Grid Site Monitoring tools developed and used at SCL

V. Slavnić, B. Acković, D. Vudragović, A. Balaž, A. Belić
Scientific Computing Laboratory
Institute of Physics Belgrade
Pregrevica 118, 11080 Belgrade, Serbia
http://www.scl.rs/

M. Savić

Faculty of Electrical Engineering
University of Banja Luka
Patre 5, 78000 Banja Luka, Republika Srpska
Bosnia and Herzegovina

Abstract — **In today's deployment of Grid services, continuous monitoring of computing and storage resources and core services is crucial for achieving high level of service reliability. A number of system parameters have to be under constant observation and control: from hardware conditions (temperature of the motherboard and of the CPU, fan speed, disk state) through state of operating system, to Grid-related events and daemons. In the Scientific Computing Laboratory of the Institute of Physics Belgrade we are using several such monitoring tools, with a number of them partially or fully developed and publicly available. Here we present the most used Grid monitoring tools at SCL.**

Keywords- Grid; SCL; SAM; Linux; Ganglia; Pakiti; CGMT; WatG; WMSMON;

I. INTRODUCTION

Grid site is a complex system, which consists of number of different subsystems. A basic hardware layer is only a foundation for other layers stacked above. The operating system (OS) layer provides the interface between computer hardware and installed software. On top of it, in the Grid computing environment there is an additional layer of middleware software stack that provides Grid services to end users, as well as integration of all Grid services and interoperation of Grid sites comprising a given Grid e-Infrastructure. However, working environment of a Grid site is even more complex: it relies also on the network and cooling subsystems, which are essential for the successful operation of a resource center. Each of these subsystems has a set of attributes that define its state. For the hardware layer there are parameters such as temperature, voltage, fan speed and many others. For operating system examples of such attributes are CPU load, disk and memory usage, while for middleware layer relevant attributes are e.g. number of jobs (running and queued), number of available CPU and storage resources, test results etc.

To achieve and maintain high availability and reliability of computing and storage resources, Grid site administrators have to monitor and supervise each of the important attributes measuring the health of the system and quality of the offered services to end users.

In the Scientific Computing Laboratory (SCL) [1] of the Institute of Physics Belgrade we use several monitoring tools. Some of these tools are developed by SCL for its specific needs, and we have made them publicly available to all other interested parties from our SVN and RPM repository. In this paper we describe such tools, as well as Grid site monitoring tools developed elsewhere, but deployed at SCL and regarded as highly useful not only in everyday operations but also from user's point of view.

II. GRID SITE MONITORING TOOLS

We will first describe Cumulative Grid Monitoring Tool (CGMT) [2] in Section *A*, used to present the status of all monitored services in an integrated way. Section *B* presents WMSMON [3] tool, which monitors the status of WMS core services deployed at SCL, while in Section *C* we present WatG browser [4], a web-based Grid Information System (GIS) visualization application. We also present security-related tool Pakiti [5] which is slightly modified and deployed at SCL in Section *D*. Highly-customizable general-purpose monitoring tool Ganglia [6] is described in Section *E*. Purely Grid-specific testing framework SAM [7] together with the BBmSAM [8] is presented in Section *F*. Section *G* describes another Grid-specific information system monitoring tool, GStat [9].

A. CGMT

Cumulative Grid Monitoring Script (CGMT) is set of scripts accompanied by the simple web interfaces developed for Grid site monitoring and integrated presentation of the results provided by various monitoring tools. Some of these scripts can be deployed on any general-purpose computing cluster, without the involvement of gLite middleware [10]. CGMT gives fast overview of the state of a Grid site and easy access to more detailed monitoring tools. It combines other tools developed in SCL and tools developed by other developers, but deployed locally, such as Ganglia, GStat, SAM, etc. Block diagram of the structure of CGMT tool is given in Figure 1.

