Service Availability Monitoring

Pedro Andrade, Marian Babik[1], David Collados, Paloma Fuente, Wojciech Lapka
European Organisation for Nuclear Research CERN
CH-1211 Geneva 23, Switzerland
E-mail: Pedro.Andrade@cern.ch, Marian.Babik@cern.ch, David.Collados@cern.ch, Paloma.Fuente@cern.ch, Wojciech.Lapka@cern.ch

Emir Imamagic
University Computing Centre SRCE
Josipa Marohnica 5, 10000 Zagreb, Crotia
E-mail: Emir.Imamagic@srce.hr

Christos Triantafyllidis
AUTH SCC/GRNET S.A.
Aristotle University of Thessaloniki – Thessaloniki GR 541 24
E-mail: ctria@grid.auth.gr

The grid infrastructure for the CERN (European Organization for Nuclear Research) LHC (Large Hadron Collider) continuously operates thousands of grid services scattered around more than 300 computer centres providing more than 340,000 cores. Participating sites are organized in regions and support more than 270 virtual organizations running different middleware distributions, thus creating a very complex and heterogeneous environment. The Service Availability Monitoring (SAM) framework is responsible for the monitoring of this infrastructure.

[1] Speaker

EGI Community Forum 2012 / EMI Second Technical Conference,
Munich, Germany
26-30 March 2012

1. Introduction

The Worldwide LHC Computing Grid (WLCG) [1] project is a global collaboration linking grid infrastructures and computer centres worldwide. It was launched in 2002 to provide global computing resources to store, distribute and analyse the 15 Petabytes of data annually generated by the Large Hadron Collider (LHC) [2], the particle accelerator at CERN, on the Franco-Swiss border. The infrastructure built by integrating thousands of computers and storage systems in hundreds of data centres worldwide enables a collaborative computing environment on a scale never seen before. WLCG serves a community of more than 8,000 physicists around the world with near real-time access to LHC data, and the power to process it, regardless of the physicists' physical location. The resources are distributed across the world for funding and sociological reasons and the infrastructure is therefore managed by a worldwide collaboration between the LHC experiments and the participating computer centres.

The complexity of the WLCG is not only in the number of CPU cores or data that are part of it, but also in terms of interoperability among different middleware stacks and users from diverse countries grouped into different virtual organizations (VOs) [3], and the corresponding security layers to implement the required authentication, authorization and accounting. In this respect, operating the grid is not a trivial task and a real-time monitoring of services is vital not only to detect and correct defective services as soon as possible, but in general for running the LHC successfully.

This document describes the Service Availability Monitoring (SAM) framework, which is used daily for monitoring the WLCG. It is composed of several modules, some of them based on widely used open source technologies, what reduces the manpower required to maintain the code by having the support and development of big communities behind. At the same time, the monitoring is performed following a decentralized model and with components which scale up easily, what improves and simplifies the availability of the overall service.

2. Description of Architecture

SAM is a monitoring system that was developed after some years of experience providing high level monitoring tools for the WLCG grid infrastructure, in particular through the EGEE [4] European grid initiative. The concept of high level monitoring emerged during the EGEE project as the solution to manage the growing infrastructure that started with about 20 sites and quickly grew up to 60, soon after more than 100, and ultimately beyond 300 computational sites. The number of these centres and the diversity of low-level fabric monitoring tools made it impossible for a single operational body to know and understand the status of the whole grid and individual sites.

As a consequence of emerging different grid operational teams inside EGEE, it was clear that a global monitoring solution was needed to probe the different services, raise alarms to the sites in case of failures, and generate availability reports to evaluate the status and gradual improvement of the sites from an operational point of view. This was the origin of the SAM framework.

2.1 The Predecessor of SAM and Reasons for a Decentralized Model

The first implementation of SAM followed a centralized architectural model. This had some advantages in the past, like for example having a controlled and consistent set of tests that were submitted regularly (every hour) to all services using credentials of the Operational Virtual Organization (OPS VO), and storing all the monitoring results in a central location to compute status and availability of sites and services. Soon after, it was clear that some parts of this architecture had to be redesigned for at least the following reasons:

1) Reduced central operational effort: Originally SAM was 100% hosted centrally at CERN, and developed and maintained by staff funded by the EGEE project. With the end of EGEE in April 2010 and the emergence of the European Grid Initiative (EGI) [5] project, to enable access to computing resources for European researchers from all fields of science by promoting and enhancing National Grid Initiatives (NGIs), the responsibility of monitoring and operating regional grids moved to many different regions in Europe. Therefore, the existing SAM service had to be modified to fit the new distributed model.

