WMSMON – gLite WMS Monitoring Tool

D. Vudragović, V. Slavnić, A. Balaž and A. Belić
Scientific Computing Laboratory, Institute of Physics Belgrade
Pregrevica 118, 11080 Belgrade, Serbia
Phone: +381 11 371 3152, Fax: +381 11 316 2190
E-mail: dusan@scl.rs, Web: http://www.scl.rs/

Abstract - The complex task of computing resources discovery and management on behalf of user applications in the gLite Grid environment is done by the Workload Management System (WMS) service. However, the current implementation of Grid Service Availability Monitoring framework does not include direct probes of this essential service. In this paper we describe the newly developed WMSMON tool, which provides a site independent, centralized, uniform monitoring of gLite WMS services. This tool is based on the collector-agent architecture, and offers aggregated status view of all monitored WMS services, as well as a detailed status page for each service, with links to the appropriate troubleshooting guides when problems are identified. The WMSMON tool is currently deployed by the SEE-GRID-SCI Grid e-Infrastructure.

I. INTRODUCTION

Processing enormous amounts of data generated by science experiments requires huge computational and storage resources, and associated human resources for operation and support. In order to enable seamless use of available distributed resources and automatize as much tasks as possible, a new layer is built on top of the existing network infrastructure – Grid layer of software (Grid middleware) able to interconnect distributed computing and storage resources, and make them interoperate, providing users with the unified access to all resources.

There are many kinds of Grids with different purposes, such as national Grid infrastructure (aiming to couple high-end resources across a country, such as AEGIS [1], D-Grid [2], or the UK e-Science program [3]), Grid projects (funded by various international funding agencies, e.g. EC-funded SEE-GRID-SCI [4], EGEE [5], or NorduGrid [6] jointly funded by Nordic countries), Grid infrastructures provided by joint efforts of many individuals aiming to help in solving important common problems (e.g. in finding drugs for diseases, SETI@HOME [7], etc.), consumer Grids established by commercial companies, etc.

Usually all Grid resources centers (sites) in one Grid infrastructure or project use the same type of Grid middleware, such as gLite [8] (used in EGEE, SEE-GRID-SCI and many other projects), Globus Toolkit [9], Virtual Data Toolkit [10] (used by the OSG project [11]), ARC [12] (used by the NorduGrid collaboration), Garuda [13] (Indian Grid middleware), etc.

The Workload Management System (WMS) is one of the key Grid services composing the gLite Grid middleware stack. It provides a service responsible for the distribution and management of tasks across resources available on the Grid, i.e. it performs a complex task of distributed scheduling and management of users' jobs on a dynamically determined set of available computing resources.

The gLite WMS service has been in continuous development since before the start of European DataGrid project [14], i.e. for about 8 years. The name used by the developers was always Workload Management System, but name Resource Broker (RB) was most commonly used in the past by the users.

The Large Hadron Collider (LHC) [15] collaboration has setup LHC Computing Grid (LCG) [16] in 2003, which used the then-current release version of RB for the LCG-1 production service, but the measured performance of WMS was far from production quality. A lot of effort was devoted to make this key service more robust. However, its functionality was not extended nor improved, and RB service has had minimal maintenance for several years now.

Meanwhile, the developers of Grid middleware organized around the EGEE project kept developing WMS service and they produced a major new version in 2004, which was incorporated into the glite 1.x release series. However, this was never deployed in production. In 2006 the glite and LCG releases were forcibly merged to glite 3.0 and finally the new WMS was put into production. The performance of the new service was unstable and it was largely unused, keeping RB service still in production.

For this reason, the development of WMS continued until major upgrade in the glite 3.1 release series. With the advent of EGEE-II project, the acceptance criteria for certification became realistic. Now it has finally been certified as production-quality, hence the old RB SERVICE was finally retired, and the new gLite-WMS is used as a main production-quality service.

