



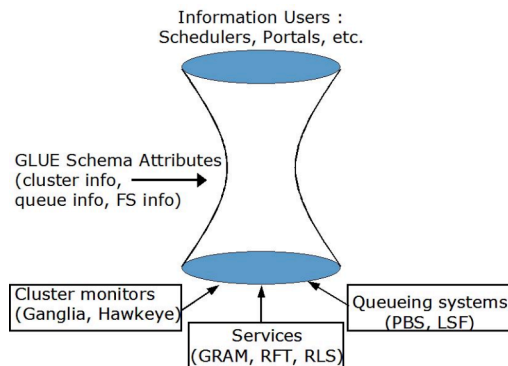
MDS4: The GT4 Monitoring and Discovery System

Contact email: jms@mcs.anl.gov

Contact web: <http://www.globus.org/toolkit/docs/4.0/info>

The Globus Toolkit's Monitoring and Discovery System (MDS) implements a standard Web Services interface to a variety of local monitoring tools and other information sources.

MDS4 is a "protocol hourglass," defining standard protocols for information access and delivery and standard schemas for information representation. Below the neck of the hourglass, MDS4 interfaces to different local information sources, translating their diverse schemas into appropriate XML schema (based on standards such as the GLUE schema whenever possible). Above the neck of the hourglass, various tools and applications can be constructed that take advantage of the uniform Web Services query, subscription, and notification interfaces to those information source that MDS4 implements.

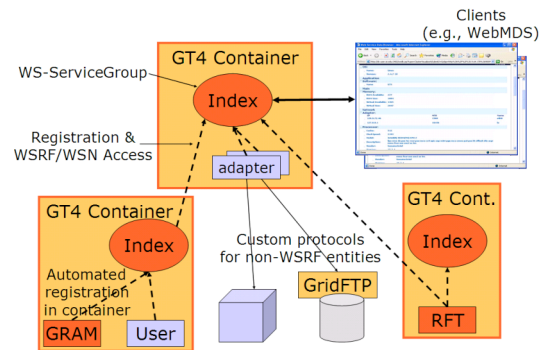


MDS4 builds on query, subscription and notification protocols and interfaces defined by the WS Resource Framework (WSRF) and WS-Notification families for specifications and implemented by the GT4 Web Services Core.

Building on this base, we have implemented a range of *information providers* used to collect information from specific sources. These components often interface to other tools and systems, such as the Ganglia cluster monitor and the PBS and Condor schedulers.

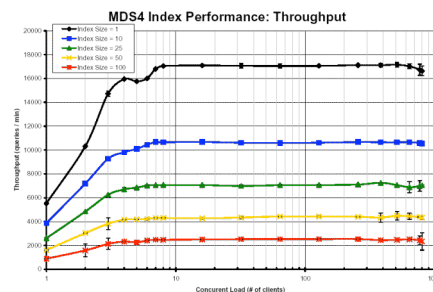
MDS4 also provides two higher-level services: an *Index* service, which collects and publishes aggregated information about information sources, and a *Trigger* service, which collects resource information and performs actions when certain conditions are triggered. These services are built upon a common *Aggregation Framework* infrastructure that provides common interfaces and mechanisms for working with data sources.

Finally, a web-based user interface called *WebMDS* provides a simple XSLT-transform based visual interface to the data. The figure below depicts a typical MDS4 project deployment.



Testing to date of the GT 4.0.1 Index server shows very positive levels of stability and scalability. A long running test for a 100-entry index, where each query returns ~190K of data has been running over 47 days, and has so far processed over 20 million requests, averaging 5 per second, with no noticeable performance or usability degradation.

We have also examined the general scalability of the MDS4 index server with respect to numbers of clients and index size. Using the DiPerf framework, developed at University of Chicago, we have run LAN experiments on the UC TeraGrid cluster (dual 2.4GHz Xeon processors, 3.5 GB RAM, 1GB.s network) examining index sizes of 1, 10, 25, 50 and 100 and clients, distributed evenly over 20 nodes, from 1 to 800. The figure below shows the 8 minute averages of these results.



Additional information on MDS4 can be found at <http://www.globus.org/toolkit/mds>. See also the tech report available from <http://www.mcs.anl.gov/~jms/Pubs/mds4.sc.pdf>

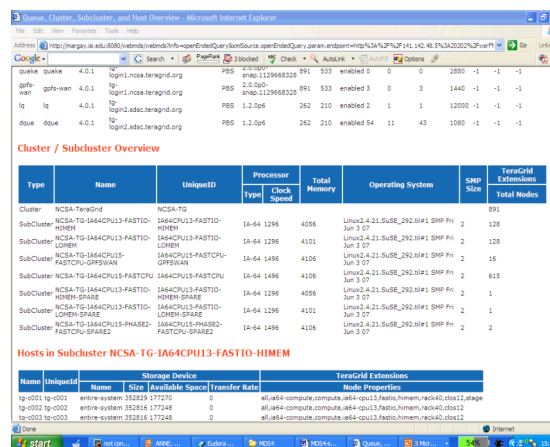
Finding the Right Resource to Use: Work with TeraGrid

As an example of an MDS4 deployment, we have worked with the TeraGrid project to provide data relevant to selecting the “best” set of resources to use for a particular job. End-users (via a web interface), metatasking systems, and other applications will be able to use this deployment to find the resources that best meet their needs.

