GRIDportal Test Report

Martin Matusiak

22nd November 2005

Contents

0.1 Preface

This document is intended for the reader who is familiar with GRIDportal. For an explanation of concepts and the system itself, please see the White Paper.

1 What to test?

1.1 The frontend perspective

In the frontend perspective, GRIDportal interacts with WebKit, Webware and Apache. The level of complexity in this construct, as regards GRIDportal, is rather low. It is a question of correct configuration of the http server and the consequence is that either it works or it does not. Thus testing the frontend is fairly trivial, once the http server has been configured, one can tell whether GRIDportal operates correctly.

Consequently, no testing was done of the frontend beyond verifying that it does serve pages correctly.

1.2 The backend perspective

Given that GRIDportal is a component of a greater whole, it would make sense to consider how it should be tested and exactly what should undergo testing. GRIDportal is closely linked to the NorduGrid/ARC middleware, it depends entirely on services supplied by the middleware. Thus there are two apparent perspectives to testing.

- Integration test to determine how well GRIDportal works as part of the software stack.
- Isolation test where the deployment environment is largely disregarded to test the functionality of GRIDportal alone.

An integration test makes a lot of sense, because the purpose of the project is to integrated GRIDportal into the runtime environment and verify that it is functional. An integration test will also uncover stability issues with the middleware and possibly other components further down the software stack. This can be considered beneficial, as it brings out faults in the entire system and not just GRIDportal. However, it can also be seen to hamper the testing, as stability issues in the middleware or elsewhere could conceal problems with GRIDportal.

To get around that issue, an isolation test could be used to determine GRIDportal's stability and performance without relying on the middleware or other software. However, such a test would have to be far more complicated and probably would not uncover many serious errors which were not already discovered in the integration test. Fine grained isolation testing could be accomplished with unit testing, but the way GRIDportal was designed, it simply does not lend itself to unit testing very well, and choosing that avenue would imply simulating parts of the middleware. The reason why this is so is that GRIDportal does not contain much business logic, it is a gateway between the user and the middleware. So it consists mostly of functions which either

- interact with the middleware, or
- return html code,

thus writing unit tests would soon become very obscure and tedious. Consequently, it was decided to scrap the isolation test entirely and only undertake integration testing.

2 Test procedures

2.1 Integration test

2.1.1 Automated testing

No facility was created to automate the integration test. Such a system could be constructed, it would have to contain a web browser and logic to verify that navigating the GRIDportal web site a certain way returns expected results. But this kind of system would most likely take a long time to build, it would also be terribly inconvenient to maintain as GRIDportal evolves and the html pages retrieved by the test system change slightly in appearance.

2.1.2 Manual testing

So in the absence of automated testing, the apparent alternative is to do manual testing. The manual testing follows familiar use cases and experiences GRIDportal just as a typical user would. It first sets certain criteria for input data or outcome, and then proceeds to test the full functionality of GRIDportal. (As navigation in GRIDportal is handled through buttons tied to html forms, it simply means testing all of the functions operated by these buttons.)

Since GRIDportal is a small website (judging by the number of pages), we have the luxury of testing all of its functionality in this test.

2.1.2.1 Limitations

Special scenarios geared at unveiling uncommon errors which the user is unlikely to encounter are not tested (such as trying inconceivably extreme input values or purposely trying to provoke unexpected situations). This should be tested in the context of an isolation test instead, since an isolation test is more suited to yield a thorough run down of the bugs in each function.

3 Tests results

3.1 Integration test

3.1.1 Typical use cases

BLAST testing was impaired by the lack of installed BLAST databases on the grid, so only blastp and blastn jobs were possible to run. On top of that, only data sets for blastp were available. But given that all blast* variants are very similar in how jobs are executed and all jobs are launched by the same binary, testing one BLAST variant is highly significant to the whole.

In testing Matlab, only a trivial data set was available.

Where no outcome is stated, the test finished without error.

Scenario > Outcome	**Criteria**	**Verdict**
Run Matlab job (implies: create & edit, submit & monitor & get & view files & delete)	trivial data set	PASSED
Run blastp job (implies: create & edit, submit & monitor & get & view files & delete)	typical data set	PASSED
Run blastp job (as above) > job was terminated by grid for exceeding max time limit of 12h	huge data set	PASSED
Run a job > job was deleted by grid	job not retrieved within 24h	PASSED
Kill a job		PASSED
Clean a deleted job		PASSED

3.1.2 Session handling

A few tests were run to verify that session handling is functional.

While it was stated that the testing would not deliberately provoke unexpected results, a user who mistakenly chooses the wrong file to upload as a proxy will receive a surprising outcome.

Scenario > Outcome	**Criteria**	**Verdict**
User logs in > login accepted	valid proxy	PASSED
User logs in > denied access	expired proxy	PASSED
User logs in > WebKit returned unhandled exception	random file uploaded as proxy	FAILED
User inactive > WebKit session times out, user must log in anew	inactivity exceeds 60 minutes	PASSED

3.1.3 HTTP transfer constraints

An inherent weakness of GRIDportal is handling of big input/output files. All file transfers are done over HTTP, thus network speed and congestion can easily disrupt an ongoing transfer and there is no function to resume a transfer so the user is forced to start over and try again.

Connection speeds are given as top extremes, so any transfer is very unlikely to reach the speed indicated.

Scenario > Outcome	**Criteria**	**Verdict**
Create blastp job > transfer failed to complete	input file is 75MB, connection is 96KB/s	FAILED
Download job results > transfer completed	.tar.gz is 150MB, connection is 660KB/s	PASSED