The event notification and alarm system for the Open Science Grid operations center

S Hayashi, S Teige and R Quick
Indiana University
University Information Technology Services
E-mail: hayashis@iu.edu steige@iu.edu rquick@iu.edu

Abstract.
The Open Science Grid Operations (OSG) Team operates a distributed set of services and tools that enable the utilization of the OSG by several HEP projects. Without these services users of the OSG would not be able to run jobs, locate resources, obtain information about the status of systems or generally use the OSG. For this reason these services must be highly available. This paper describes the automated monitoring and notification systems used to diagnose and report problems. Described here are the means used by OSG Operations to monitor systems such as physical facilities, network operations, server health, service availability and software error events. Once detected, an error condition generates a message sent to, for example, Email, SMS, Twitter, an Instant Message Server, etc. The mechanism being developed to integrate these monitoring systems into a prioritized and configurable alarming system is emphasized.

1. Overview of the Grid Operations Center
The Grid Operations Center (GOC) has primary responsibility for the operation of services that enable Open Science Grid [1] users to exploit the resources available there. It comprises 64 Virtual Organizations (VO) representing a variety of research disciplines including High Energy Physics. The largest VOs are engaged in HEP research and, in fact, form a majority of OSG users.

The GOC is based at Indiana University and is tasked with supporting the operational needs of users, resource providers, and collaborators of the OSG. It consists of human and automated services provided to maximize operational functionality of the OSG. It is the first point of contact for members of the OSG needing assistance and tracks those problems through their resolution. The GOC also provides numerous services to the OSG community that support usage of the Open Science Grid. The GOC does not provide computational, storage or data transfer services. Discussed below is an overview of the services provided by the GOC.

2. Overview of the services
There are many ways to categorize the services supported. They are described here arranged by how they interact with each other and with the people using them. Services can be described as fully automated, that is, a service where a computer interacts directly with another computer; semi-automated, where a computer provides information to or accepts information from a person; or human scale, where the service provides a mechanism facilitating interpersonal
communication. Additionally, there are services that facilitate the operation of other services but are not directly exposed to machines or people outside the GOC. These are referred to below as auxiliary services.

2.1. Fully automated services
The fully automated services used by the OSG include the Berkeley Database Information Index (BDII), the Resource and Service Validator collector (RSV) and Ticket Exchange (TX). BDII is the central information source describing the current state of the OSG. It provides, for example, information required for job scheduling. RSV collects data reported by OSG resources regarding the status of those resources. This information is processed to give a visual representation of the state of the OSG and for accounting purposes. Ticket exchange [2] is a custom written system to allow different trouble ticket systems to seamlessly inter-operate. BDII events occur at a rate 700k/day and RSV events at 55k/day. There are a few ticket exchanges in a typical day.

2.2. Semi-automated services
Semi-automated services include the Gratia collector, GOC monitor and notification filters. The Gratia collector gathers job and data transfer information used for accounting and display purposes. Monitor is the central destination for status information reported by GOC services. It produces alerts and alarms when problems are detected. Notification filters interact with monitor to determine what information is passed to people and by what mechanism. If a sufficiently critical service is determined to be in a sufficiently degraded state, for example, a text message is sent. Unfiltered GOC monitor events are generated at a rate of approximately 100 per day.

2.3. Interpersonal services
Interpersonal or communication services include TWiki [3], Blog spot, Footprints [4] ticket system, and Repo. TWiki is the primary collaboration tool for OSG staff and users, Blog spot is a central service collecting posts from several blogs associated with the OSG and, the ticket system is used to report and track problems. The Repo service is used to distribute software.

2.4. Auxiliary services
These services are required to accomplish the purposes of other services. They include registration systems, accounting processors, database servers, monitoring systems, etc.

3. Sources of events
All of the services discussed above are, at any given instant, in some defined state. That state can change in response to an outside influence; for example, when a data transfer completes a notification is sent to the Gratia collector. It can also change due to an internal change such as a software error condition or a hardware failure. These changes of state are referred to as “events” below.

