JENNIFER M. SCHOPF AND BEN CLIFFORD

Monitoring Clusters and Grids

One of the first questions anyone asks when setting up a cluster or a Grid is, "How is it running?" This inquiry is usually followed by the questions, "Why isn't it running any faster?", "What's going on over there?", and "Is that broken or just really slow?"

Monitoring systems, in the broadest sense, are tools that report some set of measurements to higher-level services. Users can employ these tools in various ways. For example, large file transfer time information is useful to application scientists (asking "How long will my file transfer take?"), system administrators (asking "Is the file transfer server running properly?"), and network administrators (asking "How big is the pipe from the user's point of view?"). Only network administrators, on the other hand, may need packet loss information.

In this article we focus on the comparison of several common monitoring tools used to disseminate information about the performance of clusters and Grids, and the features to consider when selecting one for your environment.

What Is a Monitoring System?

All monitoring systems have three major components: information collectors (sensors or probes), support services (collection, archiving, management), and interfaces (GUIs or APIs). These components are often implemented in different ways, sometimes subdividing or sometimes combining the basic pieces, depending on the goals of the individual monitoring system.

Information collectors are general lightweight probes, benchmarks, or daemons that interface to a node in a cluster or some resource in a Grid to gather up a specific measurements or pieces of data. These can be as simple as a script that gets information from `top` every 10 minutes to a full benchmark for the cluster, or a set of data transfer tests.

Support services, the second component of any monitoring framework, provide the basic functionality to collect service data, aggregate that data in a meaningful way, archive the data, and allow both simple and complex queries of the data. These necessary pieces give us the basic infrastructure needed for any monitoring service.

Interfaces are the third major component of a monitoring framework. They enable users to access basic data. These interfaces can be simple commands that report back a set of values or elaborate GUIs with maps and changing toolbars to report the data.

Internal to a monitoring system, a *schema* must be defined to express the data collected. With a well-defined schema, two very different monitoring implementations can share a mapping of data. For example, using the GLUE schema (see the sidebar), LDAP-based MDS2, OGSI-based MDS3, and R-GMA, all of which use a relational database backend, have a common data mapping. Having a common schema allows different implementations to understand one another's data sources.

Scalability is another important factor to consider. Monitoring systems must be able to scale with respect to a number of factors — the number of data sources being collected, the size of the data, the number of resources being contacted for these data sources, the length of time the data is stored, the number of clients that are going to access the data and so forth.

TABLE ONE

Cluster and Grid Monitoring Systems

	CLUSTER	GRID
Network connectivity between nodes	Dedicated high-performance LAN (e.g. Gigabit Ethernet)	WAN with high-variability bandwidth and latency behavior
Homogeneity	Nodes have only small variations between them	Every node can be very different
Resources	Nodes in a cluster, local scheduling information, LAN performance	Full cluster data, instruments, data stores, wide-area networks, etc.
Administrative control	Single administrative domain	Many administrative domains

Table One shows some of the major differences between cluster and Grid monitoring systems. Grid systems may have more and different types of data being collected than do most standard cluster installations. However, less low-level data is collected because one of the guiding principles of a Grid monitoring system is to publish only data of interest to multiple users at different administrative domains, in part because security and privacy concerns.

When evaluating what monitoring system is right for your environment, you should consider what information you want to gather - if you're primarily interested in network performance between your clusters, a tool that doesn't have this data won't help your users no matter how nice the GUI is. You should also consider the set of basic support services offered and any scalability numbers available for them. And don't forget the interface: without the right kind of interface, no matter how much data you have, it won't be useful, although most tools offer a way to extend basic APIs and GUIs in a straightforward way.

In the remainder of this article we review three common cluster monitoring tools, Clumon, ServiceGuard, and Ganglia, as well as four Grid monitoring tools, Inca, the Globus Toolkit MDS, R-GMA, and Hawkeye.

Clumon

The Clumon performance monitoring system was developed for monitoring Linux-based clusters at the National Center for Supercomputing Applications (NCSA) and to give an overview of the current state of the cluster at a glance. It displays both scheduler information and data about the hosts themselves in a combined format. The system was designed to scale to potentially thousands of hosts and, at the same time, to avoid creating too large a load on the hosts from which data was being collected or the network layer by having only a passive set of software running on the hosts.