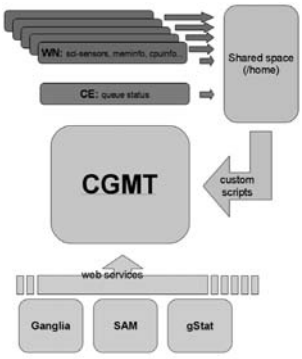

Figure 1: CGMT structure block diagram.

First source of information for CGMT are local nodes and the other source are web based monitoring tools stated above. CGMT gets the data directly from nodes about environment conditions, such as CPU and motherboard temperature, CPU load, memory and swap information, and hard disk usage. This is done through the scl-sensors script [11], which should be installed on all nodes. This script is getting some of the data from IPMI [12] card installed on the motherboard, as well as from OS-implemented sensors. The collected data are stored on the shared home directory. Another source of important information is the Computing Element job management system, which provides the data about number of scheduled and running jobs per Virtual Organization, using the scl-jobs [13] script.

CGMT collects further data about a given Grid site from several different web services, such as Ganglia, SAM and GStat, as shown in Figure 2.

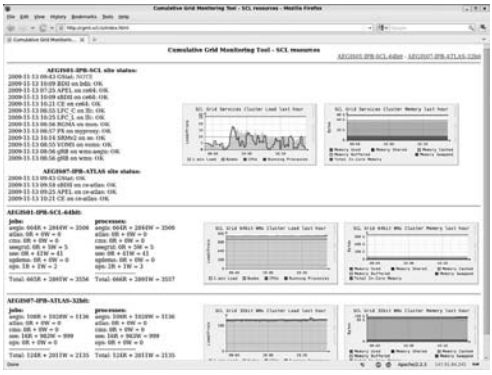

Figure 2: Main web page of CGMT tool.

CGMT can be configured to monitor multiple Grid sites or multiple clusters. Each monitored site/cluster defined in CGMT has its own page (Figure 3), with information about relevant SAM tests, GStat status, number of running and waiting jobs per VO, which can be customized according to the local needs. On the right side of the individual cluster page, a detailed info about node names, CPU loads, RAM and swap usage, and CPU temperatures is shown in a table, as can be seen in Figure 3.

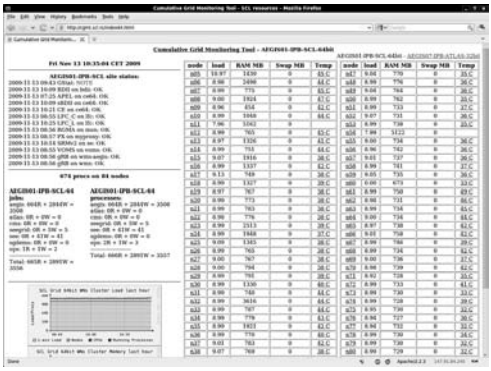

Figure 3: Cluster page of CGMT.

More details about each node can be found on individual node pages (Figure 4). These pages display information collected from IPMI sensors such as: measured temperatures, voltages, fan speeds, as well as OS information such as CPU load, detailed memory info, hard disk status and others. The scl-sensors scripts running on all the monitored nodes provide data presented on individual node pages. In addition to this, for each monitored node a separate page with node temperatures graphs is generated using the MRTG tool, as can be seen in Figure 5. For each monitored property a threshold can be set, and if the threshold value is exceeded, this information is shown in a visually different way, so that identified warnings or errors can be easily detected by site administrators.

Figure 4: Node info page.

Figure 5: Node temperature page.

B. WMSMON

WMSMON is a tool developed by SCL which provides a site independent, centralized, uniform monitoring of gLite WMS services.

The Workload Management System (WMS) is one of the key Grid services of the gLite middleware software stack. It performs a complex task of distributed scheduling and management of users' jobs on dynamically determined set of available resources. Properties of WMS that can be used to monitor the health and availability of this service can be classified as load averages, job queues properties, properties of the file system, log file properties, and availability /responsiveness of gLite services/daemons on WMS.

