



QCDgrid2 Load and Stress Testing Results

Project Title: QCDgrid2

Document Title: QCDgrid2 Load and Stress Testing Results

Document Identifier: QCDGRID2-STRes-1.0

Document Filename: LoadStressTestingResults.doc

Distribution Classification: Commercial in Confidence

Authorship: Daragh Byrne

Approval List: QCDgrid Development Team

Distribution List: UKQCD Collaboration

Document History:

<i>Personnel</i>	<i>Date</i>	<i>Summary</i>	<i>Version</i>
DJB	24/Feb/05	Completed document	1.0

Contents

1	Introduction	2
1.1	The QCDgrid project	2
2	Status of requirements identified in WP1.2	3
3	Methodology	7
4	Results	7
4.1	Write 75MB Files, One Client	7
4.2	Write 75MB files, Multiple Clients	8
4.3	Write 1MB Files, Multiple Clients	9
4.4	Read 100 x 1MB Files.....	11
4.5	Read 10 x 75 MB Files.....	11
4.6	Read 100 x 1MB Files, Multiple Clients	11
4.7	Write 100 Folders Containing 100 Metadata Files.....	12
4.8	Read Metadata.....	13
4.9	Write 1 Folder Containing 50000 Metadata Files	14
5	Conclusions	14
5.1	Write Data Use Case	14
5.2	Read Data Use Case	16
5.3	Write Metadata Use Case	16
5.4	Read Metadata Use Case	16
5.5	Overall Conclusions	16
6	References	18

1 Introduction

1.1 The QCDgrid project

The UKQCD Collaboration aims to “procure and jointly exploit computing facilities for lattice field theory calculations whose primary aim is to increase the predictive power of the Standard Model of elementary particle interactions through numerical simulation of Quantum Chromodynamics” [5]. Such numerical simulations produce large amounts of data in the form of binary files. The first QCDgrid project developed a software suite running on dedicated hardware (the UKQCD Grid system [3]) that allows UKQCD members to store and share data amongst one another in a straightforward and reliable manner. Metadata in the form of XML files can also be stored on the system. A graphical data browser tool was developed to facilitate ease of use. The software suite also allows users to initiate simulations and post-processing jobs on machines on the grid.

This document is a deliverable of the QCDgrid2 project, Work Package 1.3 Stress Testing [4] and contains a formal description of the load testing and stress testing activities that have been conducted within the scope of the activity (that is Work Packages 1.2 and 1.3). The initial requirements for load and stress testing are documented in [1]. Based on these requirements, a family of client applications have been designed and implemented covering the main, high level usage cases supported by the QCDgrid software:

- Reading files/directories from the QCDgrid data grid;
- Writing files/directories to the QCDgrid data grid;
- Reading metadata from the QCDgrid Metadata Catalogue.
- Writing metadata to the QCDgrid Metadata Catalogue.

The client applications have been committed to the NeSCForge project repository [2] for future use.

As ascertained during requirements capture [1], current usage of the UKQCD Grid is modest and makes it difficult to gather, with confidence, detailed information about the anticipated usage/load that will be imposed on the grid into the future. Despite this the following load testing tasks were generated in [1]:

- L1** Writing files/directories to the QCDgrid data grid at a rate of 1Tb/day should be simulated, with individual files size from 75MB to 500MB.
- L2** Read files/directories from the QCDgrid data grid with 10—20 concurrent users should be simulated.
- L3** Read files/directories from the QCDgrid data grid with a volume of 12×500MB files should be simulated.
- L4** An estimate of the time taken for a file to be available for retrieval from the grid, from the point of submission should be calculated.

In addition to these four tasks, the QCDgrid team have elected to simulate typical failure points within the system, with the aid of the following stress tests:

- S1** The disk capacity on the control node is exhausted.
- S2** Many clients try to access the data grid simultaneously.

In Section 2, we describe the final outcome of each of the requirements defined in [1]. Then, in Section 3 the methodology of the load and stress testing activity is described. Results from the load testing activity, plus observations from the stress testing activity are presented in Section 4. Based on these results and the experiences of the QCDgrid development team during the work package, conclusions and proposals for further work are presented in Section 5.

2 Status of requirements identified in WP1.2

During the requirements gathering activity conducted in WP1.2 [1], seven specific requirements were defined. In this section, we describe the status of each of these requirements at the conclusion of the load and stress testing activity.

