GridRM: A Resource Monitoring Architecture for the Grid

Mark Baker and Garry Smith

The Distributed Systems Group,
University of Portsmouth, Portsmouth UK.
{mark.baker, garry.smith} @computer.org

1 Abstract

Monitoring resources is an important aspect of the overall efficient usage and control of any distributed system. The resources of interest can include all manner of networked devices, from a remote sensor or satellite feed through to a computational node or a communications link. In this paper, we describe a generic open-source resource monitoring architecture that has been specifically designed for the Grid. The paper consists of three main sections. In the first section, we outline our motivation and briefly detail similar work in the area. In the second section, we describe the general monitoring architecture and its components. In the final section of the paper, we summarise the experiences so far and outline our future work.

2 Introduction

A wide-area distributed system such as a Grid requires that a broad range of data be monitored and collected for a variety of tasks such as fault detection and performance monitoring, analysis, prediction and tuning. A range of tools has been developed for monitoring distributed resources (see section 4).

In this paper, we present and describe a generic open-source resource monitoring architecture that has been specifically designed for the Grid. The system we describe here, called GridRM, is based on the Global Grid Forum's [1] Grid Monitoring Architecture [2], Java technologies (applets, servlets and JDBC) a SQL database and SNMP (agents and objects). Unlike many other monitoring systems, GridRM is designed to monitor Grid resources, rather than the applications that execute on a Grid. The GridRM is capable of registering interest in events, remotely observing devices, as well as gathering and displaying monitoring data.

3 The Grid Monitoring Architecture

The Global Grid Forum [1], Grid Monitoring Architecture Working Group [11] has defined a Grid Monitoring Architecture (GMA) that allows Grid monitoring tools to interoperate. The GMA is a producer/consumer-based architecture that is driven by a standard and extensible set of events. Figure 1 shows the basic GMA architecture.

The GMA uses directory services to support information publication and discovery between consumers and information providers. The GMA protocols support streaming publish/subscribe and query/response data models. Although security is not explicitly specified, it is likely, that emerging GMA implementations will utilise the Grid Security Architecture, which is based on X.509 certificates and SSL.

The GMA is designed to be extensible so that producers and directory servers can be added as required. GMA consumers subscribe to receive GMA producer events. GMA provides the means to group providers and consumers together and thus reduce various system overheads. GMA avoids a single point of failure and increases scalability, by using independent monitoring components instead of a centralized monitoring server. However, there is a central repository service (which may physically be distributed) that is used to bind system components together. GMA components that accept requests are required to publish metadata detailing their state and types of event they process.

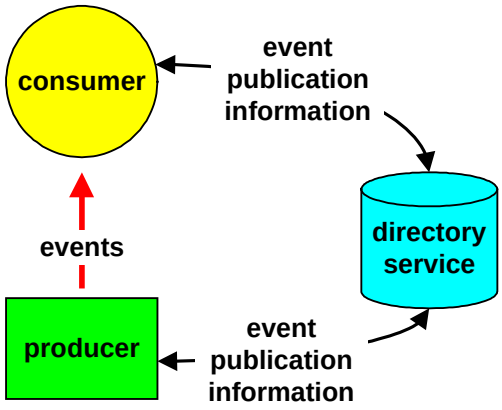

Figure 1: The Grid Monitoring Architecture [11]

4 Related Work

4.1 The Globus HeartBeat Monitor

The Globus Heartbeat Monitor (HBM) [3] was designed to provide a simple but reliable mechanism for detecting and reporting the failure (and state changes) of Globus system processes and application processes. The HBM module has been deprecated in the Globus toolkit.

A daemon ran on each host gathering local process status information. A client was required to register each process that needed monitoring. Periodically, the daemon would review the status of all registered client processes, update its local state and transmit a report (on a per process basis) to a number of specific external data collection daemons. These data collecting daemons provided local repositories that permitted knowledge of the availability of monitored components based on the received status reports. The daemons also recognised status changes and notified applications that registered an interest in a particular process.

The HBM was capable of process status monitoring and fault detection. HBM was unable to monitor resource performance, just availability, was based on a non-standard message format and required component daemons to be ported to all the available platforms used with a Grid.

