Monitoring the EGEE/WLCG Grid Services

Alexandre Duarte(1)(2), Piotr Nyczyk(1), Antonio Retico(1), Domenico Vicinanza(1)(3)
Firtname.Lastname@cern.ch

(1)CERN - European Organisation for Nuclear Research, Geneva - Switzerland
(2)Federal University of Campina Grande - Computer Science Dep., Campina Grande - Brazil
(3)University of Salerno - Dep. of Mathematics and Computer Science, Salerno - Italy

Abstract

Grids have the potential to revolutionise computing by providing ubiquitous, on demand access to computational services and resources. They promise to allow for on demand access and composition of computational services provided by multiple independent sources. Grids can also provide unprecedented levels of parallelism for high-performance applications. On the other hand, grid characteristics, such as high heterogeneity, complexity and distribution create many new technical challenges.

Among these technical challenges, failure management is a key area that demands much progress. A recent survey revealed that fault diagnosis is still a major problem for grid users. When a failure appears at the user screen, it becomes very difficult for her to identify whether the problem is in the used application, somewhere in the grid middleware, or even lower in the fabric that comprises the grid.

In this paper we present a tool able to check if a given grid service works as expected for a given set of users (Virtual Organisation) on the different resources available on a grid. Our solution deals with grid services as single components that should produce an expected output to a predefined input, what is quite similar to unit testing. The tool, called Service Availability Monitoring or SAM, is being currently used by several different Virtual Organizations to monitor more than 300 grid sites belonging to the largest grids available today. We also discuss how this tool is being used by some of those VOs and how it is helping in the operation of the EGEE/WLCG grid.

1 Introduction

Grids have the potential to revolutionise computing by providing ubiquitous, on demand access to computational services and resources. They promise to allow on demand access and composition of computational resources provided by multiple independent sources. On the other hand, grid characteristics, such as high heterogeneity, complexity and distribution (traversing multiple administrative domains) create many new technical challenges, which need to be addressed.

Among these technical challenges, failure management is a key area that demands much progress. Even fault diagnosis, a basic step in any failure management strategy, needs to see great improvement if we are to realise the grid vision. Today, when a grid user or administrator sees a failure in their screen, they have a very hard time in pinpointing the root cause of the failure. It may be the user's own application that has a bug, it may be that the user has requested a certificate whose lifetime was too short, it may be a configuration problem in some site that was used by the application for the first time or even that a disk on a machine next door has crashed. It may be a very large number of things. To further complicate things, error messages may be misleading. A recent study shows that even specialised users, such as system administrators, can spend as much as 25% of their time following wrong paths suggested by unclear error messages[2].

In this work we want to check if a given grid service works as expected for a given user or set of users on the different resources available on a grid. To achieve this objective we developed a framework that uses acceptance-like tests to help diagnose failures on the grid. In this new framework we deal with the grid services as single components that should produce an expected output to a pre-defined input. The framework is called Service Availability Monitoring or SAM, and is being currently used to monitor some of the largest (maybe the largest) production grids available nowadays.

We start by presenting in Section 2 a brief introduction on the gLite Grid Middleware, which is being used by the grid monitored with SAM. In Section 3 we present our solution, with some historical background and architectural de-

scriptions. Further, in Section 4 we present a case study of the use of our tool to monitor one of the largest (maybe the largest one) grid in production today. Section 5 concludes the paper with our final remarks.

2 The gLite Middleware

The gLite middleware [11][3] was born from the collaborative efforts of more than 80 people in 12 different academic and industrial research centres as part of the EGEE Project[10]. gLite provides a leading-edge framework for building grid applications tapping into the power of distributed computing and storage resources across the Internet. Its first version, gLite 1.0, was released on April 4th 2005. The latest version before converging toward a common architecture with the LCG[12] middleware was 1.5, released in January 2006. In May 2006 gLite 3.0 was released, merging LCG2.7 and gLite 1.5. This release contained all the services from LCG 2.7 with the addition of several components from gLite1.5. Starting from this gLite 3.0, there were no longer separate releases of the two middleware stacks it is being now used by EGEE and by WLCG (the worldwide LCG extension).

2.1 gLite architecture

The gLite services can be thematically be grouped into 5 service groups: Access Services, Security Services, Information and Monitoring Services, Data Services and Job Management Services. Among the gLite services one can distinguish user, site, virtual organisation (VO), and global (i.e. multi-VO) scope where combinations are possible (authorization policies may for instance be enforced by the VO and the site). Although most services are managed by a VO, there is no requirement of having independent service instances per VO; for performance and scalability reasons service instances will in most cases serve multiple VOs.

