# Monitoring the Grid with the Globus Toolkit MDS4

Jennifer M. Schopf[1,2,4], Laura Pearlman[3], Neill Miller[2], Carl Kesselman[3], Ian Foster[1,2], Mike D'Arcy[3], Ann Chervenak[3]
[1] *Mathematics and Computer Science Division, Argonne National Laboratory*
[2] *Department of Computer Science, The University of Chicago*
[3]*Information Science Institute, University of Southern California*
[4] *UK National eScience Centre, University of Edinburgh*

**Abstract.** The Globus Toolkit Monitoring and Discovery System (MDS4) defines and implements mechanisms for service and resource discovery and monitoring in distributed environments. MDS4 is distinguished from previous similar systems by its extensive use of interfaces and behaviors defined in the WS-Resource Framework and WS-Notification specifications, and by its deep integration into essentially every component of the Globus Toolkit. We describe the MDS4 architecture and the Web service interfaces and behaviors that allow users to discover resources and services, monitor resource and service states, receive updates on current status, and visualize monitoring results. We present two current deployments to provide insights into the functionality that can be achieved via the use of these mechanisms.

## 1. Overview

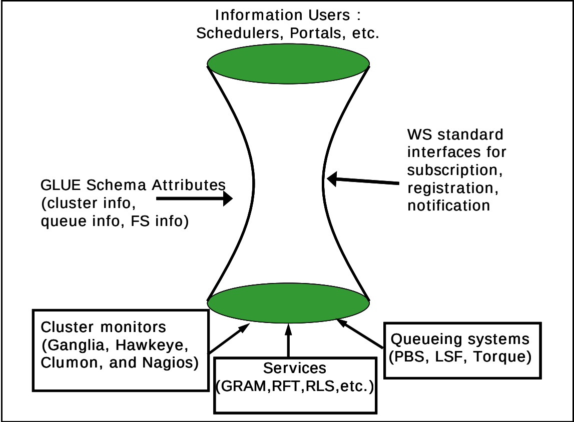
The resources available to a virtual organization (VO) in a Grid environment can change frequently as new resources and services are added and old ones are removed or become inaccessible. In addition, resource and service properties may change: for example, when a data server is upgraded to larger capacity, different access rates, or different access protocols. These dynamic behaviors can make both *discovery*—the process of finding suitable resources to perform a task—and *monitoring*—the process of observing resources or services to track their status for purposes such as fixing problems and tracking usage—significant undertakings.

Typical monitoring and discovery use cases include providing data so that resource brokers can locate computing elements appropriate for a job, streaming data to a steering application so that adjustments can be made to a running application, and notifying system administrators when changes in system load or disk space availability occur, in order to identify possible performance anomalies.

The Globus Toolkit's solution to these closely related problems is its Monitoring and Discovery System (MDS): a suite of components for monitoring and discovering resources and services. MDS4, the version in the Globus Toolkit 4 [1], uses standard interfaces defined within the WS-Resource Framework (WSRF) and WS-Notification (WS-N) specifications [2] to provide query and subscription interfaces to arbitrarily detailed resource data (modeled in XML).