4.2 NetLogger

The NetLogger (Networked Application Logger) toolkit [6] provides tools for distributed application performance monitoring and analysis. The toolkit enables users to debug, tune and detect bottlenecks in distributed applications. NetLogger components include client libraries, data monitoring, storage, retrieval and visualisation tools, and an open messaging format.

The NetLogger messages can comprise of string, binary or XML encoded formats. The client library allows developers to add calls to existing source code so that monitoring events can be generated from their applications (events can be sent to file, network server, `syslogd`, or memory). The visualisation tool provides a means for event logs to be analysed. The storage and retrieval tools include a daemon to combine NetLogger events from multiple sources to a single central host and an event archive system.

NetLogger provides a means to debug, tune and detect bottlenecks in distributed applications with a common messaging format. NetLogger requires source code modification, there is a single data collection repository, so scalability will be an issue in a Grid environment, and it appears that there is no security within the system.

4.3 Network Weather Service

The Network Weather Service (NWS) [4] is a distributed system that periodically monitors and forecasts the performance that various network and computational resources can deliver over a given time interval. The NWS has been developed to provide statistical quality of service (QoS) information and to support dynamic schedulers. NWS includes sensors for TCP performance (bandwidth and latency), and available CPU and memory. Information taken from the NWS sensors provides data for the current system conditions to be forecast based on numerical models.

Sensors measure the current performance of different resources. To achieve scalability, sensors are organised into sets (cliques), which are then ordered hierarchically. Each clique is configurable and has only one leader (determined by a distributed election protocol). The sensor controller persistently stores sensor measurements in plain text, time stamped strings. The NWS forecaster utilises this information to predict network performance over a specified time interval. NWS has a name server that provides a system wide directory service for NWS processes. All NWS processes are required to periodically refresh registration data with the name server. NWS processes are stateless.

The NWS provides a mechanism for monitoring current resource conditions and forecasting of future conditions. The NWS provides a scalable, extensible and non-intrusive means of monitoring resources. However, NWS uses non-standard message formats, its name server and forecaster are centralised, and appears to have no specialised security built in.

4.4 Autopilot

The Autopilot project, based on the Pablo Toolkit [7], is an infrastructure for real-time adaptive control of parallel and distributed computing resources. The goal of Autopilot is to support monitoring and steering of distributed applications. Autopilot allows application and runtime library developers to create software that can alter their behaviour and optimize performance in response to real-time external environment changes.

Autopilot control mechanisms automatically select and configure resource management algorithms based on observed system performance. Autopilot utilises distributed performance sensors to capture application and system performance data. Resource management policies are adapted in response to changes in the system environment. Sensor information is made available to clients in real-time, while actuators enable remote control of executing code and management policies.

Autopilot provides distributed application performance monitoring in a Grid environment. A Java GUI can be used to interact with remote sensors and actuators, and to present frequently measured host information.

4.5 Remos

The Remos (REsource MOnitoring System) [8] allows network-aware applications to obtain information about their distributed execution environment by providing an interface that hides system heterogeneity. Remos aims to balance information accuracy (best-effort or statistical information) with efficiency (providing a query-based interface), to manage monitoring overheads.

Remos consists of multiple (SNMP-based) collectors, which gather information about the network. Remos permits applications to obtain monitoring information in a portable manner, using platform independent interfaces, however it appears to lack any security mechanisms.

4.6 JAMM

Java Agents for Monitoring and Management (JAMM) [9][10] is a distributed set of components (sensors) that collect and publish monitoring data (events) about computer systems. JAMM agents securely automate the execution of monitoring sensors and the collection of event data.

Gateways allow clients to control or subscribe to a number of monitoring sensors, provide multiplexing/de-multiplexing and information filtering tasks. Loggers control the operation of sensors, and manage subscriptions for sensor output. Sensors execute on host systems, parse output from processes and transmit monitoring information to subscribers. A data collector combines data from multiple sensors into a single file for real-time visualization and analysis. A centralised directory service is required for event consumers to locate (registered) event producers.

