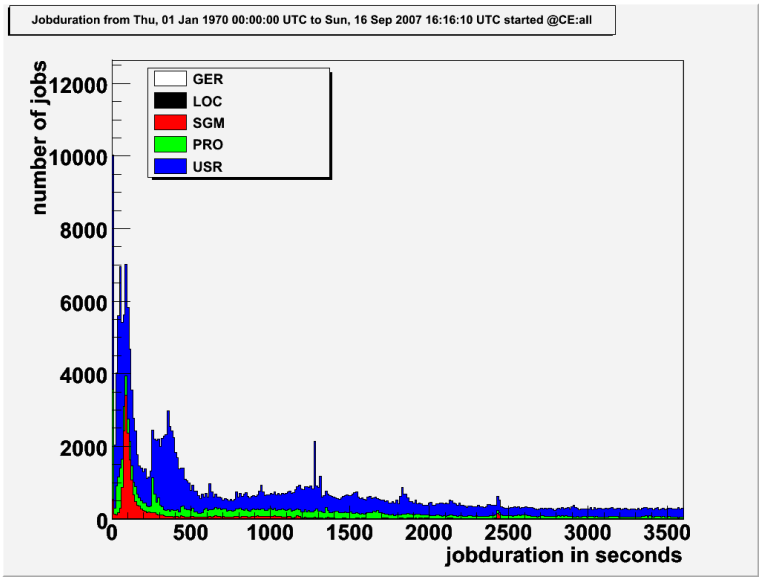

Figure 6: duration of jobs within one hour

If you choose a timescale of three days, look at fig.7, you can see other interesting peaks. The first one is in about one hour. Toolkits from the VOs might kill jobs that are running one day. An other peaks is at two days. Programs that use the CPU all the time has a limit of 2 days until they get killed by the PBS-Batch-System. All the other jobs that do not use the CPU so much and are still running get killed at after three days.

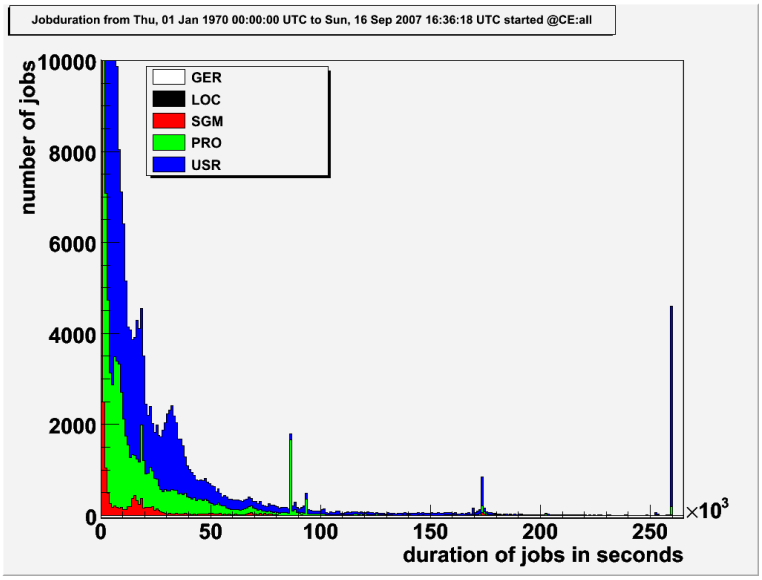

Figure 7: duration of jobs within three days

4.6 Analysis per Virtual Organization

It is also possible to do analysis per Virtual Organization. In fig. 8 you can see the duration of jobs displayed for the main important Experiments/VO. CMS and ZEUS generally have jobs, that are longer than a few minutes, whereas the ops group has only really short jobs with a duration of one minute.