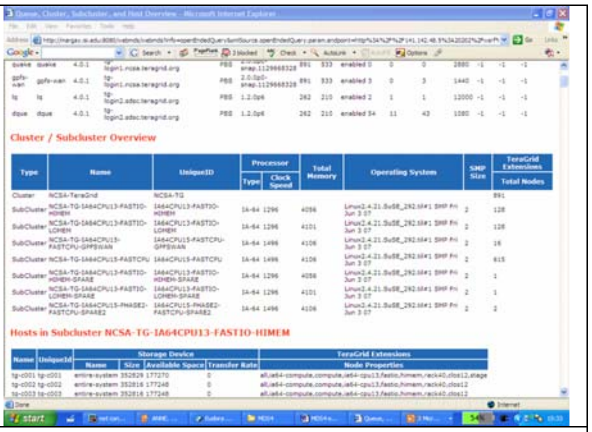
## 2. MDS4 Details

MDS4 builds heavily on capabilities provided by the WSRF and WS-N specifications; indeed, it can be viewed as an exemplary use case for those specifications, which define the mechanisms used to describe information sources, access information via both queries and subscriptions, and manage information lifetimes. The neck of the MDS4 "protocol hourglass" (Figure 1) comprises not only these standard protocols for data access and delivery but also standard schemas for information representation, such as the GLUE schema [3]. Below the neck of the hourglass, MDS4 interfaces to different local information sources, translating their diverse schemas into appropriate XML schema transmitted over WSRF/WS-N protocols. Above the neck of the hourglass, various tools and applications can take advantage of the uniform Web services query, subscription, and notification interfaces to those information sources that MDS4 implements.



Information Users :
Schedulers, Portals, etc.
WS standard interfaces for subscription, registration, notification
GLUE Schema Attributes (cluster info, queue info, FS info)
Cluster monitors (Ganglia, Hawkeye, Clumon, and Nagios)
Services (GRAM,RFT,RLS,etc.)
Queueing systems (PBS, LSF, Torque)

**Figure 1: MDS4 provides a protocol hourglass.**



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [illegible] | [illegible] | 4.0.1 | tg-login1.ncsa.teragrid.org | PBS | 2.0.0p0-snap.1129668328 | 891 | 533 | enabled | 0 | 0 | 0 | 2880 | -1 | -1 | -1 |
| gpfs-wan | gpfs-wan | 4.0.1 | tg-login1.ncsa.teragrid.org | PBS | 2.0.0p0-snap.1129668328 | 891 | 533 | enabled | 3 | 0 | 3 | 1440 | -1 | -1 | -1 |
| lg | lg | 4.0.1 | tg-login2.sdsc.teragrid.org | PBS | 1.2.0p6 | 262 | 210 | enabled | 2 | 1 | 1 | 12000 | -1 | -1 | -1 |
| dque | dque | 4.0.1 | tg-login2.sdsc.teragrid.org | PBS | 1.2.0p6 | 262 | 210 | enabled | 54 | 11 | 43 | 1080 | -1 | -1 | -1 |

Cluster / Subcluster Overview

| Type | Name | UniqueID | Processor | | Total Memory | Operating System | SMP Size | TeraGrid Extensions |
|---|---|---|---|---|---|---|---|---|
| | | | Type | Clock Speed | | | | Total Nodes |
| Cluster | NCSA-TeraGrid | NCSA-TG | | | | | | 891 |
| SubCluster | NCSA-TG-IA64CPU13-FASTIO-HIMEM | IA64CPU13-FASTIO-HIMEM | IA-64 | 1296 | 4056 | Linux2.4.21.SuSE_292.til#1 SMP Fri Jun 3 07 | 2 | 128 |
| SubCluster | NCSA-TG-IA64CPU13-FASTIO-LOMEM | IA64CPU13-FASTIO-LOMEM | IA-64 | 1296 | 4101 | Linux2.4.21.SuSE_292.til#1 SMP Fri Jun 3 07 | 2 | 128 |
| SubCluster | NCSA-TG-IA64CPU15-FASTCPU-GPFSWAN | IA64CPU15-FASTCPU-GPFSWAN | IA-64 | 1496 | 4106 | Linux2.4.21.SuSE_292.til#1 SMP Fri Jun 3 07 | 2 | 16 |
| SubCluster | NCSA-TG-IA64CPU15-FASTCPU | IA64CPU15-FASTCPU | IA-64 | 1496 | 4106 | Linux2.4.21.SuSE_292.til#1 SMP Fri Jun 3 07 | 2 | 615 |
| SubCluster | NCSA-TG-IA64CPU13-FASTIO-HIMEM-SPARE | IA64CPU13-FASTIO-HIMEM-SPARE | IA-64 | 1296 | 4056 | Linux2.4.21.SuSE_292.til#1 SMP Fri Jun 3 07 | 2 | 1 |
| SubCluster | NCSA-TG-IA64CPU13-FASTIO-LOMEM-SPARE | IA64CPU13-FASTIO-LOMEM-SPARE | IA-64 | 1296 | 4101 | Linux2.4.21.SuSE_292.til#1 SMP Fri Jun 3 07 | 2 | 1 |
| SubCluster | NCSA-TG-IA64CPU15-PHASE2-FASTCPU-SPARE2 | IA64CPU15-PHASE2-FASTCPU-SPARE2 | IA-64 | 1496 | 4106 | Linux2.4.21.SuSE_292.til#1 SMP Fri Jun 3 07 | 2 | 2 |