Information relevant to resource selection is currently available from a variety of sources: cluster monitoring systems, including Ganglia, Clumon and Hawkeye, and work continues on an interface to Nagios; resource management and scheduling systems, including PBS, Torque and LSF; and common services, including GRAM, GridFTP, RFT, CAS, and RLS, all part of GT4.

The end user for this set of data is the Grid scheduling system that will be deployed across TeraGrid. As several candidates are currently under development, we plan to develop a user-friendly web interface to this data so users can better make their own resource selection decisions in the meanwhile.

By providing an hourglass, and therefore a level of indirection, we can easily accommodate additional TeraGrid sites simply by adding information providers (extending the interface at the bottom of the hour glass) and no client-level change (at the top of the hourglass) will be needed.



Type	Name	UniqueID	Processor	Total Memory	Operating System	SWP Size	TeraGrid Extensions
Cluster	NCSA-TeraGrid	NCSA-TG					
SubCluster	NCSA-TG-IA64CPU13-FASTIO-HMEM	IA64CPU13-FASTIO-HMEM	IA-64 1296	4096	Linux2.4.21.SuSE_292.SRP Ph	2	128
SubCluster	NCSA-TG-IA64CPU13-FASTIO-LOHNP	IA64CPU13-FASTIO-LOHNP	IA-64 1296	4101	Linux2.4.21.SuSE_292.SRP Ph	2	128
SubCluster	NCSA-TG-IA64CPU13-FASTIO-GPFSWAN	IA64CPU13-FASTIO-GPFSWAN	IA-64 1496	4106	Linux2.4.21.SuSE_292.SRP Ph	2	16
SubCluster	NCSA-TG-IA64CPU13-FASTIO-LOHNP-GRAB	IA64CPU13-FASTIO-LOHNP-GRAB	IA-64 1296	4096	Linux2.4.21.SuSE_292.SRP Ph	2	1
SubCluster	NCSA-TG-IA64CPU13-FASTIO-LOHNP-GRAB	IA64CPU13-FASTIO-LOHNP-GRAB	IA-64 1296	4101	Linux2.4.21.SuSE_292.SRP Ph	2	1
SubCluster	NCSA-TG-IA64CPU13-FASTIO-LOHNP-GRAB	IA64CPU13-FASTIO-LOHNP-GRAB	IA-64 1496	4106	Linux2.4.21.SuSE_292.SRP Ph	2	2

Name	UniqueID	Storage Device	Transfer Rate	Node Properties
tg-c001	tg-c001	entire-system 352816 177248	0	all-ia64-compute.compute-ia64-cpu13-fatio.hmem.ncsl01.cba22.ange
tg-c002	tg-c002	entire-system 352816 177248	0	all-ia64-compute.compute-ia64-cpu13-fatio.hmem.ncsl04.cba22.ange
tg-c003	tg-c003	entire-system 352816 177248	0	all-ia64-compute.compute-ia64-cpu13-fatio.hmem.ncsl04.cba22.ange

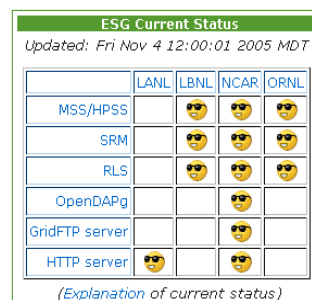
Discovering Errors: Work with the Earth Sciences Grid

The MDS4 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. This functionality was inspired by a similar capacity in Condor’s Hawkeye monitor.

Currently, the Earth Sciences Grid is using the MDS4 Trigger service to monitor the states of integral service components in that grid, such as RLS, SRM, OpenDAP, and HTTP and GridFTP filesystems. The Trigger service periodically checks that the aforementioned services are up and running, and if any one of these services has gone down or is unavailable for any reason, an action script is executed which sends email to administrators notifying them of the service that is unavailable.

The status provided by the Trigger service in ESG is also reflected in the ESG portal web page, where users can view at a glance the current up/down status of the various component services.

See <https://www.earthsystemgrid.org/> to explore the ESG portal page further and view the current system status.



	LANL	LBNL	NCAR	ORNL
MSS/HPSS	😊	😊	😊	😊
SRM	😊	😊	😊	😊
RLS	😊	😊	😊	😊
OpenDAP	😊	😊	😊	😊
GridFTP server	😊	😊	😊	😊
HTTP server	😊	😊	😊	😊

(Explanation of current status)

ESG allows access to data that is stored either on rotating storage (currently at NCAR) or on deep storage (at NCAR MSS, NERSC HPSS and ORNL HPSS). A deep storage archive may occasionally be offline for scheduled maintenance or other administrative task.

The **Storage Resource Manager (SRM)** middleware is used to retrieve data from the deep archives and transfer it (via GridFTP) to the NCAR dataportal, where it is made available to HTTP requests.

The **Replica Location Service (RLS)** is a system of cross-updating databases that keeps track of files copies stored anywhere on the Grid. The RLS is used when publishing data to the ESG system.

The **OpenDAP** server is used when requesting subsets of virtually aggregated datasets. The data is extracted from multiple files, stored into a single file, and made available to HTTP requests.