- *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.
- *File system properties*: Since thread-safe, file-system based queues are used between WMS modules, the size of these files can indicate possible problems in the operation of key WMS services.
- *Log file properties*: Log files are used by different applications for data analysis and statistic information and large log file sizes can cause significant slowdown of such services, or even block their normal operation. Also, this can cause overload of the file system.
- *Availability/responsiveness of gLite services/daemons*: Since each of WMS services is implemented as a Linux daemon, due to stability issues it is necessary to check periodically if each of them is active and responsive and take appropriate actions if some of them are not properly running.

The WMSMON tool monitors all properties stated above. It is based on collector agent architecture, and offers aggregated status view of all monitored WMS services, as well as detailed status page for each service, with links to appropriate troubleshooting guides when problems are identified. This tool triggers the alarms when certain monitored parameter values exceed predefined limits.

The architecture of WMSMON tool is shown in Figure 6.

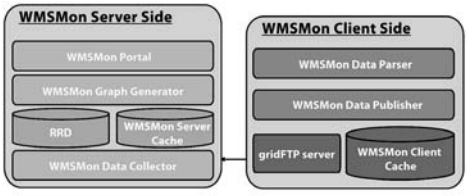

Figure 6: Architecture overview of WMSMON.

WMSMON consists of two parts of software. First one is called WMSMON Agent and it should be installed on all monitored WMS services. It locally aggregates the values of all monitored parameters. The other component of WMSMON is WMSMON Collector and it is installed on specific machine equipped with the web server and gridFTP [14] client, with purpose to collect the data from all WMSMON agents and to provide web interface to the graphical presentation of the collected data.

WMSMON Agent is implemented as a Linux daemon and it 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. All gathered data is transferred to data cache which main role is to store and keep the data for the transfer to the WMS Collector assuring that will be no loss of information in case of broken network between WMSMON Agent and WMSMON Collector.

For transfer of data gridFTP service, already deployed by the gLite middleware is used. Each WMSMON Agent has only one configuration file containing distinguished name (DN) of the WMSMON Collector host digital certificate that will be authorized for connection through gridFTP server and mapped to a local user in order to retrieve the information from the data cache. During its startup Agent configures gridFTP server on a monitored WMS to allow the connection by a specific WMSMON Collector (one or more), initializes data cache and creates appropriate data parser cron job.

WMSMON Collector consists of the following components: data collector, collector cache, database, graph generator, and the frontend of the tool implemented as a web portal. The role of data collector is the retrieve data from each monitored WMS services, and to publish it to 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.

WMSMON tool uses the Round Robin Database (RRD) [15] as a backend database. Using this lightweight solution which does not require another running service storage requirements can be easily limited by design of database. RRD was written as a system to store and display time-series data in compact manner in the form suitable for producing time graphs. The data cache, like the cache on the Agent side provides latest values of monitored WMS properties for the WMSMON portal. WMSMON graph generator uses the data from RRD to produce daily, weekly, monthly and yearly graphs.

WMSMON web portal presents information from different WMS sources in a unified way. Main page provides the aggregated status view of all monitored WMS services from the target Grid infrastructure. Data is shown in simplified way in this part of the portal with the emphasis on WMS services identified not to work properly. Portal also provides links to pages with detailed information and graphs for each monitored WMS service. WMSMON web portal is shown in Figure 7.

Figure 7: Screenshot of WMSMON web portal.

The WMSMON Collector is implemented as a Linux daemon and during its startup it creates appropriate cron jobs, initializes RRD database 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.

C. WATG

The WatG Browser (What is at the Grid Browser) is a web-based Grid Information System (GIS) visualization application providing detailed overview of the status and availability of various Grid resources in a given gLite-based e-Infrastructure. It is able to query and present data obtained from Grid information systems at different layers: from local resource information system for a particular Grid service (GRIS), to the Grid site information system (site BDII), and to the top-level information system for the whole Grid infrastructure (top-level BDII).