2.2 Executive summary of the services

Some gLite services coming from the 1.5 release were included in the gLite 3. The LCG workload management components are also available in gLite 3 and all the services are now accessible from both toolsets in order to ensure a smooth upgrade from the LCG 2.7 and gLite 1.5 to gLite 3.0.

Access and Security Services: The prime aim of the Access and Security Services is identifying users, allowing or denying access to services, on the basis of some agreed policies. It provides a credential having a universal value that works for many purposes across several infrastructures, communities, VOs and projects. To carry out this task, gLite uses the Public Key Infrastructure (PKI) X.509 technology using Certification Authorities as trusted third parties.

Information Service (IS) and Monitoring: The IS provides information about the gLite resources and their status. The published information is used to locate resources and for monitoring and accounting purposes. Much of the data published to the IS conforms to a schema that defines a common conceptual data model to be used for resource monitoring and discovery. All the LDAP URLs used to query the information services running in each site are stored in a database called GOCDB.

Job Management System, Resource Broker, Computing Element and Worker Node: The Job Management Services collects information about the resource usage done by users or groups of users (VOs). The up-to-date information about the Services/Resources is gathered via sensors (Resource Metering, Metering Abstraction Layer, Usage Records). Records are collected by the Accounting System (Queries: Users, Groups, Resource).

Within the services provided by the Job Management Service, the Computing Element (CE), represents some set of computing resources localized at a site (i.e. a cluster, a computing farm) that is responsible for job management: (submission, control, etc.) A CE provides a generic interface to the cluster, and the cluster itself, a collection of Worker Nodes (WN), the nodes where the jobs are run.

One of the most relevant services among the ones provided by the Job Management Services is the Workload Management System, a service running on a machine called Resource Broker. The RB is responsible for the distribution and management of jobs across sites. The purpose of the WMS is to accept user jobs, to assign them to the most appropriate CE, to record their status and retrieve their output.

Jobs to be submitted are described using the Job Description Language (JDL), which specifies, for example, which executable to run and its parameters, files to be moved to and from the worker node, input files needed, and any requirements on the CE and the WN.

Storage Element: The SE is the gLite component which takes care of the Data Services, providing a storage backend.

3 Service Availability Monitoring

subsectionHistory SAM is a monitoring system that was developed based on more than two years of experience with providing high level monitoring tools for EGEE/LCG

grid infrastructure. The concept of high level monitoring emerged in EGEE/LCG as the solution to manage the growing infrastructure that started with about 20 sites and quickly grew to 60, then more then 100 and ultimately beyond 200 computational sites. Number of sites and diversity of low level monitoring tools (a.k.a. fabric monitoring) made it impossible for a single operational body to know and understand the status of the whole grid and individual sites.

TestZone Tests The first approach to tackle this problem was a simple set of *bash* and *perl* scripts to perform a centralised testing of *Computing Elements* (CE) and the resources hidden behind, namely *Worker Nodes* (WN). The system contained the following components.

- set of *bash* scripts with WN tests based on site certification script provided in LCG release notes:
  - general environment tests: software paths, mountpoints, installed software,
  - Replica Management tests,
- skeleton of the test job (JDL file and main script) that can be submitted to a site and execute all the tests,
- set of *bash* scripts to submit test jobs to all sites, collect outputs as text files, parse output files and generate reports,
- *perl* CGI to process output files and generate HTML reports of test results.

The system had the following features:

- list of sites to test taken from BDII configuration file (a centrally managed file of officially certified sites),
- tests submitted as test jobs to all CEs in the grid from a single central UI,
- test jobs executed on WNs of all sites,
- results of tests returned as *output sandbox* of the jobs (HTML file with test log and "info" file with machine readable results as key-value pairs),
- full history of test results stored on a central machine (CERN AFS) as text files and available through web interface.
- the set of tests was customisable, however list of tests to be displayed on the web report was static (CGI configuration file).

Site Functional Tests Site Functional Tests (SFT) was a direct successor of TZTests. Name was change to better describe the functionality of the system and to avoid references to the concept of *Test Zone*, which was abandoned. The list of significant changes included:

- tests submission and results collecting code was rewritten from scratch,
- list of sites to test was being taken from GOCDB (using HTTP and regularly updated text file) and top-level BDII,
- several new tests were added,
- concept of critical and non-critical tests was introduced, only tests considered to be critical could change the overall site status,
- programatical interface to site status information was added (*statustable.cgi*).