II. WMS ARCHITECTURE

The purpose of Workload Management System (WMS) is to accept job submission and management requests from its clients (configured on User Interface machines) and to take appropriate actions based on these requests.

Through the User Interface (UI) [17], a user can find the list of resources suitable to run a specific job, submit a job for execution on a remote Computing Element (CE), check the status of a submitted job, cancel one or more submitted jobs, retrieve the output files of a completed job (output sandbox), or retrieve and display logging and bookkeeping information about submitted jobs.

Once submitted from UI machine, the job request passes through several other components of the WMS before it completes its execution. The WMS components [18] handling the job are: Workload Manager Proxy, Workload Manager, Job Controller, Log Monitor, CondorC, Proxy Renewal Service, Logging and Bookkeeping, and Log Monitor. The simplified architecture of WMS and component interactions are presented in Fig. 1

The Workload Manager Proxy (WMProxy) [19] provides support for the job control functionality through a Web Services based interface. It provides a core module performing validation, conversion, environment preparation and information logging for each incoming request, before delivering it to the Workload Manager.

The Workload Manager (WM) is the core component of the WMS. Given a valid request, it takes the appropriate action to satisfy it. To do so, it uses support from one or more of the following components:

1) Matchmaker: Helper component offering support to the WM in making the decision on which of available resources should be used to fulfill a particular job submission request.

2) Information Super Market: Cache of all the information on available computing and storage resources necessary for the matchmaking process. It is dynamically updated by the Information Updater through a mixture of polling resources information and receiving notifications.

3) Task Queue (TQ): Gives the possibility to keep a submission request for a while if no resources matching the job requirements are immediately available.

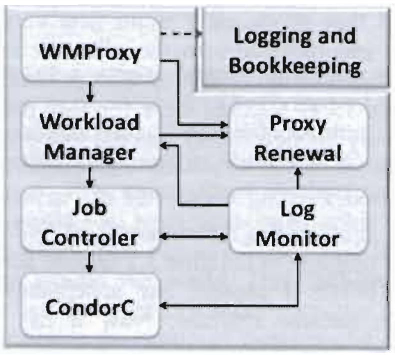

Fig. 1. Simplified gLite Workload Management System Architecture.

In addition to preparing the internal (CondorC) submission file for each job, Job Controller (JC) module is also responsible for creating the job wrapper script to ensure the appropriate execution environment is setup on the Worker Node (WN) when job starts.

The Log Monitor (LM) component is responsible for monitoring the CondorC log file, intercepting interesting events concerning active jobs (events affecting the job state machine: job done, job canceled, etc.) and triggering appropriate actions.

CondorC is the module responsible for performing the actual job management operations (job submission, job removal, etc.), issued on request of the Workload Manager.

Proxy Renewal Service is available to assure that during the lifetime of a job, a valid user proxy digital certificate exists within the WMS. The proxy renewal service relies on the MyProxy service for renewing credentials associated to the request, assuming that the user has stored longer term credential on the MyProxy server.

The Logging and Bookkeeping (LB) service [20] provides support for the job monitoring functionality: it stores logging and bookkeeping information concerning all events generated by the various components of the WMS, as well as other services handling the job (UI, CE, WN). Using this information, the LB service keeps a state machine view of each job. LB service can be deployed separately from WMS, but usually it is not the case, i.e. two services are collocated on the same machine. The WMSMON tool described in this paper assumes that LB and WMS are installed and configured on the same server.

III. WMS PROPERTIES RELEVANT FOR MONITORING

The study of the WMS architecture and design leads to the identification of key properties that can be used to monitor the health and availability of gLite WMS service. These properties can be roughly classified as load averages, job queues properties, file system properties, log file properties, and availability/responsiveness of gLite services/daemons:

1) Load averages: If the average value of the load of WMS host machine (1-minute, 5-minute, or 15-minute average) becomes too high, it will slow down the management of existing jobs and job submission from UIs. For this reason WMProxy software contains a load script used for any of the supported job operations. By default, WMS configuration load script is used during the WMProxy operation of job registration and submission. The script as an input parameter has the threshold for load average, and if the server load is too high, the requested operation will be refused. In order to avoid this scenario, the trend in CPU utilization should be tracked, and appropriate actions taken by the administrator.