Events can be characterized as follows:
- Status events are automatically generated periodic reports of a resource status
- User events occur when a person interacts with a service. Additionally, social media can generate user events, for example, when a post is made to a chat room or a blog
- Accounting events occur at the completion of a computing task
- Rootmail events are generated by the operating system when certain conditions exist. These conditions include system errors and errors generated by processes run as the root user
Service health events occur when automated diagnostic software detects a problem with a service, the data associated with a service, or the hardware hosting the service. The diagnostic software is custom written as appropriate for an individual service. Detection of hardware problems is accomplished with Munin [5] and Dell OpenManage [6].

4. Event processing

An event can signal a state of the services that requires action, it can be an ordinary update requiring no action, or it can be simply informative.

Given the large variety of event sources and responses it is desirable to have a centralized mechanism for collecting and processing events. The system to be implemented will be based on the RabbitMQ messaging system. All event sources will format the text associated with their events and pass that to the message queue (MQ). A client will be created that listens to the message queue and passes along any events its filtering algorithms determine require action of some sort or are of potential interest to human users. Events occur at widely variable rates, table 1 gives examples. Events generated at the highest rate are rarely of interest to people and demonstrate the need for a filtering mechanism between the MQ and a human user. Figure 1 gives an overview of event flow and processing for a few possible use cases.

RabbitMQ [7] is based on the Advanced Message Queuing Protocol [8], AMQP. AMQP allows data persistence to be defined for each queue. For example, for critical queues, the system can be configured to store data on disk, or for high-volume, low-criticality queues, messages can be kept in memory for higher throughput. AMQP allows clustering to provide higher throughput and availability.

While other implementations using AMQP exist (Qpid and ActiveMQ for example), we have encountered reliability problems with ActiveMQ in another project which led us to reject that option. We have chosen RabbitMQ because of our positive experience with community support and the reliability observed to date.

| Event Type         | Rate (per day) |
|--------------------|----------------|
| Data transfer      | 1,500,00       |
| BDII update        | 700,000        |
| User job ends      | 620,000        |
| RSV report         | 55,000         |
| RSV status change  | 300            |
| Monitor Alert      | 100            |
| Munin Alarm        | 20             |
| User ticket        | 7              |

4.1. Event processing examples

Following are several examples of how the event processor will function.

An OSG service generates an event signaling a problem with the service and this event is passed to the MQ. The message identifies the service, the nature of the problem, the time the problem occurred, and possibly the severity of the problem. The event processor determines the criticality of the service (as a lookup value derived from the service identity) and the severity of the problem (either directly from the message or the type of the message). In this example, the
custom software diagnostic system associated with the service and any Munin generated events will be tagged as “Critical” or “Warning”. OpenManage events are always considered “Critical” and rootmail events as “Warning” severity events. If it is determined that the service is of the highest priority and the event is of critical severity, the event processor generates a text message (24 x 7 x 365) to a configurable list of people responsible for responding to the situation. If the service is of a priority other than the highest or the event is of warning level, an e-mail is sent.
4.2. Features

The system described provides several advantages. There is a central, standardized destination for all events associated with the operation of the OSG: the message queue. There is a single source from which to obtain change of state information and a standardized means to access it. These properties allow the complex tasks described above to be simply realized. Standardization of the message queue allows the source and listener components to be reused, easily minimizing the effort associated with this realization.

For example, the generation of a text message will be done with a standard message listener and a configurable filter. The listener will pass all messages to the filter, which will then ignore any message not associated with a high priority service. If a message associated with a high priority service arrives, the filter determines if it is a status message indicating a problem of sufficient severity to warrant the generation of an alarm. The filter then consults a calendar and the clock to determine if the event has occurred outside of working hours, and if it has, it sends a text message to a configurable list of recipients.

5. Conclusions

A simple, general architecture for communicating the state of a complex system is described. Communication from machine to machine, machine to person, person to machine and person to person is supported. A wide variety of communication endpoints can be easily supported by the reuse of standardized components.

References

[1] http://www.opensciencegrid.org
[2] Hayashi S, Gopu A and Quick R 2011 GOC-TX: A Reliable Ticket Synchronization Application for the Open Science Grid J. Phys.: Conf. Ser. 331 082013 doi:10.1088/1742-6596/331/8/082013
[3] http://twiki.org/
[4] http://www.numarasoftware.com/footprints/
[5] http://munin-monitoring.org/
[6] http://en.community.dell.com/techcenter/systems-management/w/wiki/openmanage-systems-management.aspx
[7] http://www.rabbitmq.com/
[8] http://www.amqp.org/