The efficient implementation of WatG Browser allows quick and easy navigation through entries and objects of the LDAP tree retrieved by the specified query, even if the size of the output is huge and hierarchically very complex. Highly responsibility is achieved with implementation of partial refreshes and asynchonization of a web page. A partial refresh of WatG application can be observed when an interaction event is triggered, for example click on the plus icon of the LDAP tree. The server processes the information and returns a limited response specific to the data it receives, for example LDAP's subtree that requires given condition. One may notice that WatG server does not send back an entire page, like the conventional "click, wait and refresh" web applications. Instead, WatG client updates the page based on the response. This means that only part of the page is updated. In other words, WatG's initial page is treated like a template: WatG server and client exchange the data and the client updates parts of the template based on the data it receives from the server. Another way to think about it is to consider WatG application as driven by events and data, whereas conventional web applications are driven by pages.

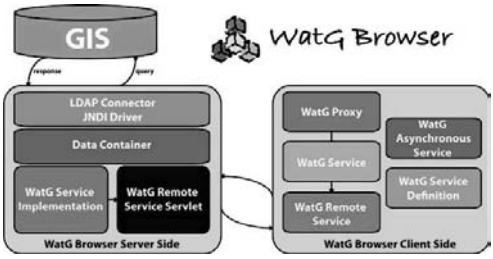

Figure 8: WatG architecture.

Asynchronization of the WatG application is reflected in the fact that after sending data to the server, the client can continue processing while the server does its processing in the background. During all this, a user can continue interacting with the client without noticing interruption or a lag in the response. For example, a user can click on any plus or minus icon even during the loading, and in that way a new request will be created and executed afterwards. The client does not have to wait for a response from the server before continuing, as is the case in the traditional, synchronous approach. Architecture of the WatG Browser is given in Figure 8.

The above main features of WatG application introduce also many secondary ones. For example, a partial refresh hides huge complexity and amount of data stored in a Grid Information System, which have to be transferred from the server to the client. Therefore, WatG is able to browse large LDAP directories keeping out of sight its nontrivial structure and size.

Figure 9: WatG front-end.

Current WatG front-end performs a search using the filter. Search filters enable defining search criteria and provide more efficient and effective searches. The filter should conform to the string representation for LDAP filters. The WaiG front-end also contains attribute field, which returns only selected entries and values (if specified).

WatG application is written in the Java programming language. GWT [16] (Google Web Toolkit) is used to cross-compile it into the optimized JavaScript that automatically works on all major browsers. WatG user front-end is shown in Figure 9.

D. Pakiti

Pakiti tool provides a monitoring and notification mechanism for checking the patching status of installed packages on an RPM-based Linux system. Pakiti is using a client/server model, in which clients and servers are exchanging information using HTTP(S).

Once installed on a client host, through a cron job, Pakiti will check each night if new patches are available and report them to the relevant Pakiti Server(s).

As a result, there will be no change on clients (i.e. no packages will be updated/installed on clients unless configured by the administrator), but Pakiti will maintain a web page providing a list of all registered systems and the list of the pending patches for each of them. This helps the system administrator keeping multiples machines up-to-date and prevents unpatched machines to be kept silently on the network.

While Pakiti clients are sending data to Pakiti servers, the Pakiti servers can also send each other general statistics or detailed reports. The list of trusted Pakiti servers is configurable in the server configuration file. The SCL Pakiti main web page is shown in Figure 10.

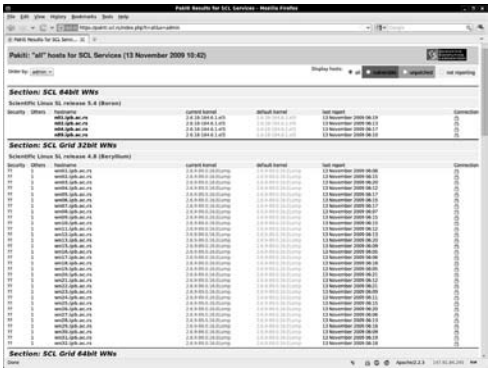

Figure 10: View of the SCL Pakiti main web page.