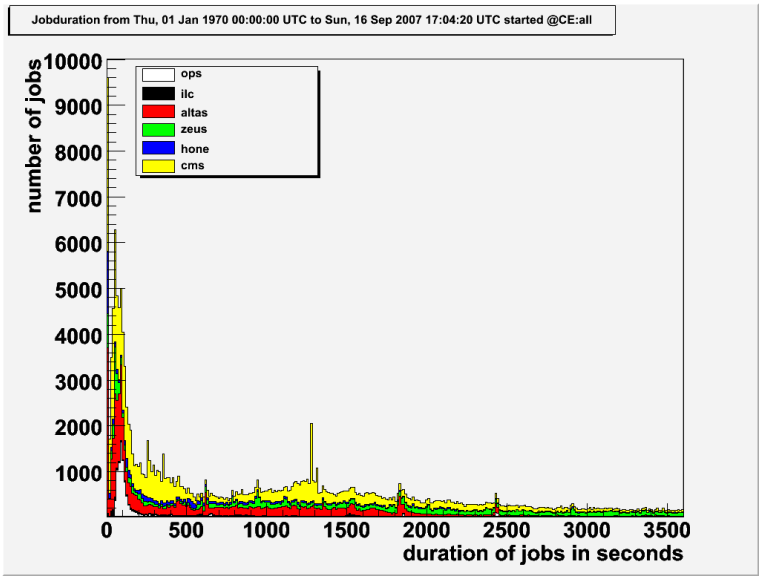

Figure 8: duration of jobs within three days

If you you want to see at what daytime and/or date the most jobs has been

send to Grid, you can print the needed cputime of all jobs in a time interval. Fig. 8 shows us that in the first half of May ATLAS needed a lot of Grid-Resources. HONE then needed a lot of resources in the second half of May after-wards. Analysis can also done at the level of daytime.

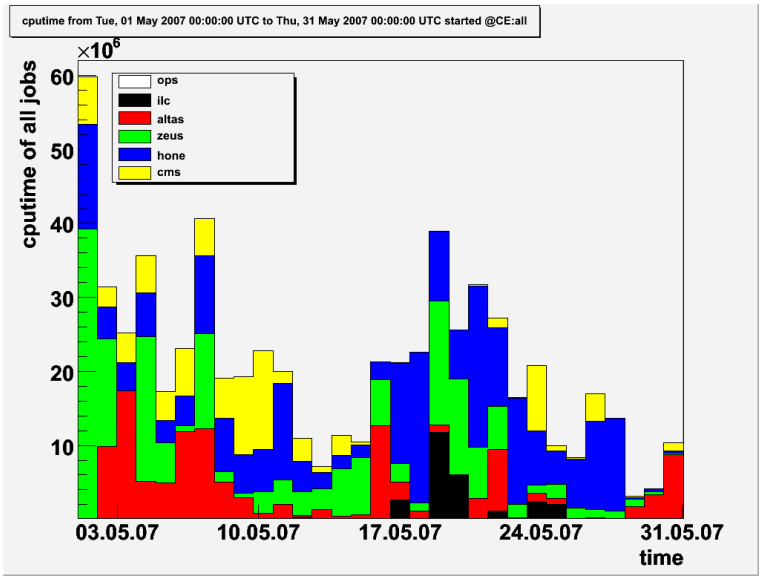

Figure 9: cputime of sent job in may

5 Technicalities

A short reference of the work. All the programs(dimt,mysql,cron) run as a daemon on grid-mon0.desy.de. All software-files for updating and running the web-interface are saved on an file server located under /root/misc.

5.1 dimt

/root/misc/accounting-mon0/dimt/bin/dimt.sh is executed on grid-mon0. A password is needed to access the web-interface.

5.2 mon0-updater

accounting-update-db.py [ce] [DAILY] will be executed daily to update the dimt database

5.3 mysql

MYSQL was installed by yum. there is a database Accounting with the table Jobs specified by tab. 1. There also the possibility to run SQL-Queries at grid-db0.desy.de/phpMyAdmin/ like

```
SELECT ‘LocalUserID‘ , count( ‘LocalUserID‘ )
FROM ‘LcgRecords‘
```

```
WHERE ‘StartTimeEpoch‘ > UNIX_TIMESTAMP( ’2007-07-01 00:00:00’ )
GROUP BY ‘LocalUserID‘
```

on the APEL-Database. This command counts all the jobs with a given LocalUserID and a StartTime after 2007-07-01.