



IST-2001-32133
GridLab - A Grid Application Toolkit and Testbed

Prototype Tool

Author(s):	Petr Holub, Martin Kuba, Luděk Matyska and Miroslav Ruda
Document Filename:	GridLab-5-D5.6-0004-Tool
Work package:	Testbed
Partner(s):	Masaryk University
Lead Partner:	Masaryk University
Config ID:	GridLab-5-D5.6-0004-1.0
Document classification:	DELIVERABLE

Abstract: This document describes briefly the prototype tools used to monitor the GridLab project testbed.



Contents

1	General overview	2
2	Basic technology	2
2.1	Machine oriented tests	2
2.2	Database of historical records	4
2.3	Tests of GridLab Web services	4
2.4	Matrix tests	4
2.5	Deployment	5
2.6	Test dependencies	5
3	Future development	5
3.1	Integration	5
3.2	Database	5
3.3	Complex testing	5

1 General overview

Running a testbed at or near production level quality is impossible without permanent monitoring of its status. The current implementation of the GridLab testbed status monitoring is based on the *pull* model, where a set of programs — individual tests — is periodically run on a testing server.

The primary collection of tests is *machine oriented*, i.e., services associated with individual machines are tested. Second collection of tests is *GridLab webservices oriented*, i.e. services developed by other GridLab workpackages, implemented as webservices with only single instance in the whole testbed, are tested. Third collection of tests are *matrix tests*, i.e. they test whether a particular service can perform an operation on any combination of machines in the testbed. The machine oriented tests are run in regular hourly intervals, with the possibility of unscheduled “on demand” tests on a particular machine initiated by anyone with a valid certificate (the certificate must be accepted by tested machine).

Results of the machine oriented tests are written to a relational database to keep historical records. A dynamic portlet then allows various views of both historical and latest results.

The GridLab webservices tests run in the same regular hourly intervals as the machine oriented tests, and the latest results are displayed on a dedicated web page. Historical records are kept as a series of web pages. It is also possible to run an unscheduled “on demand” test by anyone with an accepted certificate.

The matrix tests are not run regularly, as they can produce too much load on the tested services. A more efficient way of performing these tests is being developed.

Results of all tests are collected and displayed on the GridLab Administrative portal. The test results are accessible either directly at <http://www.gridlab.org/WorkPackages/wp-5/testbed/status.html> or through appropriate navigation on the GridLab web pages (starting from the GridLab main page <http://www.gridlab.org>).

2 Basic technology

2.1 Machine oriented tests

The GridLab testbed currently runs Globus-2 services, which means that all machines on the testbed must provide the following functionality:

- GRIS — an LDAP server providing information about the machine
- GSI-FTP server
- Gatekeeper — job submission service, providing access to one or more *jobmanagers*; some of these jobmanager can support MPI jobs
- GSI-SSH — service for GSI enabled command like access.

On top of these, a GIIS test is also performed (only one GIIS server per a Grid is required). The basic set of tests was defined as part of the TeraGrid project (see <http://www.ncsa.uiuc.edu/~jbasney/teragrid-setup-test.html>). This is a monolithic Perl script that calls Globus-2 clients to perform individual tests. The Perl script outputs a HTML page with results emphasized with colors: green for successful, yellow for timeouted and red for failed tests.

This approach has several limitations, most notably the monolithic nature of the Perl script (the whole script must be modified to add a new test), the HTML output, which makes a history storage rather complicated, and the unsatisfactory performance. The tests are run sequentially,

which means that the whole test could run for many minutes and its scalability is very restricted. While this could be overcome with a threaded version (which, in its turn, is very CPU intensive), a substantial change in the design was necessary.

The GridLab test design is based on the following principles:

- modular design with pluggable tests
- multilanguage support, for individual tests implemented in any language
- scalability, able to support hundreds of machines/services
- limited parallelism (using thread pools) and strict timing of individual tests (including proper test hang-out treatment)
- Globus independence (adherence to standards)
- writes results to a permanent storage (keeps historical records)
- easily configurable, both with respect to machines, tests and users (their credentials)
- test dependencies (do not run tests that must fail).

The testing framework fulfilling all these requirements was developed in Java and is currently used to test the GridLab testbed status. It uses a configuration file which specifies (in XML format) the list of machines to be tested, list of tests to be performed, size of the thread pool and where to get user credentials. The scheduled tests run under service credential (different from any user's credential), the unscheduled run either under the same credential (when initiated by administrator) or under the user's own credential (this is especially important if a problem with particular credential is expected).

While written in Java, the system supports multiple languages—currently shell scripts are used where a remote job must be submitted, otherwise Java is used. We use Java CoG for implementation of Globus clients. Small wrapper programs (in C) take care of timeouts and hang-up jobs, taking also care of all clean-up when a particular test does not finish correctly.

Output from tests is written to a RDBMS, making history extraction an easy job. They are also written in XML. The most recent results are converted into XHTML using a default XSLT stylesheet, to provide a matrix similar to the original TeraGrid one.