Hosts in Subcluster NCSA-TG-IA64CPU13-FASTIO-HIMEM

| Name | UniqueId | Storage Device | | | | TeraGrid Extensions |
|---|---|---|---|---|---|---|
| | | Name | Size | Available Space | Transfer Rate | Node Properties |
| tg-c001 | tg-c001 | entire-system | 352829 | 177270 | 0 | all,ia64-compute,compute,ia64-cpu13,fastio,himem,rack40,clos12,stage |
| tg-c002 | tg-c002 | entire-system | 352816 | 177248 | 0 | all,ia64-compute,compute,ia64-cpu13,fastio,himem,rack40,clos12 |
| tg-c003 | tg-c003 | entire-system | 352816 | 177248 | 0 | all,ia64-compute,compute,ia64-cpu13,fastio,himem,rack40,clos12 |

**Figure 2: WebMDS page for TeraGrid resource discovery and monitoring.**

## 2.1 *Higher-Level Services*

An MDS4 *Index service* collects information about Grid resources and makes this information available as resource properties. It differs from a UDDI registry [4] primarily in the facts that it stores not only the location from which a piece of data is available, but also a cached version of the data—and maintains that cached copy current via lifetime management mechanisms. A Grid will typically operate multiple Indexes that maintain different data for different purposes. Each GT4 container has a default Index that records resources created within the container. In addition, sites and VOs may maintain one or more Indexes to record available containers, resources, and services.

The *Trigger service* collects information and compares that data against a set of conditions defined in a configuration file. When a condition is met, an action takes place, such as emailing a system administrator when the disk space on a server reaches a threshold.

The Index and Trigger service implementations are both built on the *Aggregator Framework*, a software framework for building services that collect and aggregate data. This framework can be used to construct other services: for example, it would be straightforward to implement a variant of the Index service that makes data available in Condor ClassAds format.

Services built on this framework are sometimes called *aggregator services*. Such services share common mechanisms to:

- Collect information (via subscription, notification, or execution) from information providers
- Use a common configuration mechanism to specify what data to get, and from where.
- Use a soft consistency mode so that published information is renewed at a administrator-controllable frequency.
- Self-clean the data storage by tagging each registration with a lifetime, and if a registration expires without being refreshed, it and its associated data are removed from the server. Thus, outdated entries are removed automatically when they cease to renew their registrations.

### *2.2 Information Providers*

Information providers (IP) are the mechanism MDS4 uses to publish data into an aggregator service. An IP can be a WSRF-compliant service from which data is obtained via WS-ResourceProperty or WS-Notification mechanisms or an executable program that obtains data via some domain-specific mechanism, such as executing a script, file scraping or running an executable. Currently available IPs include cluster data interfaces to Nagios, Ganglia, CluMon and Hawkeye, basic queue data fetched from PBS, LSF or Condor, and service data from all GT4 Web services (CAS, GRAM, RFT, MDS) as well as GridFTP and RLS.

### *2.3 User Interfaces*

An advantage of using a standard, widely-adopted data format such as XML is that one can then use various commodity tools to manipulate data. For example, we have developed a tool called *WebMDS* that uses standard resource property requests to query resource property data and XSLT transforms to format and display them, as shown in Figure 2. In this way, we obtain user-friendly front-end to Index data. In addition, GT4 command-line clients (wsrf-query, wsrf-get-property, wsrf-get-properties) and corresponding Java, C, and Python API implement resource property query operations that can be used to query an Index directly, when required.