E. Ganglia

Ganglia is a scalable distributed monitoring system for high-performance computing systems, such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR [17] for compact, portable data transport, and RRDtool for data storage and visualization. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. The implementation is robust, has been ported to an extensive set of operating systems and processor architectures, and is currently in use on thousands of clusters around the world. It has been used to link clusters across university campuses and around the world and can scale to handle clusters with 2000 nodes.

In everyday system monitoring Ganglia gives fast and reliable overview of the status of site nodes. It is easy to group nodes into groups that will be shown on Ganglia web interface.

Ganglia is a client-server based system. Gmond daemon is working on each monitored node collecting various data about OS conditions, and on server side gmetad daemon collects gmond outputs and publishes them on the web interface. It is easy to add new custom monitored parameters data into gmond daemon on ganglia clients by adding a cron daemon. It is possible to add parameters such as temperatures, number of running jobs on worker nodes or number of submitted jobs on Computing Element or WMS. Ganglia main page is presented in Figure 11.

Figure 11: SCL Ganglia main page.

F. SAM

SAM (Service Availability Monitoring) is a framework used in EGEE [18] for the monitoring of production Grid sites. It provides a set of probes which are submitted at regular intervals, and a database that stores test results. In effect, SAM provides monitoring of grid services from a user perspective.

SAM uses an Oracle database to store the test definitions, node and site information, and the test results.

SAM web portal is a python-based web application to display and query the test results. Only users with valid certificates can view the pages. In addition, access is granted based on IP-address. Several configuration and customization options are available and instead of cookies, the portal will remember the user's settings from their certificate subject DN.

On the main page users can choose a service from a service list and Grid region where service is running (Figure 12). On the next page all nodes that match chosen service and region are displayed. After selection of a particular node, SAM test results are shown (Figure 13).

Figure 12: SAM Web Portal main page.

Figure 13: SAM test results page.

BBmSAM

For the needs of the SEE-GRID [19] series of projects, an alternative to the EGEE SAM framework is developed by the Faculty of Electrical Engineering of the University of Banja Luka and is currently deployed at the BA-01-ETFBL Grid site. It is completely based on non-commercial solutions (while the original SAM uses Oracle database) and is adjusted to the requirements of the SEE-GRID community. The architecture of BBmSAM is shown in Figure 14.

BBmSam is a web application implemented in PHP and relying on a MySQL database for data storage. This tool has been tested under different web servers (Apache, Microsoft IIS), and can be used with any web server supporting PHP (at list through CGI).

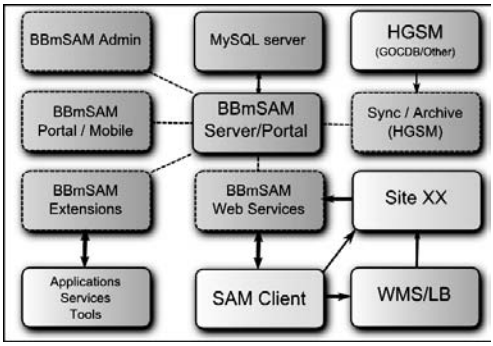

Figure 14: BBmSAM Architecture overview.

Main features of BBmSAM are:

- Use of unaltered client and sensor components of EGEE SAM system;
- Synchronization with central HGSM (Hierarchical Grid Site Management) [20] service - this service completely replaces EGEE GOCDB and eliminates the need of importing additional information from BDIIs and other sources;
- Use of free and open source technologies;
- Use of as few as possible different technologies to ease maintenance and development;
- Enabling more efficient access by mobile and small-screen devices.

Main components of BBmSAM are: database server, synchronization service, BBmSam web services, BBmSAM portal, BBmobileSAM.

Database server is based on MySQL, providing as much transparency as possible and enabling easy migration to a different database server. The database schema was kept as close as possible to the original SAM DB, with only few necessary changes. Two new tables were introduced: one for site downtime data and the other for uptime calculations.

BBmSAM web services are implemented in PHP with the use of NuSOAP [21] library. It mimics the original SAM WS component to enable use of standardized clients. The two implemented web services are query and publish, where the first one is used for querying and filtering the data needed to run tests and second for publishing the test results to the central database.