We added the following tests on top of the original TeraGrid collection:

- Accepted CAs—checks which Certification Authorities are actually accepted by a site
- gridmapfile—checks whether users have valid entry in the machine gridmapfile (conceptually, checks the authorization permissions).
- Installed software—tests on the presence of software required by the GridLab developers
- MDS extensions—tests whether extensions to standard information services, developed by GridLab's WP-10, are properly installed and configured
- MDS webservice—tests whether a webservice frontend for MDS extensions, which allows to write data into information services, also developed by GridLab's WP-10, is also properly installed and configured
- monitoring—tests whether monitoring software, developed by GridLab's WP-11, is properly installed and running
- MPI tests—several tests, check whether MPI-enabled jobs, written in C or Fortran, can be successfully compiled and run on all jobmanagers

2.2 Database of historical records

All machine oriented tests' results are stored in a database. This allows to disconnect the presentation layer, as the web pages with the test results are generated from the database and independently from the actual tests. This allows an easy deployment of even complex filters and continuous presentation of results (they are stored into the database as they are accepted, so even partial results can be presented). The database can serve as a data source not only for the web/portal presentation, but the results can be made available independently (e.g., through LDAP or SOAP).

Currently the results are displayed in two ways - either as the latest results of all tests on all machines, giving an overall overview of the testbed status, or as history of a particular test on a particular machine, giving overview of its evolution in time. More possible ways of viewing the historical data are planned, like history of all tests for a given machine or history of all machines for a given test.

2.3 Tests of GridLab Web services

With the continuous move towards the OGSA model, the Grid status monitoring cannot be restricted to services associated with particular machines only. Currently, seven web services (Scenario Broker, Adaptive service, Metadata service, Replica Catalog, Data Movement Service, Data Browsing service, Authorization service) are already deployed on the GridLab testbed, and their status is also regularly monitored. All these web services are accessible over the HTTPG (HTTP over GSI) protocol. Each service has a method (function, operation) `getServiceDescription()` which returns a string. For each service, a simple C program, using gSOAP tool, is used to call this method and report on unavailability or unexpected output. These programs are invoked from a regularly dispatched shell script, which also creates a web page using the same color notation as the general status monitoring described above. However, currently the service testing is not integrated into the general monitoring framework.

An alternative implementation in Java (using portlets and GridSphere portal framework developed by GridLab's WP-4 for presentation) was developed for running unscheduled "on demand" tests. The dynamic nature of Java, as opposed to the static C gSOAP tool, enables testing of any webservice, accessible over HTTP or HTTPG protocol, if it has a suitable method (function, operation) to be called.

2.4 Matrix tests

A new set of tests is emerging, which need to test whether a given machine can perform a given operation on a combination of machines. Examples are "Data Movement service" which needs to be able to copy files from any machine to any machine, or "GRMS service" which needs to be able to submit a job to any machine. Results of a single test of a single service provide a matrix, with one, two, or possibly more dimensions.

Some preliminary implementation of the tests show that the number of machine combinations (obviously growing fast with number of machines) to be tested doesn't allow regular (hourly, daily) testing. So a better strategy is being developed.

Also the ways how results of matrix tests are going to be presented to human users are investigated, because even simple case of history records for several tests with two-dimensional results produce a 4-dimensional matrix, which is difficult to present to users in an easily understandable way.

2.5 Deployment

Machine oriented and GridLab webservice tests are run every hour and the most recent results as well as historical records are available on the GridLab Administrative portal. An e-mail message is sent to appropriate administrator when some machine oriented tests fails. Individual tests can also be run unscheduled, on request from a site administrator or a user with a valid credential.

2.6 Test dependencies

There are dependencies among the tests, and it would be wasting of resources to run tests that must fail. For example, there is no point in trying to run an MPI job on a machine, if authorization against its Gatekeeper cannot be made, no jobmanagers are reported by its GRIS or the executable for the job cannot be compiled. So the configuration XML file for machine oriented tests can define dependency of a test on successful results of a set of other tests.

3 Future development

The GridLab testing framework and its implementation provides a valuable tool for the testbed management. However, there are still some deficiencies which we are recently addressing.

3.1 Integration

While currently the web service testing and matrix tests are done independently of the machine testing, the framework is general enough to allow full integration of web (and grid) service testing with the machine oriented tests. With the move towards OGSA (and its Globus-3 implementation), increased collection of functionalities will be provided as grid services, so the current differentiation will become more and more obsolete. The web/grid service tests will be implemented as specific plugins into the GridLab framework which will be extended to deal also with the service discovery, service replication and other advanced properties of grid services.

3.2 Database

We plan to provide access to the database as a Grid service, as well as to allow unscheduled (“on demand”) tests as another Grid service. This way the whole testbed status monitoring will become OGSA compliant.

We expect that results from all Administrative portal initiated tests (either scheduled or unscheduled) will be stored in the database, providing thus more complete data about the actual testbed behavior and status.

3.3 Complex testing

All the tests described above use the *pull* model, with a central site polling all the machines and collecting the results. While simple and efficient (at least for a reasonably large grid), it will be complemented by a testing on the application level. A complex task that uses all the GridLab developed functionality, most notably task migration, notification, ability to dynamically adjust to the available resources, will be regularly deployed on the testbed. It will check the functionalities through their actual use, reporting successes and failures to the monitoring server (the database). It will use the test dependencies to limit the amount of information on failures (it will not report what can be deduced on the server). The major advantage of this approach to

the status testing lies in its ability to test not only individual features and services, but their complex relationships.