2) Job queues properties: If the number of jobs in some of WMS queues becomes very high, it can affect or even block the proper work of WMS service and cause loss of jobs. There are three queues used by the WMS service:

- The first queue is used for a communication between WMProxy and WM. This communication is realized through a thread-safe, file-system based queue. Requests waiting to be served (i.e., submit, resubmit, match, cancel) by WM are put in this queue. The jobs that require resources not available will finish in this queue, which sometimes can slow down submission of new jobs from UI or increase the load of machine and cause refusing submission of new jobs.
- The second (also thread-safe file-system based) queue is used for JC and CondorC communication, for storing the requests waiting to be served (i.e., submit, cancel) by CondorC.
- When a job has been processed by WM and its helper modules, and the appropriate CE has been found, the job has to be transferred to the CE via CondorC. CondorC is the module responsible for performing the actual job management operations (job submission, job removal, etc.), issued by request of WM. The

information regarding all jobs submitted to different CEs while the job data is transferred can be retrieved from this CondorC queue. It is important to monitor the number of jobs in the CondorC queue, since the significant increase in the number of jobs can be used to identify problems in the operation of this WMS service.

3) File system properties: Since thread-safe, file-system based queues are used for WMProxy-WM and JC-CondorC communication, the size of these files can indicate possible problems in the operation of key WMS services.

Due to jobs that produce large output sandbox files, monitoring of Sandbox partition is necessary. Also, it is important to track the amount of data in middleware log files partition, MySQL partition and temporary partition.

Each file in the Linux file system has the associated inode number, which provides important information on the file, such as user and group ownership, access mode (read, write, execute permissions) and the type. The number of inodes is limited for each filesystem at the moment of its creation, limiting also the maximum number of files that can be created by the file system. It can happen that users submit large number of jobs with huge numbers of small or empty files, so that the maximal number of inodes is reached, and operation of file system blocked. This can also happen due to malfunction of some of WMS services. For this reason it is also very relevant to track the number of used/available filesystem inodes.

4) Log files properties: WMS service is frequently upgraded and new versions of software are released, which can cause mistakes in the implementation of logrotate scripts. Since the log files are used by different applications for data analysis and statistic information, the large log file sizes can cause significant slow down of such services, or even block their normal operation. Also, this can cause overload of the filesystem.

5) Availability/responsiveness of gLite services/daemons: Since each of WMS services is implemented as a Linux daemon, due to stability issues it is still necessary to check periodically if each of them is active and responsive. The return status of each service should be equal to zero. Normally, the services with a non-zero value are restarted automatically, thanks to the WMS cron job that checks statuses of all gLite services. For some daemons the automatic restart fails after a crash, as it tries to start the service without first stopping it (to remove locks etc.). For this reason it is important to track status of all WMS services and to take appropriate actions if some of them are not properly running.

IV. WMSMON ARCHITECTURE AND IMPLEMENTATION

WMSMON tool is based on the collector-agent architecture that ensures monitoring of all properties relevant for successful operation of gLite WMS service and triggering of the alarms if certain monitored parameter values exceed predefined limits. In addition, the tool provides links to the appropriate troubleshooting guides when problems are identified. The architecture of the WMSMON tool is shown in Fig. 2.

WMSMON tool consists of two parts of software. The first one, WMSMON Agent, should be installed on all monitored WMS services, and locally aggregates the values of all relevant parameters described in the previous section. The second component of WMSMON software is WMSMON Collector, installed on a specific machine equipped with the web server and gridFTP client, with the aim to collect the data from all WMSMON Agents and to provide web interface to the graphical presentation of the collected data.