SFT2 As a consequence of emerging Grid Operations a number of new monitoring tools was developed and popularised. One of the most important of them was GStat, a tool to monitor and analyse grid information system, namely BDII. To enable a common monitoring platform that would allow sharing test results and monitoring information between SFT and GStat, a new version of SFT with major architectural changes was introduced. SFT2 included the following new features:

- a universal relational data schema was designed to provide abstract representation of monitoring data, suitable for SFT, GStat and potentially other monitoring systems,
- test job scripts which became difficult to managed were converted into a single master script with individual tests as independent executables returning results in a common format,
- a basic pre-execution checks were added to validate test environment and minimise false alarms before launching the tests against sites,
- test results were published directly from WN by the job master script using *perl* publishing client, SOAP protocol over HTTP and *publishing web service* running on *SFT Server* machine,
- test results were stored in the MySQL database and published to R-GMA [1]

[1] In fact MySQL database was shared by SFT2 server and R-GMA archivers, but the data was directly inserted into MySQL by the *publishing web service*

- during the evolution of SFT additional "dimension" was added to test results which was the executing VO, and as a consequence the data schema was extended accordingly.

Service Availability Monitoring After EGEE/LCG grid infrastructure had grown in terms of number of sites but also number of different service types, it become clear, that the model imposed by SFT[2] is not giving enough information about the status of all important site services such as *Storage Elements*, *LCG File Catalogues*, etc. Because of the data model inherited from R-GMA[3] the performance of SFT started to slowly degrade and the database soon became difficult to maintain. In addition there was intensive ongoing development of third-party monitoring frameworks for EGEE/LCG that provided complementary information to SFT and/or covered services or areas not monitored directly by SFT. That is why a decision was made to extend SFT to a new system called Service Availability Monitoring (SAM) that would provide required features:

- optimised database schema for storing and processing test results,
- concept of *sensors* as containers of tests targeted against different types of grid services,
- concept of *standalone sensors* as third-party monitoring frameworks or test suites that could publish test results into SAM database in uniform format,
- integration with other monitoring and operational tools like: FCR (critical tests), CIC Portal (alarm system), GridView, etc.
- automatic service and site availability metrics calculation per VO based on critical tests selection,

3.1 Architecture

SAM is a system which although functionally replaces and extends SFT, but was redesigned almost completely from scratch based on the vast experience gathered during work on SFT and SFT2. The architecture of the system is shown in figure 1. The system which has a number of specialised components was logically divided into three independent layers: input, data storage and processing and output.

[2]SFT despite its name was only monitoring a single type of sites' components, namely *Computing Elements*

[3]R-GMA imposes semi-relational data model which lacks many features typical for relational data model like: table normalisation, integrity constraints, customisable indexing.

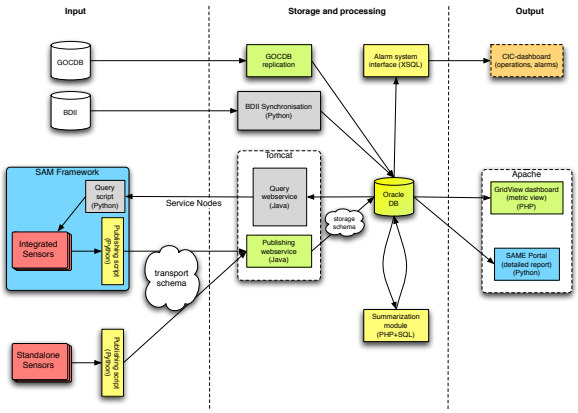

Figure 1. SAM Architecture

3.1.1 Input layer

The input layer mostly consists of components responsible for exectuing regular tests against all grid services and delivering results. There are two possibilities: either tests are executed and results published by the default component which is SAM Submission Framework[4], or the equivalent functionality is provided by the standalone monitoring tool that is publishing the results into SAM.

SAM Submission Framework is a software package which provides a uniform platform for executing the tests and publishing test results to the central database. It communicates with the system through web services and provides command line utilities to perform two simple operations on the underlying database:

query the database for information about grid infrastructure description (sites, nodes, services, VOs) using an abstract high level query format,

publish the results to the database by using a simplified *transport data schema*.

In addition the environment of SAM Submission Framework provides standardised elements like: sensors' directory structure and communication protocols, set of environmental variables (status codes, directories), common filter format for selecting nodes, test wrapper script with timeout mechanism.

