MONALISA: AN AGENT BASED, DYNAMIC SERVICE SYSTEM TO MONITOR, CONTROL AND OPTIMIZE GRID BASED APPLICATIONS

I.C.Legrand, H.B.Newman, California Institute of Technology, Pasadena, CA 91125, USA
R.Voicu, European Center for Nuclear Research – CERN, Geneva, Switzerland
C.Cirstoiu, C.Grigoras, M.Toarta, C. Dobre, Polytechnic University Bucharest, Romania

Abstract

The MonALISA (Monitoring Agents in A Large Integrated Services Architecture) system provides a distributed service architecture which is used to collect and process monitoring information. While its initial target field of application is networks and Grid systems supporting data processing and analysis for global high energy and nuclear physics collaborations, MonALISA is broadly applicable to many fields of "data intensive" science, and to the monitoring and management of major research and education networks. MonALISA is based on a scalable Dynamic Distributed Services Architecture), and is implemented in Java using JINI and WSDL technologies. The scalability of the system derives from the use of a multi threaded engine to host a variety of loosely coupled self-describing dynamic services, the ability of each service to register itself and then to be discovered and used by any other services, or clients that require such information. The framework integrates many existing monitoring tools and procedures to collect parameters describing computational nodes, applications and network performance. Specialized mobile agents are used in the MonALISA framework to perform global optimization tasks or help and improve the operation of large distributed system by performing supervising tasks for different applications or real time parameters. MonALISA is currently running around the clock monitoring several Grids and distributed applications on around 160 sites.

INTRODUCTION

An essential part of managing a global Data Grid is a monitoring system that is able to monitor and track the many site facilities, networks, and the many tasks in progress, in real time. The monitoring information gathered also is essential for developing the required higher level services, and components of the Grid system that provide decision support, and eventually some degree of automated decisions, to help maintain and optimize workflow through the Grid.

We therefore developed the MonALISA [1] system, designed as an ensemble of autonomous multi-threaded, self-describing agent-based subsystems which are registered as dynamic services, and are able to collaborate and cooperate in performing a wide range of monitoring tasks in large scale distributed applications. Each MonALISA service is capable of being discovered and used by other services or clients that require such information. The distributed service system [2] uses mobile agents to gather, disseminate and coordinate configuration, time-dependent state and other information in the Grid system as a whole. Mobile code (implemented through agents and proxy-servers) is used to provide a flexible approach to accessing and analyzing information in distributed systems.

The design of this structure is based on a Station Server unit, which is a generic network server that can host a variety of agent-based Dynamic Services. The Station Services are dynamically interconnected (peer-to-peer) and provide a distributed "fabric" for hosting services. The Station Service implementation is made available as a network service using JINI [3] technology. This framework allows cooperating services and applications to access each other seamlessly, to adapt to a dynamic environment, and to share code and configuration information transparently. This system design avoids single points of failure, and allows for automatic service replication and re-activation. These features make it capable of providing reliable, autonomous support for large scale distributed applications such as global Grid systems under real conditions, where individual (or multiple) components may fail.

THE DISTRIBUTED SERVICES ARCHITECTURE

A service in the DDSA framework is a component that interacts autonomously with other services through dynamic proxies or agents that use self-describing protocols. By using dedicated lookup services, a set of distributed services registry, and the discovery and notification mechanisms, the services are able to access each other seamlessly. The use of dynamic remote event subscription allows a service to register itself to be notified of a selected set of event types, even if there is no provider to do the notification at registration time. The lookup discovery service will then automatically notify all the subscribed services, when a new service, or a new service attribute, becomes available.

The services are managed by an efficient multithreading engine that schedules and oversees their execution, such that data handling operations are not disrupted if one or more tasks (threads) are unable to

continue. The system design also provides reliable "non-stop" support for large distributed applications under realistic working conditions, through service replication, and automatic re-activation of services. These mechanisms make the system robust against the failure or inaccessibility of multiple Grid components.

THE MONITORING SERVICE

MonALISA is designed to easily integrate existing monitoring tools and procedures and to provide this information in a dynamic, self describing way to any other services or clients. MonALISA services are organized in groups and this attribute is used for registration and discovery.

The system monitors and tracks site computing farms and network links, routers and switches using SNMP [4], and it dynamically loads modules that make it capable of interfacing existing monitoring applications and tools (e.g. Ganglia [5], MRTG [6]).