JAMM uses a subscription-based, event notification approach for monitoring processes executing on hosts. The central directory service is implemented using a hierarchy of replicated LDAP servers. Java based sensors can be implemented as Activateable objects, permitting sensors to be added, removed dynamically or reconfigured, while non-Java sensors must be installed on each host.

With JAMM, it appears that sensors are key aspects of concern. If the sensor is pure Java then there is only a limited amount of real system data that can be collected without resorting to native system calls. In addition, sensor daemons need to be installed on every resource that is to be monitored or managed.

4.7 Related Work Summary

Varieties of different systems exist for monitoring and managing distributed Grid-based resources and applications. Each system discussed in this section has its own strengths and weaknesses. However, for resource monitoring and management, we believe that none fulfil the criteria that are desired in today's Grid-based environments. That of using standard and open components (Java applets, servlets, JDBC, and an SQL database), being scalable, requiring no additional modification or installation of additional software on the monitored resources (SNMP agents and MIBs), taking advantage of the GGF advocated architecture to bind together the monitored resources (GMA) and security.

5 GridRM Architecture

5.1 Introduction

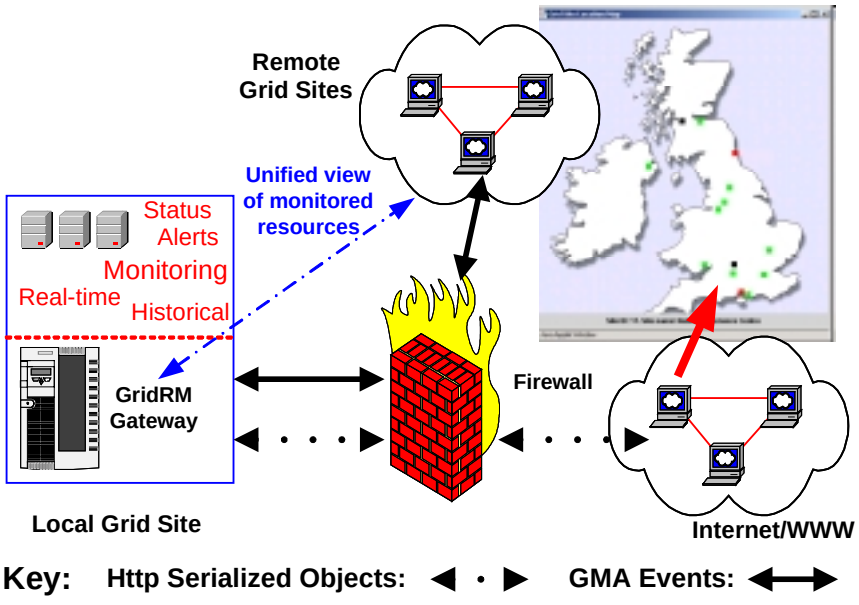

Figure 2: Grid Resource Monitoring Architecture

The GridRM architecture is shown in Figure 2. Each Grid site within a virtual organisation is viewed as an island of resources, with a GridRM Gateway providing:
i. Controlled access to local resource information and
ii. A mechanism for the user to query the status of remote site resources.

Each GridRM Gateway provides a web-based user interface allowing users to observe monitored resources at the local and remote sites; geographic maps present the 'islands of resources' as colour coded icons. These icons are then selected to reveal information about a site's monitored resources. A hierarchical approach is envisioned, whereby a user may proceed from global, to country, to organisational views. As the view is magnified, so to are the details about the interconnecting network links (for example, from intercontinental backbones, to links between 'islands' within a single organisation). Users are free to connect to any GridRM Gateway and retrieve the user interface, however access to monitored resource information is controlled by individual site access policy. Mechanisms will be implemented for distributed session management to ensure the user's view of the Grid is consistent, regardless of the Gateway serving the user interface.