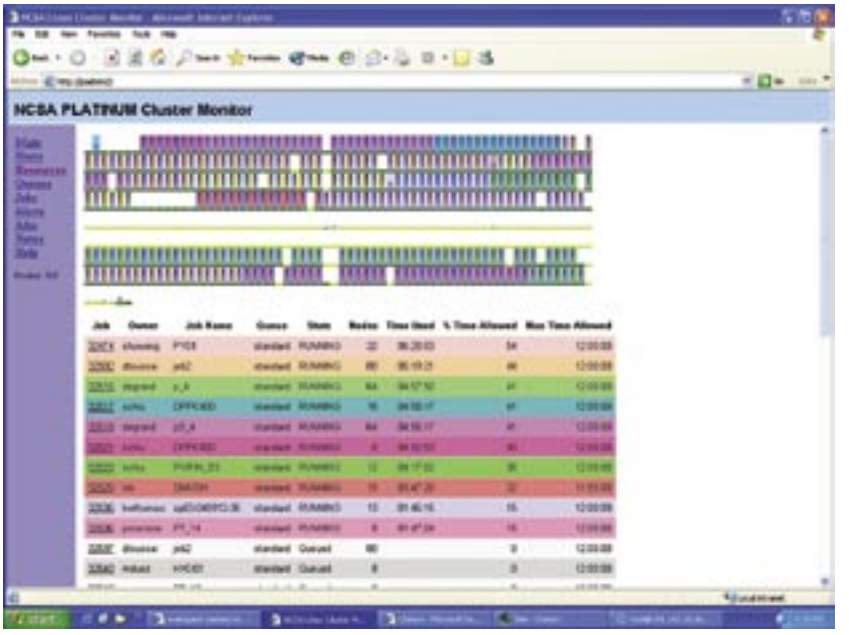

FIGURE ONE Screen shot from Clumon

Clumon's information sources are the Performance Co-Pilot (PCP) by SGI and the PBS scheduler. PCP collects a wide variety of operating system data and hardware data, including host counts for available, in use, and running process data and other low-level details. Clumon interfaces to the PBS scheduler to retrieve information about queue lengths and running jobs on the system, as well as advance reservation data. Additional information can be added to the system by writing a PCP module to collect it.

Data from hosts running Clumon are reported back to a central collector, and system administrators can easily adjust the frequency and amount of data collected, as well as adding or removing items from the set of collected data. Changes take effect without having to communicate with the hosts and even without having to restart any service on the monitoring machine. Data are archived in a MySQL database.

Clumon offers a wide variety of Web-based interfaces to access the data, but not APIs or command-line interfaces. Web-based pages showing detailed information about the resources, queues, jobs, or warnings are available, as well as a visualization of the process tree for processes on the system. The most commonly used view, though, shown in *Figure One*, is the macro view, which shows the overall status of the cluster.

HP ServiceGuard

A number of commercial monitoring systems are available, which are often integrated into cluster management products to support high availability (such as node failover). One example, HP ServiceGuard, provides monitoring targeted at supporting high availability of applications. Since this is a more focused goal than that of open source cluster monitoring products, the system has quite a different feel.

Running on each node is a cluster manager daemon, one of which is selected as the cluster coordinator. Each node in the cluster transmits heartbeat messages over the network

using TCP or over an RS232 cable to the cluster coordinator. In addition to this basic heartbeat detection, add-on modules allow monitoring of other data, such as disk status, system load, and LAN status.

User interfaces to ServiceGuard include command-line tools that provide individual node summary status, as well as delivery of notification messages to a cluster administrator. In addition, an archive is kept of minor events that do not warrant immediate notification; this archive is used to perform predictive monitoring in order to decide that a failure is likely to occur in the immediate future.

Ganglia

Ganglia is a cluster monitoring system originally developed by the UC Berkeley Millennium Project. Ganglia has a two-tier structure with an intracluster layer and a intercluster layer. The intracluster layer has a daemon that runs on each node in the cluster and is responsible for gathering local information and transmitting it across the network. Ganglia uses multicast to distribute results across the cluster network, something that is possible in a cluster environment with a dedicated LAN. The intercluster layer allows groups of clusters to be federated into hierarchies so that a single view can be seen of a set of clusters, thereby edging into functionality seen by Grid monitoring systems. Instead of using multicast for the intercluster communication protocol, TCP is used to convey results.

A typical installation on a cluster allows Ganglia to collect metrics such as load data, memory use statistics, network traffic, basic operating system information, disk usage, and uptime. Ganglia provides raw data output in a simple XML format containing name/scalar value pairs associated with each node. An installation can easily be extended to measure and distribute other metrics, using simple command-line tools.

One of the best features of this monitoring system is its extensive graphing capabilities, shown in *Figure Two*. A Web-based interface allows both current and archival data to be displayed for a whole cluster and for individual nodes.

Inca

The Inca test harness and reporting framework is a Grid monitoring system originally developed for the TeraGrid project to verify its software stack. Inca can now perform automated verification of service-level agreements by checking software versions, running basic software and service tests, and running performance benchmarks.