The core of the monitoring service is based on a multi-threaded system used to perform the many data collection tasks in parallel, independently. The modules used for collecting different sets of information, or interfacing with other monitoring tools, are dynamically loaded and executed in independent threads. In order to reduce the load on systems running MonALISA, a dynamic pool of threads is created once, and the threads are then reused when a task assigned to a thread is completed. This allows one to run concurrently and independently a large number of monitoring modules, and to dynamically adapt to the load and the response time of the components in the system. If a monitoring task fails or hangs due to I/O errors, the other tasks are not delayed or disrupted, since they are executing in other, independent threads. A dedicated control thread is used to stop properly the threads in case of I/O errors, and to reschedule those tasks that have not been successfully completed. A priority queue is used for the tasks that need to be performed periodically. This approach makes it relatively easy to monitor a large number of heterogeneous nodes with different response times, and at the same time to handle monitored units which are down or not responding, without affecting the other measurements. The system allows collecting information in both push and pull modes from different types of tools or applications.

The clients, other services or agents can get any real-time or historical data by using a predicate mechanism for requesting or subscribing to selected measured values. These predicates are based on regular expressions to match the attribute description of the measured values a client is interested in. They may also be used to impose additional conditions or constraints for selecting the values. The subscription requests create a dedicated thread, to serve each client. This thread performs a matching test for all the predicates submitted by a client with the measured values in the data flow. The same thread is responsible to send the selected results back to the client as compressed serialized objects. Having an independent thread per client allows sending the information they need, in a fast and reliable way, and it is not affected by communication errors which may occur with other clients. In case of communication problems these threads will try to reestablish the connection or to clean-up the subscriptions for a client or a service which is no longer active.

Monitoring data requests with the predicate mechanism is also possible using the WSDL/SOAP binding from clients or services written in other languages. The class description for predicates and the methods to be used are described in WSDL, and any client can create dynamically and instantiate the objects it needs for communication. Currently, Web Services technology does not provide the functionality to register as a listener, and to receive the future measurements a client may want to receive.

Other applications or clients may also use the Agent Filters to receive the information they need. The Agent Filter is a java module which can be dynamically deployed to any MonALISA service. It is designed to perform a dedicated data processing task on local data (by subscribing with a predicate to the data flow) and to return the processed information periodically. The MonALISA service provides the run time environment for these agents, which must be digitally signed by a trusted certificate. Dynamically loadable alarm agents, and agents able to take actions when abnormal behavior is detected, are currently being developed to help with managing and improving the working efficiency of the facilities, and the overall Grid system being monitored.

The clients, or any other services, use a set of proxies to connect and get information from the monitoring services. These proxy services are used to allow monitoring services to run behind firewalls, and to control the connections performed by services. At the same time, these services are used to provide an intelligent multiplexing of the same information if requested by more than one client or service. The way clients connect to monitoring information using the MonALISA proxy services is presented in Figure 2.

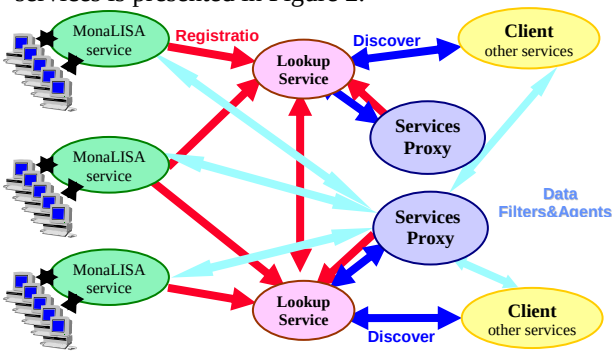

Figure 2. MonALISA proxy services are used for accessing monitoring information from clients or other services.

In general, clients discover the nearest proxy service and use it to get the information, but a dynamic load-balancing mechanism is also used to distribute the load among the available proxies, so that monitoring information is served to many clients or services without increasing the number of connections or load on individual monitoring services.
MonALISA provides a secure Administration mechanism (SSL with X.509 certificates) for dynamic configuration, using a dedicated GUI, of farms / network elements, and support for other higher level services that aim to manage a distributed set of facilities and/or optimize workflow. The system is currently deployed on many sites and maintaining and updating such widely deployed applications often requires a significant effort. For this reason we developed a mechanism in MonALISA that allows us to automatically update the monitoring service.