2) Sites were 'blind' during central monitoring failures: As before, sites dependent on the central monitoring infrastructure for discovering problems had to rely on user complaints to discover problems in their services during downtimes of the central SAM.

3) Frequency of tests was low: Sites with no fabric monitoring were relying on the central SAM tests (submitted once per hour) to evaluate the status of their grid services. To improve the reliability of the grid alerting the site administrators faster in case of problems, changes were needed to execute tests at different and higher ratios.

4) Support for several infrastructures: The SAM framework was originally designed to test services in one single grid infrastructure. Trying to integrate and support the monitoring of other infrastructures like OSG [6], EGEE and NDGF [7] implied architectural changes, like for instance in the database schema.

5) Several algorithms to compute availabilities: The complexity of the grid, with a wide variety of users with differing goals, made VOs have different criteria for evaluating the availability of their sites. While SAM tests were addressing the availability of the infrastructure, some VOs were more interested in its usability (for instance ensuring that VO-specific software was in place). Therefore, a new architecture was needed to take this into account.

6) Missing test results during outages: The lack of a retry mechanism in the SAM clients when publishing test results to the SAM publisher web service made loosing test results during SAM outages. To address this, a technology to store and forward monitoring data while guaranteeing its delivery was needed.

2.2 The SAM Architecture

The new SAM framework has been redesigned completely from scratch to cover the deficiencies identified by the previous system and to accommodate to the new operational and maintenance effort within the EGI project, which started in 2010. In particular, in the era of

National Grid Initiatives, we have migrated the grid service availability infrastructure from the home-made one, run centrally from CERN, to one based on Nagios [8], a popular open source monitoring tool. During the development phase we were testing the system by running 11 regional Nagios instances from CERN. A version of Nagios for enhanced grid and fabric monitoring was also created for Site administrators. Today, 32 different EGI SAM Nagios Instances [9] run continuously from the different NGIs and ROCs (Regional Operation Centres), while a central project level database remains for availability calculations. By using a Message Bus technology, test results are available both within the regions themselves and at the project level, so that availabilities are computed and delivered to the project management board [17].

Now we look at the individual components of the architecture presented in Fig. 1, and how they connect to each other.

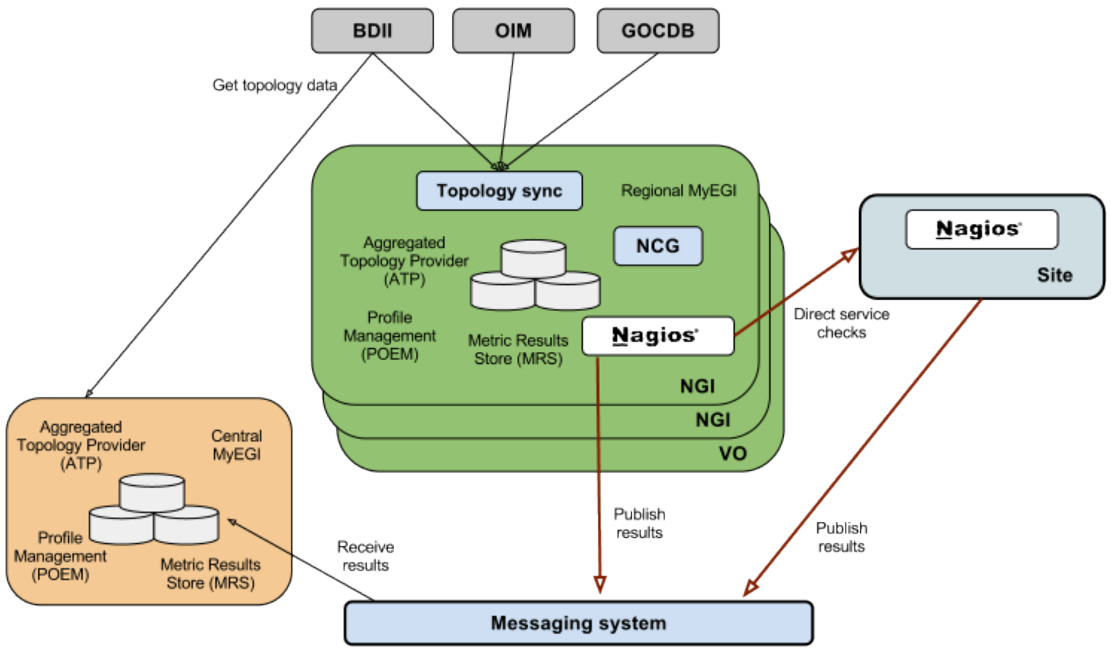

Fig. 1 Architecture of the SAM framework

The Aggregated Topology Provider, or ATP, is the component that gathers topology information from a list of authoritative sources such as GOCDB [10] for the EGI infrastructure, OIM [11] for the Open Science Grid, the BDII [12] information system, and VO specific topology (VO data feeds). The ATP supports both Oracle and MySQL databases.

