Service Availability Monitoring Framework Based On Commodity Software

P. Andrade\textsuperscript{1}, M. Babik\textsuperscript{1}, K. Bhatt\textsuperscript{2}, P. Chand\textsuperscript{2}, D. Collados\textsuperscript{1}, V. Duggal\textsuperscript{2}, P. Fuente\textsuperscript{1}, E. Imamagic\textsuperscript{3}, P. Joshi\textsuperscript{2}, R. Kalmady\textsuperscript{2}, U. Karnani\textsuperscript{2}, V. Kumar\textsuperscript{2}, J. Tarragon\textsuperscript{1}, W. Lapka\textsuperscript{1}, C. Triantafyllidis\textsuperscript{4}

\textsuperscript{1} European Organization for Nuclear Research (CERN), 1211 Geneva 23, Switzerland
\textsuperscript{2} Computer Division, BARC, Trombay, Mumbai 400085 India
\textsuperscript{3} University Computing Centre SRCE, Zagreb, Croatia
\textsuperscript{4} AUTH SCC/GRNET S.A, Aristotle University of Thessaloniki, Thessaloniki GR 541 24

Abstract. The Worldwide LHC Computing Grid (WLCG) infrastructure continuously operates thousands of grid services scattered around hundreds of sites. Participating sites are organized in regions and support several virtual organizations, thus creating a very complex and heterogeneous environment. The Service Availability Monitoring (SAM) framework is responsible for the monitoring of this infrastructure. SAM is a complete monitoring framework for grid services and grid operational tools. Its current implementation tailored for a decentralized operation replaces the old SAM system which is now being decommissioned from production. SAM provides functionality for submission of monitoring probes, gathering of probes results, processing of monitoring data, and retrieval of monitoring data in terms of service status, availability, and reliability. In this paper we present the SAM framework. We motivate the need from moving from the old SAM to a new monitoring infrastructure deployed and managed in a distributed environment and explain how SAM exploits and builds on top of commodity software, such as Nagios and Apache ActiveMQ, to provide a reliable and scalable system. We also present the SAM architecture by highlighting the adopted technologies and how the different SAM components deliver a complete monitoring framework.

1. Introduction

The continuous monitoring of the WLCG infrastructure is a complex task and a large number of monitoring tools are involved in the exercise [1]. At the core of this activity is the Service Availability Monitoring (SAM) framework [2], which hides this complexity and orchestrates the monitoring workflow execution, from the implementation of probes to the presentation of metric results. In addition, SAM brings to the WLCG management, VOs [3], and sites, a powerful and accurate view over the WLCG resources.

In numbers, 236 metrics implemented by different software providers following common guidelines are available in SAM. These metrics are executed several times per hour to probe more than 4,200 service endpoints deployed at 517 sites in 66 countries. The results are then transported via off the shelf messaging technology [4] to a central service store where status, availability, and reliability of grid services and sites are computed. This represents a flow of more than 20,000 results being computed and stored per hour. Finally, results are consumed either programmatically or via different web interfaces tailored for the diverse users of the system.
The status and availability computed by SAM is important to understand the quality of services delivered by sites. It also helps tracking the deployment of new middleware components and identifying operational issues in the infrastructure. In addition, this information is fundamental to develop high-level services for decision-making (like experiments dashboards) or tools with some degree of automated decisions to maintain the operation of Grid services uninterrupted.

2. Description of the 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 project [5]. The concept of high level monitoring emerged 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 500 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. In addition, and as a consequence of emerging different grid operational teams inside EGEE, it was clear that a global solution was needed to probe the different services, raise alarms to sites when 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.

During 2011, the third generation of SAM was released and gradually deployed to production. In order to describe the current architecture of the system, we should first understand the limitations of the previous system and the reasons that motivated the migration to an enhanced framework.

2.1. The Predecessor of SAM and the 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:

(i) **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) [6] project (to enable access to computing resources for European researchers from all fields of science by promoting and enhancing National Grid Initiatives), 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.

(ii) **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.

(iii) **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.