The information collection subsystem of Inca consists of a set of reporters, currently with more than 100 tests deployed on the TeraGrid production resources, primarily gathering information about specific software versions and functionality tests. In addition, Inca gathers data about Grid service capabilities, including checking on the Globus Tookit job submission service (GRAM) and GridFTP, OpenSSH, and MyProxy.

The support services deployed as a part of Inca consist of a distributed controller, running on each client resource that controls the local data collection through the reporters, and a centralized controller where system administrators can change data collection rates and deployment of the reporters. An archive system, the depot, collects all the reporter data using a round-robin database scheme.

Available interfaces include command line, C, and Perl APIs, as well as several GUI clients. The most commonly used GUI, shown in *Figure Three*, gives detailed information about sites and their deployed software and services. Clicking on an error box gives additional information about the fault found by the reporter.

Globus Toolkit Monitoring and Discovery System (MDS)

The Globus Monitoring and Discovery System, MDS, is a Grid-level monitor

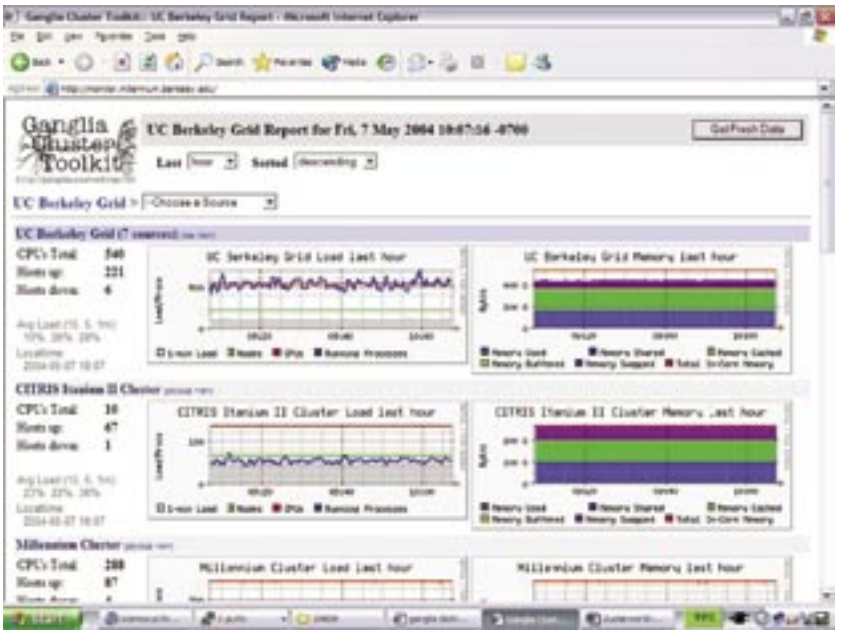

FIGURE TWO Screen shot from Ganglia

used primarily for resource discovery and resource management selection decisions. Two versions are in current use: MDS2, which is based on LDAP, and MDS3, an OGSI-compatible Web service implementation.

Both MDS2 and MDS3 provide a wide range of information about basic resources and queues and interface to the cluster monitoring system Ganglia for clusterwide data. In addition, every OGSI-compatible service provides a range of monitoring information about itself, thereby allowing MDS3 to make use of this data. MDS2 uses an LDAP-based schema, while MDS3 uses XML to represent monitoring data. Both versions use the GLUE schema for default cluster and host information.

MDS2 and MDS3 both have an index service where data is collected and cached. Index service installations can be arranged in hierarchies to provide scalability and aggregate views over a set of clusters. It is necessary to manually configure the initial relationships between indexes and resources; but once this configuration is done, clients contacting the index can see information on every indexed resource. MDS3 also has an associated archiver that stores data in a Xindice XML-based database.

FIGURE THREE Screen shot from Inca

MDS2 provides only a very basic PHP-based Web client, although extensive command line, C, Fortran, and Java APIs are available through the LDAP infrastructure. MDS3 has a richer servlet-based Web user interface, as well as command line and APIs for several languages using the OGSI standard.

Relational Grid Monitoring Architecture

The Relational Grid Monitoring Architecture (R-GMA) monitoring system is an implementation of the Grid Monitoring Architecture (GMA) defined within the Global Grid Forum (GGF). It is based on the relational data model and Java servlet technologies. Its main use is the notification of events — that is, a user can subscribe to a flow of data with specific properties directly from a data source. For example, a user can subscribe to a load-data data stream and create a new producer/consumer pairing to allow notification when the load reaches some maximum or minimum.

Currently, a standard R-GMA deployment includes information gathering by interfacing to the MDS information providers, including basic host data, queue information, and some network data. R-GMA uses the SQL mapping of the GLUE schema, and users can add additional information providers by simply defining table entries and collectors.