The Profile Management System, or POEM, is a project level component that allows flexible availability calculations using different profiles. It describes the set of tests to be run (e.g. frequency of execution, timeouts, probe dependencies, etc.) and towards what kind of grid services. The site managers can use POEM to re-configure their local NGI Nagioses while at the same time VO managers can use it to change the set of critical tests for a corresponding VO. This set of critical tests, defined by the VO manager, will become synchronized to all regions within an hour, thus enabling fast grid-wide reconfiguration to match the requirements of the VO.

The Metrics Results Store is the repository of Metric Results for service endpoints, including their status changes and the detailed output from tests.

The Nagios Configuration Generator, or NCG, configures Nagios based on information from the ATP and POEM. It's the process that automatically describes the grid topology to Nagios, relieving site administrators from the burden of configuring the software.

Winner of numerous awards, Nagios is the industry-standard in open source IT infrastructure monitoring. Its core functionality has been extended for use in the Grid environment. It schedules and executes hosts and service tests, and generates alarms on status changes.

A network of message brokers, based in the Apache ActiveMQ [13] open source technology, is used for reliable communication between components. Message delivery is guaranteed fast and efficient.

Finally, the MyEGI portal is a customized web interface that uses some of the latest GUI technologies like Django [14], jQuery [15], AJAX [18] and JSON plugins [19]. It displays the current status of resources such as Computing or Storage Elements as well as historical data.

All the different components described tie together grouped in the following areas: configuration, tests scheduling and execution, data transport, and visualization. During the configuration phase, NCG gets information from the ATP and the POEM. Then, NCG builds a Nagios configuration. At this point Nagios can schedule and execute tests against the various grid services. During the submission process, some of the tests are run directly from the User Interface (UI) deployed in Nagios by contacting specific network ports on the target services. In other cases, like probing a Computing Element (CE), tests are submitted from the UI to the CEs via the Workload Management System (WMS) [16]. Finally, tests land on Worker Nodes (WNs), execute, and contact the messaging brokers to publish the results. From the messaging infrastructure, test results are then read into the Metric Store along with profile and topology information. Once the results are in the Metric Store, availabilities are computed in real time for the different profiles defined in the database for each VO.

3. The Intended Objectives

Regional Autonomy: The operational model of the project has changed from central (EGEE) to regional/national/local in EGI. The monitoring architecture should reflect this, as even though regional systems can be autonomous, they still need to share information with each other and central systems – standards and specifications are needed for this information exchange.

Scalability: As the grid grows, operational tools should be able to scale. Appropriate models (using hierarchical aggregation and partitioning) should be used to allow the system to scale avoiding bottlenecks.

Improve Grid Reliability by Reducing the Time to Respond to Problems: Minimising unavailability of services improves reliability. If the person who is able to fix the problem gets notified sooner, availability will tend to improve.

Reuse of existing open source software: There is a large set of open source tools available for monitoring, data visualization, data transport, etc. Using these reduces the amount of development work needed, and turns the problem into an integration problem.

4. The Obtained Results

4.1 Scalable Distributed System

The distribution of the SAM software is done through two main node types: a SAM-Gridmon (project level instance), and a SAM-Nagios (regional instance). The project level contains all monitored sites in the infrastructure, making easier an overall visualization and comprehension of their status, as generating periodic availability reports. On the other hand, SAM Nagios instances can be deployed at a regional or site level. A regional instance will only monitor a certain number of sites, those that fall under the responsibility of a ROC/NGI, while a Site Level Nagios instance will just monitor services at one particular site.

At present time, the WLCG infrastructure is monitored through one Project Level instance, 32 ROC/NGI Nagios instances, and more than one hundred Site Level Nagios instances.

Level of Nagios Monitoring	**Number of Sites**
Project Level Instances	399
ROC/NGI Level Instances	334
Site Level Instances	120

Table 1 Number of sites monitored by each monitoring layer

Table 1 shows the number of sites monitored or visible by each of the different layers. The current deployment scale can be seen in the table. It shows for instance, that the EGI ROCs/NGIs are currently monitoring 334 sites. The number is obtained by observing the number of sites for which metric results appear on the ActiveMQ message bus [13].