The GridRM architecture has two hierarchical layers, the global layer, shown in Figure 3, and the local layer, shown in Figure 4. The global layer of GridRM uses GMA software to bind together the local islands that are Grid-enabled sites. These islands interact with the GMA via a GridRM Gateway. The layout of a GridRM Gateway is shown in Figure 5. The Gateway is used internally, to a Grid-enabled site to configure, manage and monitor internal SNMP-enabled resources. The gateway provides a type of firewall that ensures there is controlled access to resource information. Finally, specific devices have their SNMP agents and MIBs configured in standard way to provide resource information.

5.2 The Global Layer of GridRM

The global layer of GridRM is shown in Figure 3. This layer functions in the following way:

1. The GridRM Gateways register with the GMA as compound information providers/consumers.
2. A Web client connects to a particular GridRM Gateway and downloads a Java applet displaying monitoring information. When the applet is first initialised at the client, the Gateway utilises the GMA directory to locate other GridRM Gateways within the requested virtual organisation. GMA events are then used to return a small amount of summary information from each site to the client's interface, graphically detailing the composition of the virtual organisation.
3. Based on this information, the client can instruct the local Gateway to return monitoring information from its own Grid site resources, or any of the remote sites (assuming valid user credentials). Information requests and notifications are fulfilled between remote Gateways using GMA events.

Figure 4: The Local GridRM Layer 5.3.1 The Web

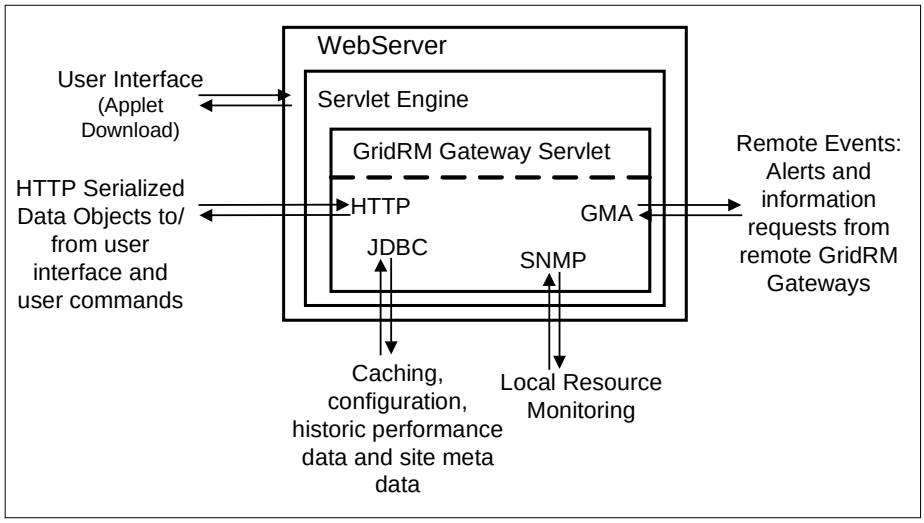

Figure 5: The GridRM Gateway

5.3.2 Interaction with SNMP Agents

Three mechanisms are used for resource monitoring at the GridRM Gateway. The first mechanism is the direct result of a consumer request; the request is received, validated and then processed; if the request was for real-time information, the necessary resources are queried directly for the specified data, which is then returned to the requester. Results from this query are also cached.

A variation on this approach is 'historical' data, which attempts to reduce monitoring intrusiveness, by returning cached information up to a maximum time to live. If a request for historical data is received, the request is fulfilled from the servlet cache, if possible. If the cached data has expired or only part of the data required is present in the cache, then the specified resources are queried directly. A response can be combined from both the cache and direct resource query.

The second approach to resource monitoring is by polling. This implies repeated queries of resource attributes over some time period at a specific frequency. The user can choose to store polled results in the Gateway's database (for later viewing) and/or receive results directly as they are obtained from the resource.

The third type of monitoring refers to events produced by monitored resources. These are captured in a database for later analysis. Users must subscribe to events in order to receive specific resource event-notifications, within their user interface, in real-time. If the user's applet is not connected directly to the GridRM Gateway where a resource event-notification originated, then a GMA event is transmitted from the event source to the remote Gateway serving the user's applet. Distributed user session management is required across Gateways to ensure that users subscribing to certain events through one Gateway receive notifications from remote Gateways on all relevant Grid sites.