(iv) **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 [7], EGEE and NDGF [8] implied architectural changes, like for instance in the database schema.
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.

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, one of the objectives was to move away from home made solutions running from central locations. Like for several other grid operational tools, SAM followed a decentralization effort in which the system was made distributed. Part of this work consisted in splitting the services to be monitored in several independent instances covering each of them the topology of a whole NGI. At this time, submission of probes was done from 11 regional instances running from CERN that were soon after migrated to the different NGIs. Today, 32 different EGI SAM Instances [9] run continuously from different NGIs and ROCs (Regional Operation Centres), while a central project level database remains for availability calculations.

Apart from the regionalisation effort, the use of standard and open source technologies, where possible and reasonable, is seen as an asset, making training and transferrable skills simpler, and easier to find and hire people having skills in widely used tools. On top of that, embracing open source tools generally brings you the benefit of its community support. In the case of SAM, we tried to move from a CERN homemade system to one based on open source tools like Nagios [18] (a well known open source monitoring platform with a very efficient implementation of a job scheduler) and Apache ActiveMQ (an enterprise-level integration framework based on messaging). By using Messaging technologies, test results coming from Nagios executions are made available both within the regions and at the project level, so that availabilities are computed and delivered to the project management.

2.2.1. Functional areas
Figure 1 presents a high level architecture of the current framework. It is composed of four major open source pillars: Nagios, ActiveMQ, Django and JasperReports [18, 19, 11, 12], that combined together define the functional areas of the system: Data Collection, Data Transport, Notification, Configuration, Filtering/Aggregation, Visualization and Reporting.

The Collection layer, for probes submission and reception of results, is based on Nagios.

The Transport layer is used to integrate the distributed instances of Nagios. This functionality is provided via STOMP protocol routed by Apache ActiveMQ. Since Nagios can both publish and consume metric results to/from messaging, we support both models for collecting metric results.

The Notification layer uses a combination of both Nagios and ActiveMQ. A low-level notification system for site managers and administrators is delivered directly by Nagios. On the other hand, ActiveMQ serves as a notification front-end for third parties, like for instance the OPS Portal [13], the EGI operational dashboard.

The Configuration layer, developed from scratch, is made of three components: SAM Topological and service meta-data aggregation, SAM Profile management system, and Nagios Configuration Generator. The former two components pre-process the topology and metrics
Figure 1. Overview of the SAM architecture.

information, which is then used by NCG to automatically generate the Nagios configuration files.

Filtering and Aggregation is performed by SAM status, availability and reliability computation, which filters, aggregates, and computes statuses and availability of services and sites.

Visualization layer is based on Django (an open web framework supporting modular distributed Web interface and REST APIs).

Finally, JasperReports (an open source reporting engine) delivers Reports detailing monthly availability and reliability numbers for different grid infrastructures and services.

2.2.2. Technology extensions On top of those areas, we added four main extensions for topology and service meta-data aggregation, profile management, availability computation and finally, visualization and reporting. These custom components are detailed in the following sections, highlighting their functionality and the way they glue together the SAM open source pillars.

Topological and service-metadata aggregation

Availability monitoring requires tracking and aggregation of topological and service metadate information from various different infrastructures, e.g., OSG, EGI, WLCG. This includes
information on existing projects, services, sites, tiers, virtual organizations but also service and site contacts, scheduled downtimes, and more. Since this information is related to the way grid middleware operates, it was necessary to establish a custom component that would pre-process the data to make possible the configuration of a Nagios instance.

Using synchronization plugins, SAM aggregates information from multiple sources like for instance: GOCDB [14], OSG Information Management (OIM) [15], EGI Operations Portal, GSTAT [16], VOMS, BDII, plus customized topological sources called VO feeds. An internal relational model aggregates this service meta-data where the core concepts are service and service type. A service is represented by a hostname-port pair, while the service type represents the interface supported by the endpoint. A set of service groups can be defined on top of the core concepts, ranging from simple groups (sites, tiers), up to custom service groups that can be provided by users via XML feeds. Custom service groups help supporting nuances in the different topological information provided by different experiments (site naming conventions, tiering, etc.) Finally, additional information includes site contacts, service downtimes, federations, capacities as well as service to VO mappings.