4.2 Messaging as an Integration Paradigm

Messaging is an attractive mechanism to simplify and extend several portions of the Grid middleware, from low level monitoring to experiments dashboards. As mentioned before, in SAM we have successfully integrated a message-oriented middleware (MOM) called Apache ActiveMQ to provide a transport layer for test results. This choice was taken based on research done within a WLCG monitoring working group on service monitoring. The performance and feature set of ActiveMQ was evaluated as relevant for the grid monitoring use case.

A production messaging service is currently being used by WLCG and is operated by the EGI project. It consists of four tightly coupled brokers running ActiveMQ software and it's designed to host in the future other grid operational tools such as SAM.

There are several benefits that messaging has brought to our use cases:

1) *Loosely Coupled Distributed Communication and Reliable Delivery of Messages*: A message broker is the core component of the message-oriented middleware. It receives messages from producers on a channel and routes them to consumers who have registered to receive the messages published on that channel. It also has the ability to transform messages as they pass through the broker. The various implementations of brokers allow them to run as master-slave broker pairs, thus providing reliability in the case of a broker becoming unavailable.

2) *Asynchronous communication*: The broker network will transparently re-synchronize a broker when it re-joins after it is temporarily down. There is a guarantee from the system that messages will not be lost.

3) *Scalability*: In order to ensure scalability a network of brokers is used. A producer sends a message to any broker, which sends it onwards in a store-and-forward fashion to all consumers (who might be connected to other brokers) [20,21].

5. Conclusions

Based on the experience obtained with the previous SAM infrastructure, we have designed a new monitoring architecture that integrates powerful open source components. By selecting commodity software and interfaces that already have a large following in the community, we are providing a robust solution while limiting the amount of support that has to be provided in-house. The original Service Availability Monitoring architecture has been realigned to encompass the organizational changes that took place after the start of the EGI project.

The monitoring solution provided for the WLCG scales from the site-level upwards. It provides the means for sites and ROCs to better monitor their grid services, ultimately contributing to a more available grid infrastructure.

References

[1] M. Lamanna, “The LHC computing grid project at CERN”, Proceedings of the IXth International Workshop on Advanced Computing and Analysis Techniques in Physics Research, pages 1–6, November 2004.

[2] Lyndon Evans and Philip Bryant, “LHC Machine”, Journal of Instrumentation, Volume 3, August 2008.

[3] Foster, Kesselman, Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, Int. J. High Performance Computing Applications, 2001

[4] EGEE, Enabling Grids for E-sciencE, http://www.eu-egee.org, 2005

[5] EGI, The European Grid Infrastructure, http://www.egi.eu/, 2012

[6] OSG, Open Science Grid, A National, Distributed Computing Grid for Data-Intensive Research, http://www.opensciencegrid.org/, 2012

[7] NDGF, The Nordic Datagrid Facility, http://www.ndgf.org/ndgfweb/home.html, 2012.

[8] Nagios, The Industry Standard in IT Infrastructure Monitoring, http://www.nagios.org/, 2012

[9] The EGI SAM Nagios Instances, https://wiki.egi.eu/wiki/SAM_Instances, 2012

[10] GOCDB, The Grid Operations Centre Database, http://goc.egi.eu/, 2012

[11] OIM, The OSG Information Management System, https://oim.grid.iu.edu/oim/home, 2012

[12] BDII, The Berkeley Database Information Index, https://tomtools.cern.ch/confluence/display/IS/BDII, 2012

[13] Apache ActiveMQ, Open Source Messaging and Integration Patterns Server, http://activemq.apache.org/, 2012

[14] Django, Python open source web application framework, https://www.djangoproject.com/, 2012

[15] jQuery, The Write Less, Do More, JavaScript Library, http://jquery.com/, 2012

[16] F Pacini, EGEE User's Guide, WMS Service, DATAMAT, 2005

[17] G Hohpe and B Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003

[18] J Garrett, Ajax: A New Approach to Web Applications, http://mmccauley.net/com601 /material/ajax_garrett.pdf, 2005

[19] D. Crockford, The application/json media type for javascript object notation (JSON), Internet RFC 4627, July 2006.

[20] R Henjes, et. al., Throughput Performance of the ActiveMQ JMS Server, In Kommunikation in Verteilten Systemen (KiVS), Springer Berlin Heidelberg, pp.113-124, Doi: 10.1007/978-3-540-69962-0_10, 2007

[21] J Casey, et. al., A Messaging Infrastructure for WLCG, Journal of Physics: Conference Series, vol. 331/6, pp. 062015, 2011, http://stacks.iop.org/1742-6596/331/i=6/a=062015