Job-monitoring over the Grid with GridICE infrastructure.

G. Donvito[†], G. Tortone[‡],
T. Coviello[†§], N. De Filippis[†], G. Maggi[†], M. Maggi[†], A. Pierro[†],
[†]Dipartimento Interateneo di Fisica & INFN di Bari - ITALY
[‡]INFN di Napoli - ITALY
[§]Dipartimento di Elettrotecnica ed Elettronica - Politecnico di Bari - ITALY

Abstract

In a wide-area distributed and heterogeneous grid environment, monitoring plays an important and crucial role. It includes system status checking, performance tuning, bottlenecks detecting, troubleshooting, fault notifying.

In particular a good monitoring infrastructure have to provide all the information needed to track the status of each job submitted to the GRID in order to be able to detect new problems as they came up. In this respect a good interaction between the monitoring system and the other Grid services is needed.

The current LCG testbed integrates GridICE as monitoring system. It already measures and publishes the status of each grid resources as a function of time.

In this paper is presented the effort to integrate in the current GridIce infrastructure additional information in order to track the status of the jobs executed, running or queued on the Grid, starting with the job name, the Virtual Organization to which the job belongs, the real or mapped user who submitted the job, the effective CPU time consumed and the job exit status.

INTRODUCTION

In a LCG-GRID[5] environnement it is important both for the user and for the GRID management to have the possibility to check the status of the submitted jobs, the amount of resources used and their locations. Since the information recorded by the logging and bookkeeping services are not enough detailed, a new monitoring tool has been developped which gives the user the possibility to control his jobs with an high level of detail.

The tool is also able of providing aggregate information in a easy customizable way: in particular the user can group the information either by Virtual Organization as well as by farm, or require information only about all of his jobs. In order to guarantee the scalability of the job monitoring system and provide to the end user the access to the complete information relative to all his jobs, the GridICE[2] infrastructure was choosed as back-end for the nedeed operations.

ARCHITECTURE

The job monitoring is based on the GridICE architecture. A breif description of the data flow is reported below:

- A *sensor* in each computing element reads the information from the local batch system
- The information are sent via TCP/IP to the GridICE *collector*
- The GridICE *collector* catches the information and publishes it trought an LDAP server such as for the other information
- the information is then read out by the central server and stored in the GridICE database
- the information can be requested by the user by means of few web pages.

Retrieving information

At the lowest layer of the architecture a *sensor* retrieves the information about all jobs running on a given Computing Element. This component is actually implemented using a perl script (called *CheckJobs.pl*).

The script is executed periodically and requires some input parameters like.

- the observation time (in hours). Jobs ended in the observation time are published.
- a flag to enable or disable the monitoring of the local jobs. (Local job are the jobs not submitted by the GRID)
- the directory where the logs of the batch system are stored
- a flag to enable the monitoring of the queued jobs
- the batch system installed

Currently the following batch systems are supported:

- OpenPBS[6]
- LSF[7]
- Torque + Maui[8]

The following files are used to extract the requested informations:

- the logs of Operating System (*/var/log/messages*)
- the logs of Globus Gatekeeper daemon (*/var/log/globus-gatekeeper.log*)

- the accounting logs of the batch system (in case of OpenPBS or Torque)

The script then retrieves the Grid job identifier for each running or queued job on the farm, getting all the data needed to give a precise snapshot of what happens on the farm.

The following information is retrieved for each job:

- the user Virtual Organization
- Local User account used on the system to run the job
- Real User who submitted the job (and the Subject of his personal digital certificate)
- CPU time used by the job
- Machine Time used
- Real memory used
- Virtual memory used
- Local batch system job identifier number
- Grid job identificative (the same that the user uses for querying the Resource Broker about the job)
- Queue time (the time that the job spent in queue)
- The job exit state code (from the batch system logs)

The first two parameters are estracted from the logs (or from commands for LSF) of the local batch system (assuming that the VO of the user is the Unix *group* of the local user). The real user (that submitted the job) is derived from the *globus-gatekeeper.log* matching the local user with the subject of certificate of the real user. The information about consumed resources is taken from the batch system, using the logs file (or command for LSF) for jobs that are executed and a command line for jobs wich have not been already executed. Grid Job identifier is retrieved from a subdirectory of the home of the user and it is available during the lifetime of the jobs. The queue time is got from the local batch system information considering the creation time and the starting time of each job.

In order to implement the monitoring system on a different batch system, it is required to adapt some of the routines to retrieve the information needed. A deep knowledge of the specific batch system is required. It is planned to allow the use of a different plug-in according to the installed batch system to semplifying this task.