All the sensors in SAM are plug–in modules that communicate with the Framework using fixed protocol. In the design of SAM Submission Framework we introduced two levels of hierarchy: sensors as containers and tests as individual code units (executables) which usually produce a single result record. The architecture of SAM Submission Framework is shown in figure 2.

[4]also referred as SAM Client, internally called `lcg-sam-client`

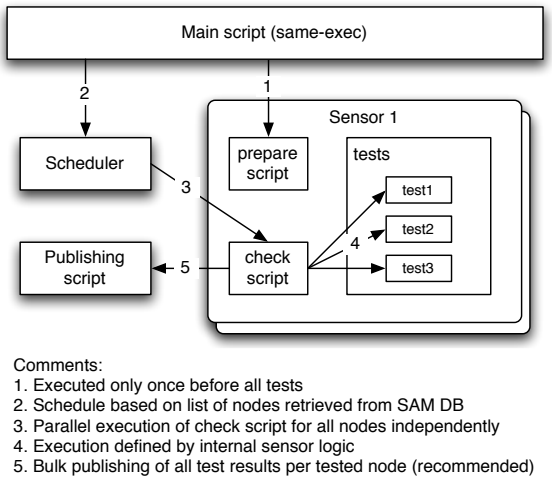

Figure 2. SAM Submission Framework

3.1.2 Storage and processing

The components of SAM which are responsible for collecting monitoring data, storage and post-processing are installed on a central machine called *SAM Server*. The core of the system is the relational database[5] which holds all the information like: Grid infrastructure description (sites, nodes, services, VOs and relations between them), test results, test criticality, availability metrics and application configuration. The components that interact directly with SAM database and are installed on SAM Server are the following:

Web services which are SOAP services providing methods to query SAM database for infrastructure description (*Query web service*) and to publish test definitions and results to the database (*Publishing web service*),

BDII Synchronisation script to discover sites and nodes in the grid information system,

GOC DB Replication script to replicate GOC Database[6] into SAM,

Summarisation module set of scripts to generate service and site status summaries and availability metrics,

Alarm system set of PL/SQL procedures, triggers and XML data exports exposed on HTTP to automatically generate alarms on failures of critical tests and give programmatic interface to the alarms.[7]

[5] In current implementation only Oracle DBMS is supported

[6] EGEE/LCG specific database

[7] used in EGEE/LCG to interface with COD Dashboard which is the main operational tool

3.1.3 Output

The presentation layer of SAM contains a number of components that are accessing SAM database directly or indirectly (through XML data exports) and are even parts of external systems. The most important components are the following:

SAM Portal a reporting tool written in Python that displays individual test results by VO, service type, and region, as an HTML table with possibility of showing history of test results and the detailed log from test execution,

GridView visualisation portal which shows configurable availability plots (intervals, VO-wise, site-wise, etc),

COD Dashboard an external portal for Grid operations in EGEE/LCG which is a front end to the alarm system and ticketing system.

3.2 Data schema

The data schema for SAM was design to give maximum flexibility for integration various existing monitoring tools in EGEE/WLCG run by different VOs. It is split into two parts: infrastructure description (used by *query* web service to get information about nodes to monitor) and test results. For publishing test results SAM defines two levels of data schema: transport schema and internal storage schema.

Transport data schema is meant to be used in communication between monitoring tools like: SAM Submission Framework, sensors, standalone monitoring tools. As a consequence it is designed to be a relational data model with the tables in 1st Normal Form (1NF). It defines three tables which are used for definitions and automatically discovery of tests and for publishing test results.

The data transported using the transport data schema is then received by the *publishing* web service and transformed into internal storage schema which is mostly in 3rd Normal Form and is used in underlying RDBMS. The description of internal storage schema and infrastructure description schema is outside the scope of this paper.

Below we present the definitions of the tables in transport data schema.

TestDef Each test performed should be registered in the test table. Here additional help information on the test and the data produced can be looked up. New tuples should be published in this table in case:

- a new test is introduced and its definition has to be published, or

- an existing test was modified (for example dataThreshold was changed for performance test) and the definition needs to be updated

Consequently, tuples for TestDef table should be published with rather low frequency but the full history will be kept (for analysis of historical results). The attributes of TestDef table are shown in table 1.

