Understanding WLCG Monitoring

An introduction to WLCG monitoring for grid service developers

Overview

The aim of this document is to introduce the WLCG monitoring framework to grid service developers, who, up to this point, may not have considered the monitoring of their services, and its implications.

Continuous improvement in the reliability of the WLCG service requires effective monitoring of the underlying grid services deployed at a site. WLCG monitoring is predicated upon the fact that site administrators will use a local fabric monitoring system to find, diagnose and correct problems with services running on their site. This is a complement to central operations support structures, which may also exist.

After reading this document, a Grid Service developer should

- understand the terms used to describe monitoring in the WLCG context
- understand how to document their service in terms of metrics that should be gathered

Local and Remote Monitoring – a birds eye view

Examples of **local** (site) fabric monitoring systems that are used currently within WLCG are LEMON[1], Nagios[2] and Ganglia[3]. In general such monitoring systems run directly on the fabric supporting the services being monitored.

Another aspect of monitoring is **remote** monitoring. This is where the site is "probed" from outside by an external entity, e.g. the WLCG Service Availability Monitoring (SAM)[4] system, mimicking operations normally performed by users of the system. To reduce the time between a problem being recognized by a remote monitoring system and its resolution WLCG believes it is important that information gathered by remote monitoring systems must be made available to

[1] http://lemon.web.cern.ch/lemon/index.shtml
[2] http://www.nagios.org/
[3] http://ganglia.sourceforge.net/
[4] https://lcg-sam.cern.ch:8443/sam/sam.py

site administrators via a programmatic interface (e.g. XML/HTTP) for integration into their local fabric monitoring system.

Status and Performance Monitoring

Service monitoring consists of gathering time stamped **samples** of various **metrics** from the service. We consider two types of metrics: Service Status and Performance. The reason for this split is that different fabric monitoring systems may handle one or the other category (e.g. Nagios for Service status and Ganglia for performance monitoring).

Monitoring of service **status** consists of checking if the service is running, i.e. available to accept user interaction in a normal method. A status metric takes on one of a small set of enumerated values signifying the current state of some agreed aspect of the service e.g. “Available”, “Degraded” or “Unavailable”.

Monitoring of service **performance** consists of gathering metrics useful in diagnosing the behavior of a service, e.g. Load, #threads in use, #jobs/transactions processed per minute.

Metric Categorization

We have identified three categories of metrics that can be gathered. The categorization is based on the type of information that the metric represents. Each category has some differences also in how it is collected, which has implications for the service developer. Since it is hard to assign names to the categories, we simply call them **Category A, B, C**, with a descriptive name associated with each.

Category A (“Traces in the fabric”)

There is a lot of information about the service that can be gathered by directly examining the supporting fabric using standard operating system level tools. It is possible to build both availability and performance metrics for a service from this information. Also, note that these metrics usually need a local **agent** running on the node in order to be collected.

Examples of such information include:

1. Is a given process running
2. What processes are attached to what network port
3. Availability of space in a file system
4. Existence and permissions on a given file (e.g. a log file)

5. Appearance of a string/regular expression in a given file
6. Load on the machine

From this information, relevant metrics could, for example, be:

1. # of processes called ‘tomcat4’ running under the user id ‘tomcat’
2. the process ‘tomcat4’ should be listening on port 8443
3. /var should have at least 5% free
4. The directory /var/log/tomcat5 should exist, and be writable by the user ‘tomcat’ and group ‘tomcat’
5. The number of occurrences of the string ‘ORA-‘ in the last 5 minutes in the file /var/log/tomcat5/catalina.out
6. Average load on the node in the last five minutes

Category B (“Interpretation of a user-level operation”)

This category of metrics is gathered by running an operation a user might want to do, and parsing the output to interrogate the status of the service. Examples are:

1. “ping”-type operations which check if the service is listening on a port correctly
2. ‘lfc-ls /grid’ on an LFC Catalog to check results are being returned from a database correctly
3. Submitting an FTS transfer job, and monitoring its progress and termination state.

It is important to note that such metrics can be gathered locally on the node, from another node on the same site, or from a node remote to the site.

Category C (“Internal metrics presented by the service”)

Some services provide a mechanism to obtain more detailed information about the internal state of the service. In general gathering this information does not also perform a function useful to a normal service user. This could be via an API call in its public API, an appropriate HTTP request to a web service or via an accounting file stored locally on the file system. Examples of possible metrics in this category are:

1. An LFC service can report, how many of it’s threads are currently serving requests
2. An R-GMA MonBox provides a HTTP request an XML document, which contains the status of all producers and consumers currently connected to it

3. A CE provides, in a file, statistics on how many jobs have terminated successfully in the last 5 minutes.