The infrastructure of R-GMA consists of a registry, or directory service, and a set of producers and

The GLUE Schema

One of the ongoing problems in getting different monitoring tools to interact is the lack of a common schema for cluster and Grid resource and service information. The Grid Level Uniform Environment (GLUE) project was formed by a group of U.S. and EU physicists to define a set of interoperable tools to allow their different infrastructures work together, one of which was a set of monitoring schema for cluster and Grid data.

Three sets of schemas have been defined: those for compute/host data, storage data, and network data. The compute data was written to allow for both queued (cluster) systems and single host resources and defines representations for identity information, scheduler information (contact point, policy, current use statistics), and host data (architecture, operating system, memory configuration, basic benchmark data). The storage schema details identify information, server load data, policy data, available protocols, and basic usage data. The network schema is still in development as part of the GGF Network Measurement Working Group, *www-didc.lbl.gov/NMWG*.

Along with a data mapping independent UML description, every schema has been mapped to LDAP, XML and SQL implementations. More information can be found at *www.globus.org/mds/glueschema.html*.

producer servlets that advertise tables and definitions of the data held in the RDBMS in the Registry. The RDBMS holds the information for all the producers, namely, the registered table name, the predicate, and some internal information. Users or other agents can contact a consumer servlet that issues SQL queries against a set of supported tables, to find suitable producers and then contact the producers directly to query for the data. Archiving is available as a data consumer for any subscribed data stream, RDBMS storage, and then the producer interface for any of the stored data.

A number of tools are available to query R-GMA producers. There is a command line tool, a Java graphical display tool, and the R-GMA browser. The browser is accessible from a Web browser without any R-GMA installation and offers a set of custom queries.

Hawkeye

Hawkeye is a monitoring tool developed by the Condor group and designed to automate problem detection, for example to identify high CPU load, high network traffic, or resource failure within a distributed system. Its underlying infrastructure builds on the Condor and ClassAd technologies. Hawkeye is also used for easier software maintenance for a set of machines running the Condor resource management software.

Information collectors in Hawkeye publish their data as ClassAds — collections of attribute/value pairs (e.g., "operating system" and "Linux") that can be broadcast for any resource by the local Condor daemon. There is no set schema, nor is there a required or expected set of data to be advertised by the local system. A resource owner can set up a ClassAd to broadcast any data they select.

Hawkeye's infrastructure uses ClassAd Matchmaking to issue

> Monitoring systems, in the broadest sense, are tools that report some set of measurements to higher-level services

warnings and perform other actions. A client submits a Trigger ClassAd, which specifies an event and a job to execute if the event occurs, and this is matched against the set of resource ClassAds. For example, a Trigger ClassAd can specify an event in which the CPU load is greater than 50 and a job that will kill a Netscape client running on the matched machine; if any machine advertises a resource ClassAd with a CPU load value of greater than 50, the daemon on that resource will kill that machine's Netscape process. Information about the resources for evaluation is collected at fixed intervals, but can also be queried directly.

Hawkeye's main interface is a Web based form that lists the set of triggers set up for a testbed and their status, along with data about the resources being queried, and summary views of resources with acknowledged problems.

Jennifer M. Schopf is a researcher at Argonne National Laboratory and part of the Globus Alliance, with a focus on monitoring and performance. She can be reached at jms@mcs.anl.gov.

Ben Clifford is a programmer at USC/Information Sciences Institute, working on monitoring and discovery components in the Globus Toolkit. He can be reached at benc@isi.edu.

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38 with the University of Chicago, by the National Science Foundation, and by IBM.

Resources

Clumon
- clumon.ncsa.uiuc.edu

HP ServiceGuard
- docs.hp.com/hpux/ha/index.html

Ganglia
- ganglia.sourceforge.net

Inca
- tech.teragrid.org/inca

Globus Toolkit Monitoring and Discovery System (MDS)
- www.globus.org

Relational Grid Monitoring Architecture (R-GMA)
- www.r-gma.org

Hawkeye
- www.cs.wisc.edu/condor/hawkeye

Global Grid Forum definition of the Grid Monitoring Architecture
- www.ggf.org/documents/GFD/GFD-1.7.pdf

Scalability comparison of MDS2, Hawkeye, and R-GMA
- www-unix.mcs.anl.gov/~schopf/Pubs/xuehaijeff-hpdc2003.pdf

Open Grid Services Infrastructure Specification
- xml.coverpages.org/OGSI-SpecificationV110.pdf

Condor Matchmaking
- www.cs.wisc.edu/condor/manual/v6.2/2_3Condor_Matchmaking.html

A review of other Grid and cluster monitoring systems
- www.lpds.sztaki.hu/~zsnemeth/apart/repository/gridtools.pdf