Field	Description
testName	short name of the test process
testTitle	full title of the test
testAbbr	user friendly name as column header
testHelp	test description or URL to help page
dataType	type of data produced by test
dataUnit	unit of measure for the test
dataThreshold	threshold on value causing failure

Table 1. TestDef table

TestEnvVars The TestEnvVars table is where the monitoring frameworks (SAM, GStat,etc..) may store additional information about the environment (RB, BDII, central SE, etc.) of a single "test session". A single test session usually corresponds to a single test cycle as a bunch of test jobs executed against all sites in short time period. It should be identified by unique "envName" and by inserting multiple tuples with the same "envName" into this table, it is possible to specify various parameters of the environment. The attributes the table are shown in table 2.

Field	Description
envName	identifies a single "test session"
name	name of the parameter (RB, etc.)
value	value of the parameter (RB hostname, etc.)

Table 2. TestEnvVars table

TestData The TestData table is where all the results produced by tests and sensors are stored in a uniform format. The attributes the table are shown in table 3. The definition of valid status values for test results is shown in table 4.

3.3 Availability metrics

Based on test results stored in the database SAM is calculating a current status of service instances of all the types and also an overall status of sites. In addition for all service instances and sites the availability metric values are calculated. The calculation is performed by the summarisation module, and a separate set of statuses and metric values are generated for each VO independently.

Field	Description
voName	VO that performed the test
testName	identifies test used to obtain data
nodeName	identify node tested to produce data
timestamp	Measurement timestamp
envName	Foreign key in TestEnvVars table
status	status of the result (see table 4)
summaryData	short test result data representation
detailedData	additional detailed information (log)

Table 3. TestData table

Code	Status	Description
0	**na**	no status available
10	**ok**	normal status
20	**info**	useful information
30	**note**	important information
40	**warn**	failure may happen soon
50	**error**	failure of local service only
60	**crit**	fatal failure affecting other services
100	**maint**	subject is under maintenance

Table 4. Test results status codes

A current status for each service instances is calculated as a logical AND of all the test results for the node. This is done for each VO independently by taking only those tests which were selected by the VO as critical. As a result each service instance is marked either as *available* or *down* for each VO independently.

After calculating status of all service instances for all VOs, SAM is calculating the overall site status for all sites by taking into account all service instances in a given site. A site is marked as currently *available* if at least one of the service instances of each type is *available*, and is marked as *down* otherwise. This is also done on per VO basis.

All the statuses are stored in the database as hourly snapshots of the current situation in the grid. Based on the snapshots the summarisation module calculates availability metric value of each service instance and site for each VO. The value is calculated as a fraction of time (percentage of snapshots) in selected integration interval, when service instance or site was *available*. Currently the summarisation module calculates and stores availability metric values for the following intervals: day, week, month.

4 Case Study: Monitoring a Global Grid

A number of grid infrastructures are currently featured by SAM. As major examples we mention here those built within the WLCG/EGEE[7], SEE-Grid [15], EELA[6], Health-e-Child[5], EuMedGrid[9], EuChina Grid[8], Baltic Grid[1] projects. In these contexts, which are in general different for scale, scope and purpose of the infrastructure, SAM platforms were deployed in slightly different configurations, according mainly to the number of sites monitored and to project's hardware and software resources.

On account of the number of its sites and Virtual Organizations, of its geographical spread, and of the complex structure of its operations, the WLCG/EGEE grid is by far the largest grid infrastructure among those featured by SAM services.

In the following sub-sections we will give a short overview on the overall organization of the operations in the WLCG/EGEE grid. Then we will explain in more detail how the different players make use of SAM features in order to perform their functions within their respective contexts. Finally we will provide a quantitative view of the impact that the introduction of SAM/SFT gave to the overall quality of the WLCG/EGEE grid services.

4.1 Grid operations in WLCG/EGEE

The WLCG/EGEE infrastructure is based on grid services provided by more than 200 sites distributed all over the world. The users of these services are organised in more than one-hundred different Virtual Organizations. The overall quality of service guaranteed to the VOs is defined by MOUs (Memorandum of Understanding) stipulated between the individual VOs and the supporting sites. In order to assure the overall quality of the service provided both in terms of availability and performance, the WLCG/EGEE infrastructure was organised into 10 *federations* or *regions*. In order to coordinate the operations throughout the regions, an operational process was defined based on two main players: a network of Regional Operational Centres and the Grid Operators.

The Regional Operation Centre (ROC) holds the overall responsibility for the services run within its region. This means, in practical terms, to make sure that all the sites in the region are operated in conformity to a set of agreed operation procedures. From the EGEE project's perspective, the ROCs represent "the infrastructure". From the site's perspective, the ROC is the project's reference point where to obtain information about project-wide policies and requirements, technical advices, recommendations and general support.