BBmSAM synchronization service was implemented to synchronize SAM database with the remote HGSM server. Synchronization is done directly from HGSM export to the SAM DB. It is possible to use different data sources (GOCDB/others) [22] as long as there is a way to generate proper export. Synchronization process has three steps:

Generating XML export on HGSM server; Importing and transforming XML data according to the SAM DB specifications; Generating HGSM "snapshot history" on the local server (optional).

BBmobileSAM is a specialized portal for devices with small screens and no support for full HTML. It consists of only the basic information (with three different levels of details and color-coding the results) and uses very small subset of HTML that enables it to be used on almost any mobile device.

BBmSAM portal component enables simple and efficient access to all data stored in BBmSAM database, including current and historical data. Front page of the portal is shown in Figure 15.

Main part of the front page is a table with summary results (overview table) that contains site name, country, tier, certification status and production type of each listed site. "SERVICE STATUS" column contains list of services, node name for respective service, latest critical test status and time since last critical test status change (uptime).

Figure 15: Front page of BBmSAM portal.

Services page displays the latest results for all instances of a specified service. The results table contains node name, site name, critical test status and statuses of individual tests. Tests are ordered by criticality so that critical test come first and they are marked by being bold the subject is mapped. Service page is shown in Figure 16.

BBmSAM system performs following operations:

- Periodical synchronization of local HGSM database with central HGSM database performed each 10 minutes.
- Regular SAM test submission performed each 3 hours for interactive tests (job based) and each hour for non-interactive tests.
- Publishing of interactive test data each 20 minutes.
- Calculating hourly uptime/availability each hour (for SEE-GRID-2 compatible SLA).
- Calculating service instance uptime (for continuous time SLA calculations in SEE-GRID-SCI).
- Generating information for end-users of portal on on-demand basis.

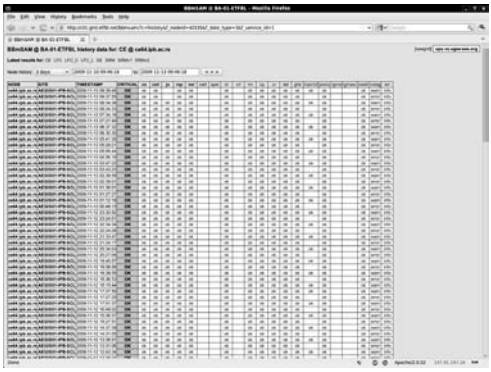

Figure 16: BBmSAM service page.

SAM client and sensors are the officially published client and sensors used in the standard EGEE SAM distribution and they operate in the same way.

Each monitored service is tested by a sensor, which consists of individual tests. Each performed test returns a status identification, which in turn defines the outcome of the test (e.g. ok/warning/error). The importance of all tests is not equal: some are require to be passed in order for the service provided by tested node to be seen as operationally functional. Such tests are designated as critical. Their outcome is logically combined using AND operation, and the status of a complete suite of SAM tests is the conjunction of all critical tests. However, the highest priority overriding any other is the MAINT status, which designates a site or service that is in the declared downtime (maintenance). For such services, results of SAM tests are published, but ignored in all SLA calculations. They also do not raise alarms on the front page of BBmSAM portal, nor on individual service pages.

G. GStat

GStat is an application designed to monitor EGEE/LCG compatible Information Systems. Its purpose is to detect faults, verify the validity and display useful data from the Information System.

GStat tests the Information System approximately each 30 minutes. The test relies on queries to site GIISes/BDIIs and not to any submitted job. This is done to gather information and perform the so-called BDII sanity checks to point out any potential problems with individual sites. The test covers the following areas:

Site and service information: Provide information about the site, services, software and VOs supported at that site;

Usage information: Provides the statistics on job slots, jobs, and storage space;

Information integrity: Checks if the Information system is publishing data that is meets specific syntax and value rules.

