
GT 5.0.0 GRAM5: Quality Profile

Table of Contents

1. Test coverage reports	1
2. Code analysis reports	1
3. Outstanding bugs	1
4. Bug Fixes	3
5. GRAM5 Throughput Tests	3
6. GRAM5 Condor-G Tests	5

<titleabbrev>Quality Profile</titleabbrev>

1. Test coverage reports

- [GT 5.0.0 Test Coverage Status Report](#)¹

2. Code analysis reports

- No code analysis reports have been generated at this time.

3. Outstanding bugs

- [Bug 108](#):² Fork perl zombies
- [Bug 106](#):³ Fix test failures with SGE LRM adapter
- [Bug 105](#):⁴ Held Condor jobs should be reported as SUSPENDED
- [Bug 104](#):⁵ globus-job-manager-event-generator loads all historical events the first time run
- [Bug 103](#):⁶ Ease two phase end commit timeout
- [Bug 102](#):⁷ Fix Two Phase Commit Semantics for Failed Jobs
- [Bug 100](#):⁸ one of the RSL parameters is not supported error doesn't indicate which it is
- [Bug 99](#):⁹ Add a high-level diagram for the approach doc
- [Bug 98](#):¹⁰ Add Condor-G doc for using GRAM 2 and 5

¹ ../reports/coverage/index.html

² <http://jira.globus.org/browse/GRAM-108>

³ <http://jira.globus.org/browse/GRAM-106>

⁴ <http://jira.globus.org/browse/GRAM-105>

⁵ <http://jira.globus.org/browse/GRAM-104>

⁶ <http://jira.globus.org/browse/GRAM-103>

⁷ <http://jira.globus.org/browse/GRAM-102>

⁸ <http://jira.globus.org/browse/GRAM-100>

⁹ <http://jira.globus.org/browse/GRAM-99>

¹⁰ <http://jira.globus.org/browse/GRAM-98>

- [Bug 96:](#)¹¹ GRAM-106 SGE LRM mishandles invalid environment definition
- [Bug 95:](#)¹² GRAM-106 SGE LRM doesn't check for executable permissions
- [Bug 94:](#)¹³ GRAM-106 SGE LRM doesn't check for executable existence
- [Bug 93:](#)¹⁴ GRAM-106 SGE LRM script doesn't handle environment vars with whitespace
- [Bug 92:](#)¹⁵ stdout to local file doesn't work if count >1
- [Bug 88:](#)¹⁶ Missed two phase commit causes job to not be destroyed
- [Bug 86:](#)¹⁷ Prioritize script invocations to improve throughput
- [Bug 80:](#)¹⁸ GRAM 5 beta2 release
- [Bug 79:](#)¹⁹ Add support for OSG's "NFS Lite" concept
- [Bug 77:](#)²⁰ GRAM zombie
- [Bug 71:](#)²¹ GRAM protocol test package contains expired test certificate
- [Bug 70:](#)²² globus-job-status acts strange for completed jobs in GRAM5
- [Bug 69:](#)²³ globus-job-get-output -f doesn't work in GRAM5
- [Bug 68:](#)²⁴ Bad error when proxy is too short-lived
- [Bug 54:](#)²⁵ make globus-job-manager-event-generator not require configuration by default
- [Bug 53:](#)²⁶ Generalize log path configuration
- [Bug 51:](#)²⁷ configurable control of number of perl scripts that can run simultaneously
- [Bug 47:](#)²⁸ simplify the throughput tester program and use improved version as doc
- [Bug 24:](#)²⁹ Debug/verbose flags for globusrun, globus-job-run
- [Bug 23:](#)³⁰ Improved error codes and error reporting for users

¹¹ <http://jira.globus.org/browse/GRAM-96>

¹² <http://jira.globus.org/browse/GRAM-95>

¹³ <http://jira.globus.org/browse/GRAM-94>

¹⁴ <http://jira.globus.org/browse/GRAM-93>

¹⁵ <http://jira.globus.org/browse/GRAM-92>

¹⁶ <http://jira.globus.org/browse/GRAM-88>

¹⁷ <http://jira.globus.org/browse/GRAM-86>

¹⁸ <http://jira.globus.org/browse/GRAM-80>

¹⁹ <http://jira.globus.org/browse/GRAM-79>

²⁰ <http://jira.globus.org/browse/GRAM-77>

²¹ <http://jira.globus.org/browse/GRAM-71>

²² <http://jira.globus.org/browse/GRAM-70>

²³ <http://jira.globus.org/browse/GRAM-69>

²⁴ <http://jira.globus.org/browse/GRAM-68>

²⁵ <http://jira.globus.org/browse/GRAM-54>

²⁶ <http://jira.globus.org/browse/GRAM-53>

²⁷ <http://jira.globus.org/browse/GRAM-51>

²⁸ <http://jira.globus.org/browse/GRAM-47>

²⁹ <http://jira.globus.org/browse/GRAM-24>

³⁰ <http://jira.globus.org/browse/GRAM-23>

- [Bug 22:](#)³¹ client connections can't be timed out
- [Bug 15:](#)³² transition from httpg to https
- [Bug 14:](#)³³ increase availability of GRAM in linux distributions
- [Bug 12:](#)³⁴ Gatekeeper's syslog output cannot be controlled
- [Bug 5:](#)³⁵ Add gram-level prologue and epilogue script execution for mpi jobs
- [Bug 4:](#)³⁶ Add support for a "managed fork" service
- [Bug 2:](#)³⁷ Investigate how to setup GRAM5 services in a HA setup