The Grid Operators (COD) are a distributed team in charge of providing an active and continuous monitoring of the availability and performance of the grid services. The key function of the COD is to detect issues affecting the grid services, to provide possibly a first analysis, to report existing problems to the relevant ROCs (generally via service tickets) and, finally, to validate the solution. The COD team also assures the service tickets assigned to the ROCs to be followed-up correctly and in the due time. The interaction between ROC and COD is instrumented by an agreed escalation procedure, which defines exactly the expected times for the resolution of classes of problems, as well as the measures to deal with unresponsive sites, these measures ranging from the reminder e-mail to the suspension of the site. The escalation procedure, as well as the Operation Manual, are developed and maintained in collaboration by the ROCs and the COD.

The overall activity of ROCs and CODs is supervised and coordinated within the EGEE project by an Operations Coordination Committee (OCC), which is in charge of monitoring the WLCG/EGEE operations as a whole and reporting to the project management.

The ROC and SAM

In order to be integrated in the WLCG/EGEE grid, a site has to demonstrate its technical suitability to run grid services at a convenient level of quality and, more important, not to introduce unexpected perturbations in the grid. Just for example, it is technically possible to configure a grid site in such way that, independently on the actual amount and quality of its resources, it starts attracting user jobs in what is called a "black-hole" effect. In order to prevent this one well as other intentional or accidental security issues, each site that wants to join the WLCG/EGEE grid has to undergo a preliminary testing and validation of it services, generally indicated as "site certification", to be done by the relevant ROC. The ROC, which will be responsible for the site throughout its future life, is given by the EGEE project the responsibility of defining and implementing a convenient certification procedure for sites in the region.

On account of SAM (and its earliest version SFT) being among the first utilities available in the WLCG/EGEE context able to test the overall functionality of a whole grid site, almost all ROCs gave to SAM's results a predominant relevance in the definition of their certification procedures. In the most common scenario, as one of the conditions for a site to be certified, the ROC wants a site to successfully pass the default SAM test across some days before allowing it to receive production jobs. The main technical difficulty in using SAM for certification purposes is that the production instance of SAM, by definition, does not automatically submits tests to uncertified sites. In the aim of helping the ROCs in the implementation of SAM-based certification tests a centralised utility was set-up to enable on-demand submission of SAM tests to any registered sites in WLCG/EGEE. This utility is known as the*SAM Admin's*

page [16].

The COD and SAM
The function of the COD is currently covered by 10 specialised ROCs alternating in weekly shifts. Members of the COD have got a number of tools available to support their activity. Specifically, in order to monitor the grid service availability, the main tool is the *COD Dashboard*, developed in the framework of the CIC portal. The COD Dashboard raises alarms using at this purpose data provided by SAM. The alarms are then followed-up by the CODs via service tickets. The COD dashboard gives also access, for further analysis, to site availability data and test results made available from the SAM database.

4.2 Monitoring activity by Virtual Organizations

In addition to the "institutional" monitoring activity done by the CODs, also some of the Virtual Organizations take active part in grid service availability monitoring by providing, maintaining and running specialised tests for VO-specific grid applications. Usually these tests are run, within the SAM framework, across a subset of sites, namely the ones supporting the VO. These tests are in general developed independently by working groups in the VOs, and therefore they are not under the configuration control of the standard SAM platform.

All the four LHC experiments are deploying or planning to deploy a set of custom tests within the EGEE/WLCG SAM production infrastructure. The general aim of these test is to run sanity checks against both middleware and application services. In particular, at the time when this paper is being written, the VOs *cms*, *alice* and *lhcb* , corresponding to the homonymous LHC experiments, are already running their own SAM tests in production.

CMS from one side and Alice and LHCb from the other used two different strategies in order to integrate their tests in SAM. We can classify these two methods, respectively, as a sort of "advanced usage of the Submission Framework", chosen by CMS, and an "hybrid submission methods" implemented by ALICE and LHCb. we will now briefly go some details concerning each of the mentioned VOs.

CMS is submitting custom tests to the CEs since the beginning of 2007 [14]. SAM tests are used to validate features concerning both the CMS applications (e.g., the existence of the CMS software area, the local configuration of CMS sites, the CMS software version, the ability to stage out files from WN to local SE, the discovery of local Squid server, the ability to read calibration data via Squid server) as well as the middleware (namely SRMv1 and v2 interfaces with respect to the translation LFN/SURL, data access from UI to remote SE).