Profile Management

Profiles are essential for configuring SAM, as they define the coupling between the Nagios probes (metrics) and the grid resources to be tested. Since SAM currently supports more than 200 metrics on more than 4,000 services, providing an easy way to configure each monitoring instance is crucial. The Profile Management system (POEM) defines and groups metrics into profiles and based on those, all the other SAM components get configured. POEM is implemented as a distributed system in which various different instances are distinguished by the concept of namespaces. A namespace uniquely identifies a set of profiles that can be configured via a dedicated web interface. Each SAM instance can synchronize profiles from many different namespaces and thus offers a wide range of possibilities to configure a SAM instance. The union of all synchronized profiles in a particular SAM instance determines how services are coupled with metrics to constitute a working Nagios configuration that can be generated on demand.

Status and Availability computation

Status and availability computation, shown in Figure 2, aggregates and filters results published by Nagios and transported by ActiveMQ. On the central instance of SAM around half a million metric results are processed daily. The service status represents a discrete value for a set of metric results computed for each service type (interface), e.g. the status of a given SRMv2 service is determined by aggregating several SRMv2 metrics.

Status computation is performed over a buffer of metric results in regular intervals. In each step, metric results are aggregated and an overall status for a given service is computed. The intermittent state is saved to a historical table, which is then processed to compute service availability. The current delay between the actual Nagios measurement and the final status computation is just 15 minutes, which is thanks to the use of a reliable messaging infrastructure and a fast status computation implemented in database stored procedures. In general, status computation delivers the key missing functionality in the open source stack that can aggregate, compute and store status of services coming from multiple Nagios probes.

Availability and reliability computation is processing the service status information to compute the fraction of time the service was up (in OK state) during the period the service was known. Unlike availability, reliability does not consider scheduled downtime in the known period of a service. Once availabilities of all services instances are determined, we combine them to compute the overall site availability. This is based on applying a custom aggregation formula to a set of service statuses. Service statuses are usually grouped by the interface they support (service type, e.g. SRMv2, CREAM, etc.).

The following are sample aggregation formulas:
Availability in this case will consider any SRMv2 endpoint (logical OR) and either CE or CREAM-CE endpoints for a given site. Availability for a site would therefore require an OK status in at least one of its SRMv2 endpoints as well as OK status in either CE or CREAM-CE endpoints (logical AND). It is also possible to customize how the status is aggregated for a given service type, e.g. (and SRMv2)

In this case status of all SRMv2 endpoints will be considered in the availability computation. For a detailed technical description of the availability computation algorithm please refer to [17].

Visualization and Reporting
Visualization and reporting constitute the main outputs of the SAM framework. Apart from the standard Nagios web interface, SAM has a set of applications written in Django [11] web framework that offer different views on service availability and service status. In addition, management dashboard has been added to display availability trends and experiment usage statistics together with WLCG data transfers showing overall incoming and outgoing FTS transfers. A set of available views is shown in Figure 3.

Reporting is another essential part delivering monthly reports to all relevant projects. It is based on the Jasper reporting engine [12], which is enhanced by custom templates and profiles to generate various different reports.

3. Related Work
There are currently several existing enterprise level Nagios modifications or monitoring alternatives including: Icinga\(^1\), Zenoss\(^2\), Zabbix\(^3\), Shinken\(^4\), New Relic\(^5\), Librato Metrics\(^6\), Splunk\(^7\), and more. Unlike SAM, they offer limited support for running a world-wide monitoring

---

\(^1\) https://www.icinga.org/
\(^2\) http://www.shinken-monitoring.org/
\(^3\) http://newrelic.com/
\(^4\) https://metrics.librato.com/
\(^5\) http://www.splunk.com/
platform capable of mixed model testing of the complex grid middleware. At the same time, SAM offers many features available in these alternatives such as database storage, advanced notifications, complex user interfaces, web APIs or automated generation of monitoring configurations.