4. Bug Fixes

- [All Fixes:](#)³⁸ This link lists all 67 of the improvements and bug fixes that were done for the GT 5.0.0 release

5. GRAM5 Throughput Tests

5.1. Experiment Hardware and Software Configuration

The following experiments were run on the `nomer.mcs.anl.gov` virtual cluster. The cluster consists of 6 partitions, each having a single Intel(R) Xeon(R) CPU E5430 @ 2.66GHz core, and 2GB RAM. The virtual machines in the cluster each had a single virtual network interface. The cluster was configured as follows:

- 1 node: Master node
- 5 nodes: Test/execution nodes

All nodes ran an **apache2** http server, **gmond** (ganglia monitoring), and **pbs_mom** (torque LRM).

The master node also ran a **globus-gatekeeper**, **globus-gridftp-server**, **globus-job-manager-event-generator**, **gmetad** (Ganglia Meta Daemon), **pbs_sched** (Torque LRM scheduler), **pbs_server** (Torque LRM server), and **nfsd** linux kernel NFSv4 server for the execution nodes.

The test nodes (1 for single-client tests, 5 for multiple client tests) ran the **gram-throughput-tester** program. All 5 test/execution nodes executed the test job executables as scheduled by the LRM.

These tests were done with GT 5.0.0 beta1. The throughput tester was compiled from CVS trunk October 30, 2009.

5.2. Experiment Scenarios

The first set of tests submitted jobs to the GRAM5 service for one hour. After one hour elapsed, the client would terminate any jobs which were being managed by the GRAM5 service. The test client recorded the time of the experiment start, the time of the termination start, and the time after which all jobs had reached a DONE or FAILED state.

³¹ <http://jira.globus.org/browse/GRAM-22>

³² <http://jira.globus.org/browse/GRAM-15>

³³ <http://jira.globus.org/browse/GRAM-14>

³⁴ <http://jira.globus.org/browse/GRAM-12>

³⁵ <http://jira.globus.org/browse/GRAM-5>

³⁶ <http://jira.globus.org/browse/GRAM-4>

³⁷ <http://jira.globus.org/browse/GRAM-2>

³⁸ <http://tinyurl.com/yagefou>

The throughput test client would generate a maximum of 50 simultaneous job requests. For all but the *uncapped* test below, a maximum of 2000 jobs were managed by the job manager at any given time (pending or active). Once the client had submitted 2000 jobs, it would stop submissions until a job completed. A total of 10 execution slots were available for the LRM to schedule jobs into, so many were in the GRAM5 PENDING state during the duration of the test.

The different **gram-throughput-tester** experiments consisted of:

Table 1. GRAM5 Throughput Tester Experiments

Experiment Name	LRM monitoring method	Number of clients	Number of users	Maximum number of simultaneous jobs / client
1-client-poll	POLL	1	1	2000
1-client-seg	SEG	1	1	2000
1-client-seg-uncapped	SEG	1	1	unlimited
5-client-seg	SEG	5	1	2000
5-client-seg-diffusers	SEG	5	5	2000

5.3. Throughput Test Results

The following table contains a summary of the results of these experiments. The columns contain the following information:

Experiment	Experiment name, the same as in the previous section
Total Jobs	The total number of GRAM jobs that were <i>submitted</i> to the GRAM5 service by the throughput tester in one hour.
Termination Tasks After 1 Hour	The total number of jobs that were being managed by the GRAM5 service when the one hour test period elapsed. These jobs were terminated using the GRAM5 cancel protocol message by the throughput tester.
Termination Duration (hh:mm:ss)	The amount of time it took for the termination tasks to complete and all jobs to reach the DONE or FAILED state.
Master Node Max 1 min. Load Average	The maximum value of the one-minute load average on the master node, that is, the node running the GRAM5 and Torque service.
Master Node Average 1 min. Load Average	The average value of the one-minute load average on the master node over the duration of the test.
Errors	Description of any errors which occurred on the client side that prevented operations from completing as expected.