CMS integrated custom tests by simply using standard features of the submission framework. (CITA) They operate a SAM client, installed at CERN, where custom tests were plugged into the existing SAM CE sensor. The lifecycle of CMS test in SAM is exactly the same as for standard tests. On top of that, CMS is calculating independently the perceived site availability according to user-oriented metrics. In particular they consider the "Availability" of a given site as the fraction of *up-time/total-time* (so not considering downtimes) The downtime is considered, instead, in the calculation of site *Reliability*, defined as *(site up-time)/(total-time - scheduled downtime)*

Alice is mainly interested in using SAM to submit functionality tests to VOBOXES [8]. The tests results have then to be displayed in MonALIsa, the tool used by Alice to monitor the complex of the VO applications [13].

Alice uses a different approach in order to submit custom VOBOX tests: basically tests are scheduled and submitted and results are retrieved by proprietary application of Alice's. Test definitions, conditions (environment) and results are then inserted in SAM database using the SAM transport schema. It is worth pointing out that, from that moment on, test records pertaining from Alice submissions are formally identical to records inserted by the standard submission framework.

By accessing the records in the SAM database through the Programmatic Interface an by making them available to MonALIsa for displaying, Alice finalised the original test requirement. The added value of using the SAM platform is that, by doing so, Alice gets in addition all the standard fetures provided by the SAM framework (e.g. visualization and availability sums in GridView). Moreover test data are permanently available in the database, to be easily interfaced with other application via the Programmatic Interface.

LHCb adopted a very similar approach to Alice's, by using DIRAC to submit custom SAM tests [4]. Critical test objects for LHCs are e.g., the length of LHCb queue on the CE, the architecture and OS version, the whole MonteCarlo chain as implemented by LHCb applications, installation of LHCb software and publication of tags. Then as far as the grid services are concerned, LHCb runs also custom tests against the SRM storage elements.

In the LHCb submission model, the whole SAM client is shipped on the Worker Node at the site, runs the tests locally and publishes the results in SAM DB using the standard SAM protocols. This particular submission method allows

[8]The VOBOX is a component of the gLite distribution to be listed among the "site services". It consists essentially of a box, equipped with a full collection of grid clients (grid User Interface), providing in addition a controlled and secured environment where application servers part of the VO software are allowed to run.

LHCb to take advantage of both the SAM features (displays, programmatic interface, notification), together with the ability of closely monitoring the evolutions of SAM jobs by using the DIRAC built-in job-tracking and logging features.

The integration cases described above are different as conception and strategy, but however all successfully implemented, which is a confirmation of the flexibility and extensibility of SAM's open architecture.

A thing to be pointed out is that, all the intense development activities started by the WLCG/EGEE VOs, aimed to creation and integration of new VO-specific tests, did not entail, so far, any modification's in the SAM "core" database schema, originally designed with an eye to possible external contributions to the monitoring features offered by SAM.

4.3 Measuring SAM effect

In terms of positive impact on the day-to-day grid operations, in the relatively short history of grid monitoring, and specifically dealing with the WLCG/EGEE context, we must say that the most significant improvement in the overall availability and stability of the WLCG/EGEE grid was undoubtedly reached before SAM went to operations, when the monitoring activity was featured by SAM's predecessor, SFT (see 3).

SAM took over SFT in production one year ago in order to better scale with the increasing number of monitored sites and tests to be run. In order to prove that SAM is indeed scaling quite well, we analysed the test results concerning the last year of operation, starting from the date when SAM went in production.