The WMSMON Agent is composed of data parser and data publisher. The data parser is a bash script implemented as a cron job that searches for predefined WMS properties and parses their values. Since the data are extracted from different sources, the script is using separate methods for parsing and building of a vector of relevant properties values.

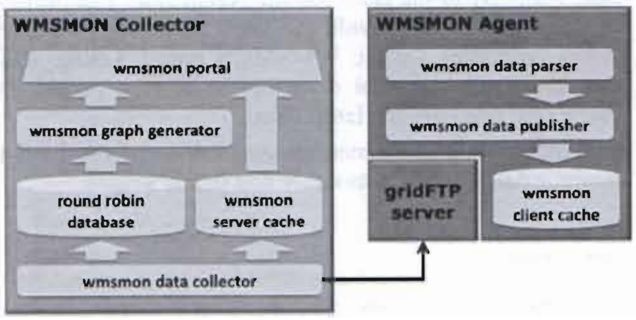

Fig. 2. WMSMON tool Architecture.

When the data vector is built, the data publisher transfers it to the data cache. The main role of the data cache is to store and keep the data for the transfer to the WMS Collector. In the case of a broken network connection between WMSMON Agent and WMSMON Collector, the data cache ensures that there will be no loss of information.

The data cache keeps the values of monitored properties until they are transferred to the WMSMON Collector. In order to provide high-performance, secure, reliable data transfer, we use the gridFTP service [21], already deployed by the gLite WMS middleware. Each WMSMON Agent has only one configuration file containing the distinguished name (DN) of the WMSMON Collector host digital certificate that will be authorized to connect via gridFTP server and mapped to a local user in order to retrieve the information from the data cache.

The WMSMON Agent is implemented as a Linux daemon. During its startup it configures gridFTP server on a monitored WMS to allow the connection by a specific WMSMON Collector (one or more), creates appropriate data parser cron job, and initializes the data cache. The Agent is released as an RPM package and the latest version is available from the SCL RPMs repository [22].

WMSMON Collector consists of the following components: data collector, collector cache, database, graph generator, and the frontend of the tool, released as a web portal. The role of data collector is to retrieve the data from each of monitored WMS services, and to publish it to the database and local cache. Using the data from the database, the graph generator component produces graphs, which are, together with the information from the data cache, displayed through the WMSMON web portal.

The data collector is a bash script implemented as a cron job. It periodically gathers and assembles the information from all monitored WMSMON Agents via gridFTP service, and publishes these data to the local database and cache.

WMSMON tool uses as a backend database the Round Robin Database (RRD) [23]. In this way the data are stored in an easy and lightweight database that does not require another running service, and has storage requirements that can be easily limited by the design of the database. RRD was written as a system to store and display time-series data in a compact manner, with the size that will not expand over time, and in the form suitable for producing time graphs.

The data cache is introduced on WMSMON Collector side, similarly to the cache on the Agent side. The purpose of this cache is to provide the latest values of monitored WMS properties for the WMSMON portal. Unlike RRD that keeps all historical data for the chosen period, the cache contains only the latest data.

WMSMON graph generator uses the data from RRD to produce daily, weekly, monthly and yearly graphs.

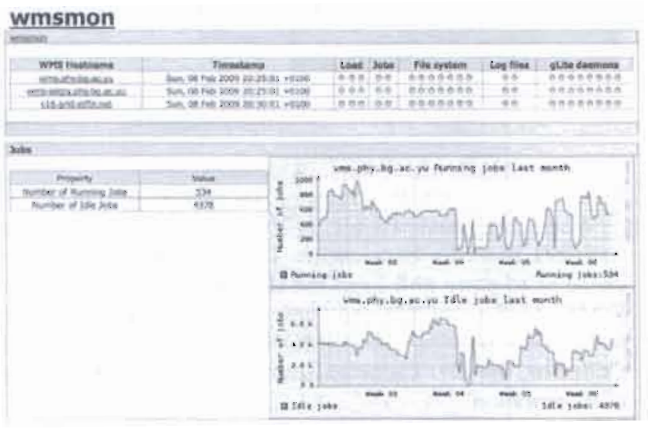
wmsmon
wms.phy.bg.ac.yu Running jobs last month
wms.phy.bg.ac.yu Idle jobs last month

Fig. 3. Overview of WMSMON web portal.