## **3. Use Cases**

### *3.1 TeraGrid: Resource Discovery*

TeraGrid [5] is an infrastructure project across nine US sites to create an integrated, persistent computational resource, which enables more than 102 teraflops of computing capability and more than 15 petabytes (quadrillions of bytes) of online and archival data storage with rapid access and retrieval over high-performance networks. TeraGrid has been used by over 1,000 projects.

One of the near-term goals for the TeraGrid project is to deploy a scheduling system that can interact with the TeraGrid sites to make resource selection decisions. Progress toward this goal, however, has been impeded by the lack of a common monitoring framework across the various sites, in part due to different local policies – different sites have deployed different queuing systems (Open PBS, PBS Pro, Torque, etc.) and different cluster monitoring systems (Clumon, Nagios, Ganglia, etc.). As part of the current GT4 deployment on TeraGrid, sites are deploying MDS4 infrastructure that interacts with local systems to gather data and provides a single, standard interface to the data. While there is currently no common agreement for all of the data needed to make resource selection decisions, based on our previous work with scheduling systems and analysis of several common schedulers used with queued platforms such as the TeraGrid, we have defined a set of approximately 30 attributes to gather from each site.

We currently gather queue-specific information (e.g., location of the GRAM server for a queue, the number of jobs currently waiting in that queue) and host-specific information (e.g., the CPU type and speed and the operating system). In addition, we allow for the grouping of similarly configured hosts into clusters and subclusters as a way of aggregating some of the host information. Subclusters are defined by the local site administrators such that they contain a set of hosts that have essentially the same configuration (CPU information, OS information, etc.), and data is reported for the subcluster as whole. In addition, each host record includes the name of the subcluster that the host belongs to.

As a result of the current MDS4 deployment, users will have access to a simple Web interface (shown in Figure 2) to examine resource selection decisions, and metaschedulers will have a common interface to the data they need across the TeraGrid, through either command line or Java APIs.

### *3.2 Earth Systems Grid: Warning on Errors*

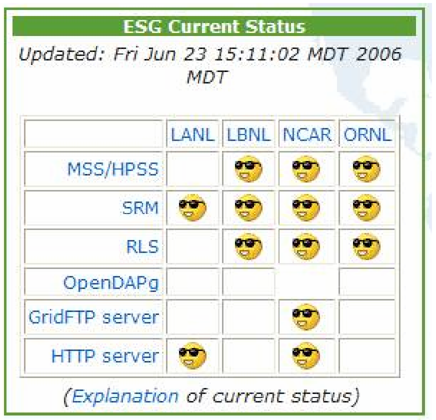
The Earth System Grid (ESG) project [6] supports the next generation of climate modeling research by providing the infrastructure and services that allow climate scientists to publish and access key data sets based on climate simulation models. Important datasets that are provided by ESG to the climate community include simulations generated by the NCAR Community Climate System Model (CCSM)

[7] and the Intergovernmental Panel on Climate Change (IPCC) [8] using infrastructure spread across 7 sites. These datasets are accessed by scientists throughout the world.

The Earth System Grid has become an important community resource for climate scientists. In 2005, users of the NCAR web portal issued 37,285 requests to download 10.25 terabytes of data. Use of the ESG portal has steadily increased, both in terms of the amount of data downloaded and the number of registered users of the system. By the fourth quarter of 2005, NCAR portal users downloaded approximately two terabytes of data per month. ESG registered 1881 users in 2005 and is currently adding users at a rate of more than 150 per month.

Currently, the MDS4 deployment for ESG involves information providers to check the status of the GridFTP data transfer services [9], the OPeNDAP [10] service which is used to filter and subset data to reduce the amount of data that must be transferred the ESG, the web portal, two HTTP servers for alternate data access, Replica Location Service catalogs [11], Storage Resource Managers  [12], and hierarchical mass storage systems.

Data from these information providers are reported to an Index service, and the resource information collected by the MDS4 Index Service is also queried by the ESG web portal, shown in Figure 3. The information in the Index Service is also polled periodically by the Trigger Service, which determines whether specified trigger rules and conditions are satisfied and, if so, email is sent to the system administrators to notify them of a service failure.