5.3.3 The SQL Database

The Gateway is configured with a relational database for storing resource monitoring information, and general site metadata (such as, administrator contact details, URLs to site specific and project information). The availability (lack of) of remote Gateways and other general runtime conditions are logged within the database so that an event trace of the distributed environment can be provided for GridRM administration purposes. Finally, the database also caches results when local resources are probed for status information.

5.3.4 Resource Monitoring Agent Types

A key GridRM criterion is to reduce the amount of administration required to monitor diverse resources; in particular, it is not desirable to install monitoring daemons on all resources. The Simple Network Monitoring Protocol (SNMP) has been chosen to provide performance information, as it is pervasive across a large range of heterogeneous resources. The use of SNMP does not preclude other monitoring protocols and the GridRM Gateway's design is flexible enough to allow other approaches to be used concurrently (such as WBEM). However, a metadata mapping is required between different monitoring protocols, so that regardless of the source, data meaning and equivalence are consistent.

5.3.5 The GridRM User Interface

The GridRM graphical user interface uses a Java Swing-based Applet to provide a seamless view of monitored data across the Grid. Any number of clients can use the GridRM concurrently and the user interface can be downloaded from any GridRM Gateway within the virtual organisation (if the nearest Gateway is unavailable, a user can simply choose another server to download the user interface from).

After initialisation, users are presented with a graphical (geographic) map showing the status of GridRM Gateways at each Grid site. Figure 6 shows an early prototype user interface [12][13]. A coloured icon representing the site's geographic location shows the status of GridRM Gateways and other key resources within the associated Grid site. Selecting an icon reveals detailed monitoring information for the chosen site. A hierarchy of maps can be used to present different views from global sites to country or county level.

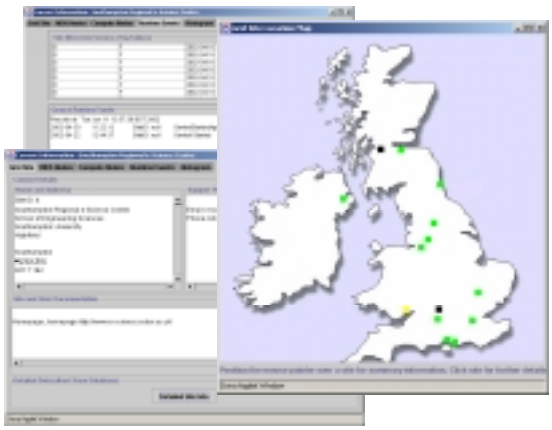

Figure 6: Initial Prototype User Interface for GridRM

The user interface allows resources to be monitored in real-time. The user can directly probe resources, register for event notifications (i.e. when resource thresholds have been exceeded) and schedule historical data capture for analysis later. The GridRM Gateway serving the user interface handles all user requests by transparently coordinating with remote Gateways in order to provide the user with the required data.

5.3.6 Security

At the global GridRM level, security is based on the GSA security model. At the local GridRM level the user interface communicates with the Gateway using encrypted HTTP connections. Users are required to enter a password/certificate when logging into the user interface. Each Gateway maintains an access control list for the local site resources, determining the type/source of information that a user may view. When a Gateway forwards a user request to a remote Gateway, the request is formed under the users login identity; the remote site compares the user certificate in the request with its own access control criteria before proceeding to fulfil the request.

6 Conclusions

GridRM is an open source, generic, distributed, resource-monitoring system for Grid environments. It is composed of a global GridRM layer, based on the Global Grid Forum's, Grid Monitoring Architecture, and a local GridRM layer. The global layer permits GridRM Gateways to exchange monitoring data between Grid sites in a scalable and secure manner, using a combination of subscription based event notification and direct requests for information. The local layer operates within an individual Grid site providing a mechanism (the GridRM Gateway) to monitor management and performance information from local site resources. Gateways collaborate (at the global GridRM layer) to provide a consistent view of the performance of the Grid.