WMSMON web portal [24] presents information from diverse WMS sources in a unified way, as can be seen on Fig. 3. The main page provides the aggregated status view of all monitored WMS services from the target Grid infrastructure. This part of the portal presents the data in a simplified way, with the emphasis on WMS services identified not to work properly.

The portal also provides links to pages with detailed information and graphs for each monitored WMS service. These pages contain the latest data, as well as historical data presented in the graphical form, as shown on Fig. 3.

The WMSMON Collector is implemented as a Linux daemon. During its startup it creates appropriate cron jobs, initializes the RRD and local cache, and provides Apache server with the path information to WMSMON PHP scripts. The WMSMON web portal is realized as a set of PHP scripts. The WMSMON Collector is released as an RPM package and the latest version is available from the SCL RPMs repository.

V. CONCLUSIONS

We have presented the WMSMON tool, deigned and implemented to monitor the properties relevant for operation of gLite WMS services. This service is an essential element in Grid infrastructures, providing distributed scheduling of user jobs submission and management requests. The WMSMON tool is implemented in collector-agent architecture and enables identification of operational problems with monitored WMS services, as well as assessment of hardware/software bottlenecks. The WMSMON tool has been deployed in the SEE-GRID-SCI infrastructure.

ACKNOWLEDGMENTS

This work is supported in part by the Ministry of Science and Technological Development of the Republic of Serbia through research grant No. OI141035, and by the European Commission through projects CX-CMCS (FP6), SEE-GRID-SCI (FP7) and EGEE-III (FP7).

REFERENCES

[1] Academic and Educational Grid Initiative of Serbia (AEGIS), http://www.aegis.rs/
[2] German Grid Initiative (D-Grid), http://www.d-grid.de/
[3] UK e-Science Programme, http://www.rcuk.ac.uk/escience/
[4] SEE-GRID eInfrastructure for regional eScience (SEE-GRID-SCI), http://www.see-grid-sci.eu/
[5] Enabling Grids for E-sciencE (EGEE), http://www.eu-egee.org/
[6] NorduGrid, http://www.nordugrid.org/
[7] SETI@HOME, http://setiathome.ssl.berkeley.edu/
[8] gLite Middleware, http://glite.org/
[9] Globus Toolkit, http://www.globus.org/toolkit/
[10] Virtual Data Toolkit, http://vdt.cs.wisc.edu/
[11] Open Science Grid (OSG), http://www.opensciencegrid.org/
[12] Advanced Resource Connector (ARC), http://www.nordugrid.org/middleware/
[13] India's National Grid Computing Initiative (Garuda), http://www.garudaindia.in/
[14] European DataGrid project, http://eu-datagrid.web.cern.ch/
[15] Large Hadron Collider (LHC), http://lhc.web.cern.ch/
[16] LHC Computing Grid (LCG), http://lcg.web.cern.ch/
[17] https://edms.cern.ch/file/722398/gLite-3-UserGuide.pdf
[18] Workload Management System User and Reference Guide, https://edms.cern.ch/file/572489/1/WMS-guide.pdf
[19] WMProxy User Guide, https://edms.cern.ch/file/674643/1/WMPROXY-guide.pdf
[20] Logging and Bookkeeping User and Reference Guide, https://edms.cern.ch/file/571273/2/LB-guide.pdf
[21] GridFTP data transfer protocol, http://www.globus.org/grid_software/data/gridftp.php

[22] Scientific Computing Laboratory of the Institute of Physics Belgrade, RPM repository, http://rpm.scl.rs/

[23] RRDtool, http://oss.oetiker.ch/rrdtool/

[24] SEE-GRID WMSMON web portal, http://wmsmon.scl.rs/