**ESG Current Status**

*Updated: Fri Jun 23 15:11:02 MDT 2006 MDT*

| | LANL | LBNL | NCAR | ORNL |
|---|---|---|---|---|
| MSS/HPSS | | 😎 | 😎 | 😎 |
| SRM | 😎 | 😎 | 😎 | 😎 |
| RLS | | 😎 | 😎 | 😎 |
| OpenDAPg | | | | |
| GridFTP server | | | 😎 | |
| HTTP server | 😎 | | 😎 | |

*(Explanation of current status)*

**Figure 3: The ESG portal queries the Index service and displays resource satus -- smiling faces indicate functioning services.**

MDS4 has also been used to deduce the reason behind failures in at least two ways. First, we have found that a failure examined in isolation does not accurately reflect the state of the system, and that with system-wide data a pattern of failure messages that occur close together in time can indicate a problem at a higher level. For example, failure messages indicated that hierarchical storage resource managers at three different locations have failed simultaneously. Since the chance of such simultaneous failures is remote, these errors are more likely an indication of a network outage or some failure of the monitoring service or the client that queries the state of storage resource managers and hierarchical storage systems.

Similarly, we have used the MDS to track data at a lower level to understand which component is causing errors. The ESG portal at NCAR crashed periodically due to a lack of available file descriptors. By using the monitoring infrastructure to check the usage of file descriptors, we were able to detect how many file descriptors had been opened by the different services running on the portal machine and eliminate some suspected sources of problems. We were also able to detect a sudden spike in file descriptor usage to help debug the problem.

## 4. Conclusions

We have described how monitoring and discovery capabilities can be integrated into the design of a distributed computing infrastructure so that any and every resource and service can be monitored and discovered in a uniform manner. Using Web service standards that define the primitive interfaces and behaviors, we have built the basis of a monitoring system for Grid use. It is currently deployed nationally for both resource discovery and error notifications.

## Acknowledgments

Additional MDS support was provided by John Bresnahan, Mike Link, Eric Blau, and Mei-Hui Su. 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. Additional support was provided by NSF NMI Award SCI-0438372.

## References

[1] I. Foster, “A Globus Toolkit Primer”, www.globus.org/primer, 2005.

[2] I. Foster, K. Czajkowski, D. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, “Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF”, *Proc. of the IEEE*, 93 (3). 604-612. 2005.

[3] “Glue Schema Specification”, www.hicb.org/glue/glue-schema/schema.html , 2005.

[4] “UDDI Standard”, http://www.uddi.org , 2006.

[5] “TeraGrid”, www.teragrid.org , 2006.

[6] D. Bernholdt, S. Bharathi, D. Brown, K. Chanchio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox, J. Garcia, C. Kesselman, R. Markel, D. Middleton, V. Nefedova, L. Pouchard, A. Shoshani, A. Sim, G. Strand, and D. Williams, “The Earth System Grid: Supporting the Next Generation of Climate Modeling Research”, *Proc. of the IEEE*, 93 (3), p 485-495, 2005.

[7] "Community Climate System Model”, http://www.cgd.ucar.edu/csm/, 2006.

[8] "Intergovernmental Panel on Climate Change”, http://www.ipcc.ch/, 2006.

[9] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus Striped GridFTP Framework and Server," *Proc. SuperComputing 2005 (SC05)*, 2005.

[10] OPeNDAP Inc., "OPeNDAP: Open-source Project for a Network Data Access Protocol”, http://opendap.org/, 2005.

[11] A. L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, and R. Schwartzkopf, "Performance and Scalability of a Replica Location Service," *Proc. Thirteenth IEEE Int'l Symposium High Performance Distributed Computing (HPDC-13)*, 2004.

[12] A. Shoshani, A. Sim, and J. Gu, "Storage Resource Managers: Middleware Components for Grid Storage," *Proc. Nineteenth IEEE Symposium on Mass Storage Systems (MSS '02)*, 2002.