Transferring the information

All information about jobs are transferred from the CE (using the *edg-fmon-agent*) to the GridICE *collector* (that usually is located on the Storage Element of the farm) in which the *edg-fmon-server* daemon receives and caches all information. This cached information is then published from the *gridice-mds* daemon via LDAP. For these reasons, is necessary to *translate* the output of the *CheckJobs.pl* into a ordered LDAP[4] tree to publish the information about jobs monitoring. This operation is done by a *fmon2glue* program running on the GridICE *collector*.

This layer will be replaced by a SOAP[1] protocol for trasporting the information in future.

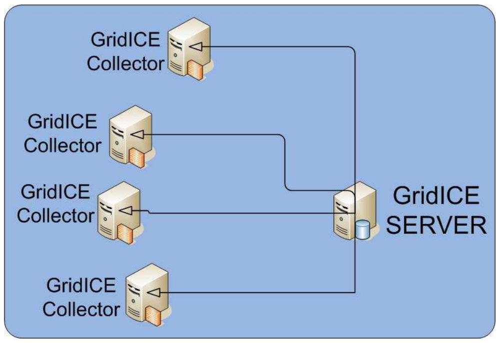

Figure 1: The GridICE server query all LDAP tree of the resource to collect all the information published

Collecting information

As shown in figure 1 the information published by all the GridICE *collectors* are collected by a central GridICE server that reads that information from the *gridice-mds* servers with a *round-robin* schedule and with a fixed frequency.

The GridICE server can be configured to watch only the resources that are interesting so avoiding to overload the server with information not used by users.

In this way the scalability to the system is ensured; in fact it is possible to change the frequency of operation of reading the information according to the load of all the system. There is also the possibility to access to various metrics using different reading frequencies.

Information read by various GridICE *collectors* are organized in the GridICE database. The RDBMS used is PostgreSQL[3] which implements advanced features such as:

- triggers
- view
- Stored procedures
- functions

In Figure 2 the tables involved in the Job Monitoring and the relations between them are shown.

With this database organization it is possible to retrieve, all the information about the job, for example through the GridJobID; it is also possible to trace the history of the submitted job for a given user.

Using the data

The data are stored persistently in this PostgreSQL database from where the history of all jobs can be retrieved. The data are made available to the end-user on the web. It is possible to browse the job information by VO, farm , user. etc.

The information can be filtered by specifying a job submission date, a time interval with the starting and ending time, or the job exit status. The information can be used to extract on line reports of the resource used by a given VO on a given farm or by a given user.

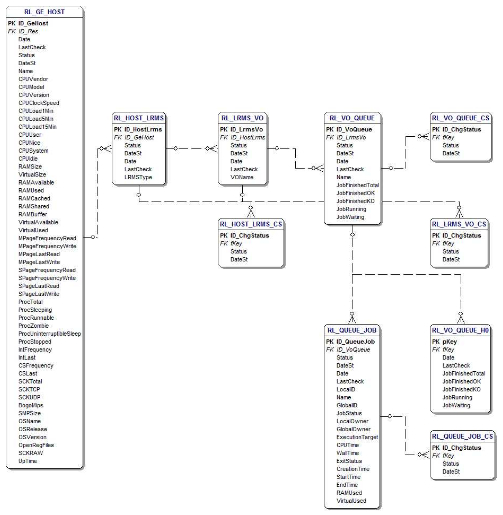

Figure 2: Tables of Job Monitoring and relationship between them.

It is also possible to correlate the job monitoring information with that provided by the resource monitoring in order to correlate the CPU, memory and network used on a particolar WN to the job running on that node.

In order to speed up the creation of the reports by a common web form, some work is in progress to develop some Stored Procedure and functions.

CONCLUSIONS

The Job Monitoring descibed in this paper provides the user with a powerful tool to check the excution of his jobs on the GRID. The system is also useful for a farm, a VO or a GRID administrator to control wath is happening on the farm, in a VO and on the entire GRID infrastructure.

ACKNOWLEDGEMENTS

We would like to thank all the GridICE team and all the site administrator that helped us during the tool implementation.

REFERENCES

[1] www.w3.org/TR/soap/

[2] http://infnforge.cnaf.infn.it/gridice/

[3] www.postgresql.org/

[4] www.openldap.org

[5] LHC Grid Computing Project http://lcg.web.cern.ch/LCG/

[6] www.openpbs.org

[7] http://www.platform.com

[8] http://www.supercluster.org