In terms of existing commodity software that can be used, and in particular architectural layers, there is a wide range of existing solutions and new open-source projects that are rapidly gaining ground. The collection layer has several other alternatives to Nagios including Collectd and Ganglia. The transport layer has besides STOMP, a large set of existing products such as XMPP, Smiple, SMPT, 0MQ, APMQ, rsyslog or IRC. Besides existing relational-based storages, there is also RRD, Graphite, OpenTSDB as well as many other non-relational storages like HBase, Cassandra, MongoDB, etc. The filtering/aggregation contains many log processing libraries such as logstash, logster, statsd, Graylog2 but also more complex frameworks such as Esper, cepmon and Riemann. For notifications PagerDuty or Response could be considered. Reporting and visualization is mostly dominated by Graphite and Jasper or ReportLab, but many custom interfaces can be easily implemented via web frameworks. In configuration cucumber-nagios or monitors.txt are both very interesting projects.

4. Summary
SAM is currently the de facto tool for availability and reliability monitoring for WLCG and EGI infrastructures. Its distributed model is currently deployed in 40 regions on 4 continents. The integration of open-source tools has helped reducing its maintenance support while bringing customized availability/reliability computations, custom metrics, Web2.0 visualizations, REST APIs and automated reporting. Currently there are more than 200 metrics monitoring more than 4,200 endpoints, what represents around 36,000 service instances monitored round-the-clock. The use of SAM has helped identify many operational issues and improve the overall stability and quality of services delivered by the various infrastructures to the WLCG. To certain extent it also contributed to the development of high-level services that can effectively handle the resource scheduling and data management in large Grids.
References

[1] Lamanna M 2004 The LHC Computing Grid Project at CERN (Proceedings of the IXth International Workshop on Advanced Computing and Analysis Techniques in Physics Research, pages 16, November 2004)

[2] Andrade P, Babik M, Bhatt K, Chand P, Collados D, Duggal V, Fuente P, Hayashi S, Imamagic E, Joshi P, Kalmady R, Karnani U, Kumar V, Lapka W, Quick R, Teige S, Tarragon J and Triantafyllidis C 2012 Distributed Monitoring Infrastructure for Worldwide LHC Computing Grid (Computing in High Energy and Nuclear Physics 2012, New York)

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

[4] L Cons and M Paladin 2012 The WLCG Messaging Service and its Future (Computing in High Energy and Nuclear Physics 2012, New York)

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

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

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

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

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

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

[11] Holovaty A and Kaplan-Moss J. 2009 The Definitive Guide to Django: Web Development Done Right (Apress Series, ISBN 9781430219361)

[12] Danciu T and Chirita L 2007 The Definitive Guide to JasperReports (Apress Series, ISBN 1590599276)

[13] EGI Operations Portal, http://operations-portal.egi.eu/

[14] GOCDB, Grid Operations Center Database, http://goc.egi.eu/

[15] OIM, OSG Information Management, http://oim.grid.iu.edu/oim/home

[16] GSTAT, http://gstat2.grid.sinica.edu.tw/

[17] Kalmady R, Chand P, Kumar V, Duggal V, Bhatt K, Sonvane D 2011 Computation of Service Availability Metrics in ACE (http://goo.gl/rnvAW)

[18] Barth W 2008 Nagios: System and Network Monitoring (No Starch Press, Second Edition edition, 2008, ISBN 978-1593271794)

[19] Snyder B, Bosanac D and Davies R 2011 ActiveMQ in Action (March, 2011, ISBN 1933989940)

[20] Kalmady R, Sonvane D, Chand P, Bhatt K, Kumar V, Nyczuk P and Sekera Z 2009 Monitoring the Availability of Grid Services Using SAM and Gridview (Grid Computing Part IV, 163-168, DOI: 10.1007/978-0-387-78417-5)

[21] Badger M 2008 Zenoss Core Network and System Monitoring (Packt Publishing)

[22] Olups R 2010 Zabbix 1.8 Network Monitoring (Packt Publishing: From technologies to solutions Series)