CLIENTS AND DATA ACCESS

We have developed a global graphical client which uses the discovery mechanism to find all of the active services from a list of user defined groups. This graphical client is implemented as a Web Start application that can be started and used from any web browser with little effort.

A MonALISA service may provide its own GUI to any client as a complex proxy containing the marshaled components as an attributed to the service [1]. This GUI communicates with each service from which the user wants detailed information and plots the requested values. MonALISA provides flexible access to real-time or historical monitoring values by using either a predicate subscription mechanism or dynamically loadable filter agents. These mechanisms are used by any interested client to query and subscribe to only the information it needs, or to generate specific aggregate values in an appropriate format. When a client subscribes with a predicate to certain values, the GUI will automatically update as new values matching the subscriptions are collected.

The graphical user interface allows users to visualize global parameters from multiple sites, as well as detailed tracking of parameters for any individual site or component in the entire system. The graphical clients also use the remote notification mechanism, and thus are able to dynamically show when new services are started or when services become unavailable.

In Figure 3, we present a few examples in how real-time and historical data are presented in MonALISA.
The GUI panels allow statistical operations to be run on the monitoring data in order to generate aggregate values or distributions. MonALISA can be used by any application to report specific parameters such that the execution can be monitored and results verified.

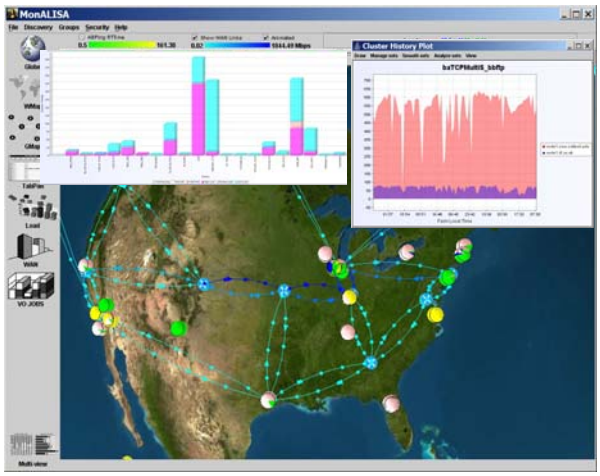

Figure 3. MonALISA is used to monitor the Grid2003 [7] (around 30 sites and ~2800 CPUs) and the Abilene backbone. It provides real-time and historical data for each component in the system as well as the number of jobs or grid-ftp transfers executed on any of the sites.

GLOBAL REPOSITORIES

A generic framework for building “pseudo-clients” for the MonALISA services has been developed. This has been used for creating dedicated Web service repositories with selected information from specific groups of MonALISA services. The “pseudo-clients” use the LUSs to find all of the active MonALISA services from a specified set of groups. Next they subscribe to these services with a list of predicates and filters. These predicates and filters specify the information that the pseudo-client wants to collect from all the services. The “pseudo-client” stores all of the values received from the running services in a local database, and uses procedures written as Java threads to compress old data.

The storage client is a Java application that connects to a given set of MonALISA services, registers a set of predicates, and stores the data received in a database. This data is used to plot a variety of charts in the web interface attached to the storage client. This way the users have a customized global view of the Grid. Data storage is optimized for space and speed.

The Global Repository [8] also has the ability to dynamically display the stored data in multiple chart types: history charts, real-time bar charts, real-time pie charts, history pie charts, combined views and map information. Each chart has a corresponding configuration file with a very simple structure that offers flexibility to the site administrators. The plots are dynamically generated and allow the user to select any time interval and to compare different sites or parameters.
The same web interface is used to display details for any measured metric. Figure 4 shows the main parameters describing the operations on an individual site.

The storage client and a Tomcat servlet engine are started inside the same JVM to ensure that the real-time charts display the most recent information, and to minimize the repository memory usage.