Table 2. GRAM5 Throughput Tester Results Summary

Experiment	Total Jobs	Termination Tasks After 1 Hour	Termination Duration (hh:mm:ss)	Master Node Max 1 min. Load Average	Master Node Average 1 min. Load Average	Errors
1-client-poll	2110	2000	00:14:18	11.46	7.96	None
1-client-seg	2110	2000	00:10:36	2.82	0.93	None
1-client-seg-uncapped	6664	6584	00:42:46	3.26	2.75	None
5-client-seg	6800	6434	00:57:20	3.19	2.49	Connection refused during termination
5-client-seg-diffusers	7226	6720	00:45:41	3.79	3.13	None

6. GRAM5 Condor-G Tests

6.1. Experiment Hardware and Software Configuration

The following experiments were run on the `nomer.mcs.anl.gov` virtual cluster. The cluster consists of 6 partitions, each having a single Intel(R) Xeon(R) CPU E5430 @ 2.66GHz core, and 2GB RAM. The virtual machines in the cluster each had a single virtual network interface. The cluster was configured as follows:

- 1 node: Master node
- 1 nodes: Test/execution nodes
- 4 nodes: execution nodes

All nodes ran an **apache2** http server, **gmond** (ganglia monitoring), and **pbs_mom** (torque LRM).

The master node also ran a **globus-gatekeeper**, **globus-gridftp-server**, **globus-job-manager-event-generator**, **gmetad** (Ganglia Meta Daemon), **pbs_sched** (Torque LRM scheduler), **pbs_server** (Torque LRM server), and **nfsd** linux kernel NFSv4 server for the execution nodes.

The test/execution node ran the condor-g daemons. The condor job classified ad included attributes to submit the jobs to the GRAM5 service running on the service node. The tests were done with Condor version 7.4.1. The Condor-G configuration parameters in the `condor_config` file were as follows:

Figure 1. Condor-G Experiment Configuration

```

GRIDMANAGER_MAX_PENDING_SUBMITS_PER_RESOURCE=50
GRIDMANAGER_MAX_SUBMITTED_JOBS_PER_RESOURCE=2000
GRIDMANAGER_MAX_PENDING_REQUESTS=50
GRIDMANAGER_JOB_PROBE_INTERVAL=300
GRIDMANAGER_MAX_JOBMANAGERS_PER_RESOURCE=0
ENABLE_GRID_MONITOR=FALSE
GRIDMANAGER_DEBUG= D_FULLLDEBUG
GRIDMANAGER_GLOBUS_COMMIT_TIMEOUT=12000

```

The execution nodes executed the test job executables as scheduled by the LRM.

For this test, the test/execution node and the execution nodes were configured to run up to 20 job processes each simultaneously.

6.2. Experiment Scenario

This test submitted a 2000 job condor job cluster, using the following classified ad:

Figure 2. Condor-G Classified Ad

```
Universe=grid
grid_resource = gt5 nomer1.mcs.anl.gov:2119/jobmanager-pbs
executable=/bin/sleep
arguments=300
transfer_executable=False
stream_output = False
stream_error = False
output = test.out.$(Process)
error = test.err.$(Process)
log = test.log
notification=Never
queue 2000
```

The configuration parameters are similar to the GRAM5 tests described in GRAM5 Throughput Tests section. The key difference being that this test runs until all 2000 jobs have completed and does not submit any jobs after the maximum of 2000 has been reached.

To provide a point of comparison, another test using similar parameters was run using the **gram-throughput-tester** program in place of Condor-G. Note that the Condor-G service provides file staging and a scratch directory beyond what the throughput tester job did.

The two experiments consist of:

Table 3. GRAM5 Condor-G Experiments

Experiment Name	LRM monitoring method	Number of clients	Number of users	Total number of jobs
Condor-G	SEG	1	1	2000
Throughput Tester	SEG	1	1	2000

6.3. Condor-G Test Results

The following table contains a summary of the results of these experiments. The columns contain the following information:

Experiment	Experiment name, the same as in the previous section
Time to Submit 2000 Jobs	The total number of GRAM jobs that were <i>submitted</i> to the GRAM5 service by the throughput tester in one hour.

Time For All Jobs To Complete	The amount of time it took for all 2000 jobs to complete. The theoretical minimum value for this is 50 minutes if all 2000 jobs were submitted instantaneously and there was no overhead for them to be deployed to the 200 execution nodes.
LRM Submit Rate (Jobs/Minute)	The total number of jobs that were being managed by the GRAM5 service when the one hour test period elapsed. These jobs were terminated using the GRAM5 cancel protocol message by the throughput tester.
Master Node Max 1 min. Load Average	The maximum value of the one-minute load average on the master node, that is, the node running the GRAM5 and Torque service.
Master Node Average 1 min. Load Average	The average value of the one-minute load average on the master node over the duration of the test.

Table 4. GRAM5 Throughput Tester Results Summary

Experiment	Time to Submit 2000 Jobs (hh:mm:ss)	Time For All Jobs To Complete (hh:mm:ss)	LRM Submit Rate (Jobs/Minute)	Master Node Max 1 min. Load Average	Master Node Average 1 min. Load Average
Condor-G	00:32:34	02:00:56	94	6.56	1.64
Throughput Tester	00:17:58	01:54:49	111	2.43	0.53