GStat runs on a single server and from this server, GStat executes queries, process the results and generates static HTML reports. Execution of queries and result processing are accomplished by agents and filters respectively

GStat agents are responsible for making queries and collecting raw data for further analysis by filter components. Filters execute test logic and generate processed data which is in turn used to create a web based test reports.

The configuration of GStat heavily depends on the data found in the GOCDB. GStat queries the GOCDB for the site's GIIS contact string, nodes information and other basic site information.

Gstat currently stores numeric data in RRD databases with data reduction. SEE-GRID GStat main page is shown in Figure 17.

Figure 17: SEE-GRID GStat main page.

On the main page summary table view shows site names and their most severe status of all tests associated with. Clicking on the site name will display the detailed GStat report for the site. Each site also has a small table cell to the right. This cell indicates and links to the results of the SAM Tests page. Multiple cells indicate that this site hosts multiple CEs.

Below the summary table is the table view which shows individual test results for each site. This table can be reorganized into different perspectives by with the sort by links at the top of the table.

At the bottom of the main page, one can find a total statistics table for entire instance. The 'Total' link will display graphs associated with these statistics.

Finally there is a detailed report generated for each site (Figure 18). At the top of the report, you will find links to the site's homepage, SAM results, GOCDB and graphs for all test result data associated with the site. The body of the report will consist of individual sections for each test performed and their detailed results. Each test section will display the name of the test, the results status, link to alert status history graph and help documentation for the test. The bottom of the report shows test data results in both tables and graphs. Long term graphs can be located by following the link for each graph.

Figure 18: GStat site page.

All of GStat tests respect the scheduled downtimes booked in GOCDB to alert level of the result status. If the whole site is in downtime, the site alert level is changed to the maintenance status. In addition, the test alert level of section 'GOC DB Info' in site report will be marked as maintenance, but all tests associated with the site will still work normally to present the real status and details of test result even though the site is in downtime. If some specific nodes of the site are in downtime, the test alert levels of the tests associated with such nodes will be marked as in maintenance, but the site alert level will not be affected. Particularly, the test section 'Service Check' in the site report will ignore the nodes in maintenance status in this section and retrieve the most severe status as the test alert level.

IV. Conlusions

We have presented a set of operational and monitoring tools used at the Scientific Computing Laboratory of the Institute of Physics Belgrade for overseeing two large Grid sites. Some of the tools are developed locally, and are provided to all interested site administrators through our SVN and RPM repository. All described monitoring tools are deployed at SCL, and are seen as vital for Grid operations. They provide essential information about Grid sites' and services' health not only to site administrators, but also to end-users.

Acknowledgment

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] SCL, http://scl.rs/
[2] CGMT, http://cgmt.scl.rs/
[3] WMSMON, http://wmsmon.scl.rs
[4] WatG, http://watgbrowser.scl.rs:8080/
[5] Pakiti ,http://pakiti.sourceforge.net/
[6] Ganglia, http://ganglia.sourceforge.net/
[7] SAM, https://lcg-sam.cern.ch:8443/sam/sam.py
[8] BBmSAM, http://c01.grid.etfbl.net/bbmsam/
[9] GStat, http://goc.grid.sinica.edu.tw/gstat/
[10] gLite, http://glite.web.cern.ch/glite/
[11] scl-sensors, https://http.ipb.ac.rs/tools/scl-sensors
[12] IPMI, http://www.intel.com/design/servers/ipmi/
[13] scl-jobs, http://http.ipb.ac.rs/tools/scl-jobs/
[14] grid-FTP, http://www.globus.org/grid_software/data/gridftp.php
[15] RRD, http://oss.oetiker.ch/rrdtool/
[16] GWT, http://code.google.com/webtoolkit/
[17] XDR, http://www.rfc-editor.org/rfc/rfc4506.txt
[18] EGEE, http://www.eu-egee.org/
[19] SEE-GRID-SCI, http://www.see-grid-sci.eu/
[20] HGSM, https://hgsm.grid.org.tr/
[21] NuSOAP, http://sourceforge.net/projects/nusoap/
[22] GOCDB, https://goc.gridops.org/