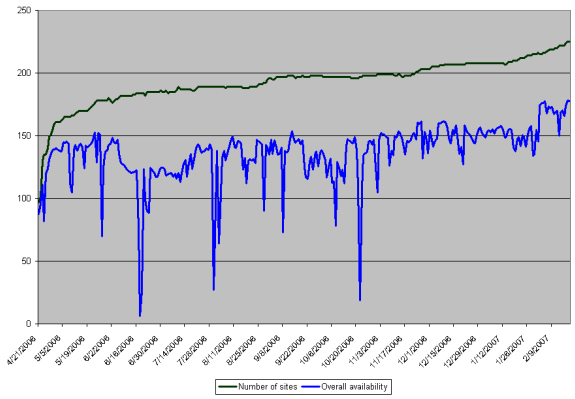

Figure 3. SAM availability

Figure 3 shows two plots. The upper one is the number of sites registered to the WLCG/EGEE project; the lower one is an indicator of the overall service availability. Specifically the overall service availability is calculated as the sum of the daily availability metric of each sites in production, being the daily availability metric for a site is the percentage of time in the day during which the site has been fully operational. We tried and reduce the negative bias on the sum due to test results coming from still uncertified sites. Results of tests run across uncertified sites, in fact, are mixed-up in the database with those of certified sites. Since an historical record of status transitions for a site in not currently available, in order to correct this effect we decided to count in the sum the contribution of a given site starting from the first day in which the daily availability for that site resulted to be equal to 1. The consideration behind this assumption is that, when a site is available continuously for 1 day it is reasonable to foresee that its initial set-up phase is over and the site is getting close to the certification . In this way, without making a neat distinction between certified and uncertified sites, we meant to measure the "SAM effect" in terms of usefulness of the tool to the "operations" in broad sense, which means to the certification activity done by the ROC, as well as to the monitoring activity done by the COD. As an indirect proof of the robustness and scalability of the monitoring tool, Figure 3 demonstrates how the increasing number of monitored sites (about 50% more than the precedent year) did not affect the overall availability of the grid services although the manpower and effort spent for the monitoring activity was basically left unchanged.

5 Conclusion

We presented a framework that uses acceptance tests to help diagnose failures on the grid. In this new framework we deal with the grid services as single components that should produce an expected output to a pre-defined input. The framework, called Service Availability Monitoring or SAM, is being currently used to monitor some of the largest (maybe the largest) production grids available nowadays and proved to be helpful on improving the reliability of the monitored grid services.

References

[1] BalticGrid. Develop and integrate the research and education computing and communication infrastructure in the baltic states into the emerging european grid infrastructure. In *http://www.balticgrid.org*, 2007.

[2] R. Barrett, E. Haber, E. Kandogan, P.P. Maglio, M. Prabaker, and L.A. Takayama. Field studies of computer system administrators: Analysis of system management tools and practices. In *Procceedings of Computersupported Cooperative Tools*, pages 388–395. ACM Press, 2004.

[3] S. Burke, S. Campana, A. D. Peris, F. Donno, P. M. Lorenzo, R. Santinelli, and A. Sciaba. glite 3.0 users guide. In *https://edms.cern.ch/file/722398/1.1/gLite-3-UserGuide.pdf*, 2006.

[4] J. Closier. Ensuring grid resource availability with the sam framework in lhcb. In *Proceedings of CHEP2007*, Dirac House - Temple Back Bristol BS1 6BE, UK, 2007. IOP Publishing.

[5] Health e child. An integrated platform for european pediatrics based on a grid-enabled network of leading clinical centres. In *http://www.health-e-child.org*, 2007.

[6] EELA. E-infrastructure shared between europe and latin america. In *http://www.eu-eela.org*, 2007.

[7] EGEE. Enabling grids for e-science. In *http://www.eu-egee.org*, 2007.

[8] EUChinaGrid. the euchinagrid initiative. In *http://www.euchinagrid.org*, 2007.

[9] EUMEDGrid. Empowering escience across the mediterranean. In *http://www.eumedgrid.org*, 2007.

[10] F. Gagliardi, B. Jones, F. Grey, M.-E. Bgin, and M. Heikkurinen. Building an infrastructure for scientific grid computing: status and goals of the egee project. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 363(1833):1729–1742, 2005.

[11] The gLite Team. glite: Lightweight middleware for grid computing. In *http://cern.ch/glite*, 2007.

[12] M. Lamanna. The lhc computing grid project at cern. In *Proceedings of the IXth International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, pages 1–6, November 2004.

[13] L.Betev, P.Buncic, A.Peters, P.Saiz, S.Bagnasco, P.Mendez-Lorenzo, C.Cistoiu, and C.Grigoras. The alice grid - the beat of a different drum. In *Proceedings of ACAT2007 (XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research)*, Amsterdam, NL, Apr 2007. IOP Publishing.

[14] A. Sciaba', S. Campana, A. Di Girolamo, E. Lanciotti, N. Magini, P. Mandez Lorenzo, E. Miccio, and R. Santinelli. Testing and integrating the wlcg/egee middleware in the lhc computing. In *Proceedings of CHEP2007*, 2007.

[15] SEE-GRID. South eastern european grid-enabled einfrastructre development. In *http://www.see-grid.org*, 2007.

[16] Poznan Supercomputing and Networking Center. Sam admin's page. In *http://monitoring.egee.man.poznan.pl/menu.php*, 2007.