The complexity of gathering resource information from diverse Grid sites is hidden behind a Web-based (Java applet) graphical user interface that seamlessly combines monitoring information from remote sites. The user interface combines all site information within a consistent view; a user can connect to any GridRM Gateway within the virtual organisation and always see a consistent representation of all related Grid resources (those local to and remote from the Gateway).

If a GridRM Gateway or a network link fails, the user need only select a different Gateway from which to obtain the user interface. In the event of a failure (network link and/or Gateway failure), Gateways are alerted and continue to operate with their remaining peers. Subscription based event propagation ensures that data is only passed between Gateways when the information is actually needed; event consumers can go off-line unexpectedly and all associated event propagation is terminated.

6.1 Future Work

A number of GridRM features need further investigation before the prototype is further developed:

1. Distributed user session management across Gateways. This feature is required to ensure a consistent (and seamless) user's view of the Grid.
2. Distribution and replication of the GMA directories. This matters needs to be considered to avoid a single point of failure and avoid scalability issues. A suitable mechanism must be used to efficiently locate Gateways within the GridRM architecture.
3. Monitoring Network Connections between Grid Sites. The primary focus of GridRM has been monitoring resource information for local and remote Grid sites. An important feature is the interconnection between Gateways; this will be added to the next prototype.

6.2 Summary

A variety of different systems exist for monitoring and managing distributed Grid-based resources and applications. In terms of resource monitoring and management, we believe that none fulfil the criteria that are desired in today's Grid-based environments. Our criteria include standard and open components, scalability, security, a seamless, consistent view of remote Grid resources, and the requirement that no additional modifications or further software installations should be required for monitored resources. This paper presents GridRM, a resource monitoring architecture that has been devised to meet this set of needs.

To date an early Grid monitoring prototype has been produced. Based on these experiences, and the architecture described in this paper, an enhanced GridRM prototype is under construction. Details of our experiences with the new prototype (for example, request processing performance, distributed session management and failure recovery issues) will be reported in later papers.

7 References

[1] Global Grid Forum, http://www.gridforum.org/

[2] A Grid Monitoring Architecture, B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany, Working Document, January 2002, http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-2.pdf
[3] The Globus Heartbeat Monitor Specification v1.0, The Globus Project, http://www-fp.globus.org/hbm/heartbeat_spec.html
[4] Network Weather Service, http://nws.cs.ucsb.edu/, 7th June 2002.
[5] R. Wolski et al, The Network Weather Service, A Distributed Resource Performance Forecasting Service for Metacomputing, http://www.isi.edu/~annc/classes/grid/lectures/mohammed.ppt, 9th June 2002.
[6] The NetLogger Toolkit: End-to-End Monitoring and Analysis of Distributed Systems, DIDC, Lawrence Berkley National Laboratory http://www-didc.lbl.gov/NetLogger/, 5th June 2002.
[7] Scalable Performance Tools (Pablo Toolkit), Department of Computer Science, University of Illinois at Urbana-Champaign, USA, http://www-pablo.cs.uiuc.edu/Project/Pablo/ScalPerfToolsOverview.htm, June 29th 1999.
[8] Remos: Resource Monitoring for Network-Aware Applications, http://www-2.cs.cmu.edu/~cmcl/remulac/remos.html, April 2002.
[9] Java Agents for Monitoring and Management (JAMM), DIDCG, Lawrence Berkeley National Laboratory, http://www-didc.lbl.gov/JAMM/, 6th July 2000.
[10] Tierney,B., et al, A Monitoring Sensor Management System for Grid Environments, June 2000.
[11] Global Grid Forum, Grid Monitoring Architecture Working Group, http://www-didc.lbl.gov/GGF-PERF/GMA-WG/, June 2002.
[12] M.A. Baker and G. Smith, A Prototype Grid-site Monitoring System, Version 1, DSG Technical Report 2002.01, http://homer.csm.port.ac.uk/publications/ technical/reports /grid/DSGmonitoring.pdf, January 2002.
[13] M.A. Baker and G. Smith, Grid Monitor Applet: Prototype 1, online demonstration, http://homer.csm.port.ac.uk:8088/MonitorServlet/GridMonitorApplet.html, November 2002.
[14] The Globus Project, http://www.globus.org, June 2002.