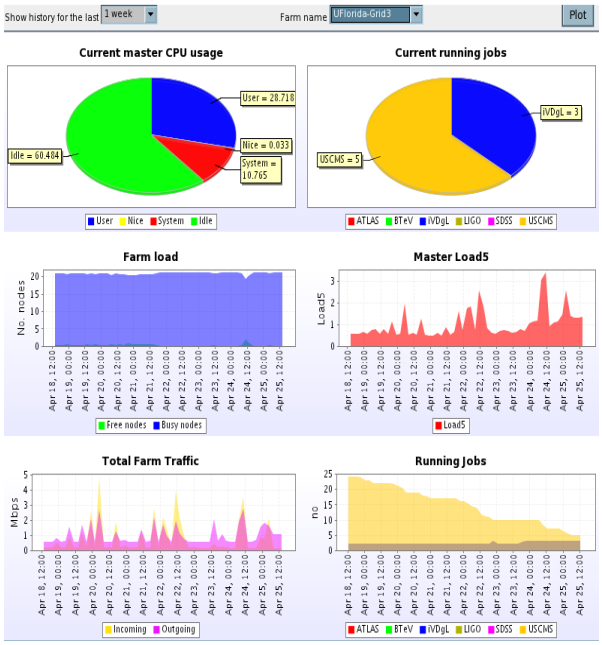

Figure 4. Plots presenting the recent history for the major parameters (jobs running, farm load, total traffic) at UFlorida Grid2003 center.

The WSDL/SOAP interface is available in both the services and the repositories so that clients can access data from a specific Grid farm or, through a repository, they can access information received from several Grid farms. One example is the STAR Scheduler [9], which will use global information provided by a repository to schedule job execution.

We have developed agents able to provide an optimized dynamic routing of the videoconferencing data streams for the VRVS [10] system. These agents use information about the quality of the alternative connections in the system to produce, in real-time, a minimum spanning to optimize the data flow at the global level.

Monitoring agents perform ping-style measurements using UDP probes to measure the quality of the connection with possible peer reflectors. These agents are deployed on all MonALISA services that run on the reflectors.

SUMMARY

Deploying these monitoring services on many sites and interfacing it with other monitoring tools (SNMP, Ganglia, LEMON, MRTG), batch queuing systems (Condor, LSF, PBS) and a large number of applications has provided very useful experience, and has enabled us to begin building reliable and scalable distributed services.

MonALISA is a robust monitoring system, providing a flexible interface. It allows rapid development of complex clients that can either display the status of the whole system in real-time, or compute complex algorithms based on monitored data in order to optimize various aspects of the observed system.

It is currently being used to monitor large facilities for data processing in High Energy Physics and it is deployed on more than 50 HEP sites participating in different experiments (CMS, ALICE, STAR, CDF, ATLAS). It is also used to monitor several major WAN networks: Abilene backbone [11], GLORIAD [12], CERN-US links, CERN-IN2P3 link.

This experience also has been important in enabling us to start building higher level services, to perform job scheduling and data replication tasks effectively; service that adapt themselves dynamically to respond to changing load patterns in large Grids.

ACKNOWLEDGMENTS

The authors wish to thank to J. Bunn, P. Galvez, S. Ravot, G. Denis, Y.Xia, X. Su, S. Singh and M. Thomas form California Institute of Technology, R. Cavanaugh from University of Florida, L. Bauerdick, I. Fisk, J. Weigand and Y. Wu from FERMILAB, N. Tapus from the Polytechnic University Bucharest, L. Cottrel from Stanford Linear Accelerator Center, M. Mambelli from University of Chicago, O. Martin, F. Carminati, P. Buncic M. Lamanna and V. Innocente from European Organization for Nuclear Research, W. Matthews from Georgia Institute of Technology, E. Boyd from Internet2, J.Laurent and E. Efstathiadis from Brookhaven National Laboratory for their help in deploying and supporting MonALISA.

REFERENCES

[1] MonALISA web page: http://monalisa.cacr.caltech.edu
[2] H.B. Newman, I.C. Legrand, J.J. Bunn, “A Distributed Agent-based Architecture for Dynamic Services” CHEP 2001, Beijing, Sept 2001,
[3] Jini web page, http://www.jini.org
[4] The Net-Snmp Web Page, http://www.net-snmp.org/
[5] Ganglia Monitoring tool, http://ganglia.sourceforge.net/
[6] MRTG monitoring tool. http://www.mrtg.org
[7] Grid2003: http://www.ivdgl.org/
[8] MonALISA web repository, http://monalisa.cacr.caltech.edu:8080/
[9] STAR Scheduler http://www.star.bnl.gov/STAR/comp/Grid/scheduler/
[10] The VRVS system web page: http://www.vrvs.org
[11] Internet2: http://internet2.org
[12] Gloriad: http://www.gloriad.org