R1 *A test grid needs to be created in order to run the load tests without disrupting the running of the production grid.*

In order to test the QCDgrid software, a test grid has been established at EPCC. The nodes on this grid and their specifications are detailed in Table 1.

Machine name	Specification
qcdgrid1	Personal Computer with 1×664 MHz Intel Pentium III processor, 512MB memory, 13G hard drive, and Linux version 2.4.21-15.0.4 O/S.
qcdgrid2	Personal Computer with 1×998MHz Intel Pentium III processor, 512MB memory, 8.1G hard drive, and Linux version 2.4.21-15.0.4.EL O/S.
qcdgrid3	Personal Computer with 1×998MHz Intel Pentium III processor, 382MB memory, 18G hard drive, and Linux version 2.4.21-15.0.4.EL O/S.

Table 1: Summary of the component server nodes in the *QCDgrid Test Grid* environment. Client services are hosted outwith the test grid, on other suitable nodes.

The test grid is dedicated to the QCDgrid project and can be stressed without impacting other users. To avoid a performance hit on the test grid, all client processes that initiate load and stress testing operations are hosted on other machines, at EPCC, installed with the QCDgrid client software. This ensures that no undue load is placed on the grid machines. At the time of writing three Sun Microsystems desktop machines (running the Solaris operation system) have been prescribed as client nodes for the activity. The following table describes the client machines.

Machine name	Specification
e3000	Sun Microsystems Enterprise 3000 with 8 x 400 MHz UltraSPARC III CPU, 8Gb memory and Solaris 9 O/S.
e3500	Sun Microsystems Enterprise 3500 with 4 x 336MHz UltraSPARC III CPU, 1Gb memory and Solaris 9 O/S.
Garnet	Sun Microsystems Sun-Blade 1000 with 1 x UltraSPARC III CPU (unknown clock speed), 1.5 Gb memory and Solaris 9 O/S

Table 2: Summary of the client systems used for the load and stress testing activities.

One should note that the QCDgrid test grid is a considerably less powerful grid than the production UKQCD Grid. This determines that it is necessary to extrapolate the results from the test grid to the production grid, as there are no plans to replicate these test on the production grid directly, except for several particular cases noted in Section 4.

R2 *The test suite should allow the “write data” use case to be run in a flexible manner. The suite should allow the simulation of writing data to the grid in a variety of file sizes, by a variable number of users and over sustained periods of time, such as one whole day, one hour, ten minutes etc. The test suite will exercise the 10 Gbyte/day and 1 Tbyte/day estimate for data production in a number of ways, using different combinations of file sizes. The test will also be run above this rate.*

A script called `data-grid-write-test` has been implemented, as described below:

Command:	<code>data-grid-write-test</code>	
Purpose:	Simulates writing to the grid multiple times.	
Arguments:	-n	Mandatory. The number of files to be written to the grid.
	-f	Optional. The size of the files to be written to the grid, in megabytes. Defaults to 75MB.
	-i	Optional. The interval in seconds between requests to submit a file. If set to 0, all requests are spawned in the background in one go. If set to a non-zero number, the script will pause for that number of seconds between issuing requests. If this flag is not present, the script will wait for the previous request to terminate before issuing the next one.
	-b	Optional. If this flag is present, the requests are to be submitted as background jobs. This facilitates the simulation of multiple clients on a single machine.

This script is written in Perl, as are all the test scripts. This script makes repeated calls to QCDgrid command `put-file-on-qcdgrid`. The files placed on the grid by this script use a directory style filename, analogous to those used on the production UKQCD Grid. Each

file has a unique identifier, followed by the date in the format YYYY-MM-DD-HH-MM, followed by the file's size. An example identifier would be 24i5jfm945mmf91c/2005-01-12-19-34/data75MB.bin. The size of the file is encoded into the filename to facilitate the "read data" load test (see description, below).

An auxiliary script, called `verify-write-successful`, was also written. This script used `qcdgrid-list` to investigate whether the files written to the grid had appeared on it or not.

Timing data was provided by use of Perl's inbuilt `time` command.

R3 *The test suite should allow the "read data" use case to be run in a flexible manner. The suite should again be flexible with respect to file size, number of users, period of time, number of reads in a period of time etc. The test should also examine performance metrics, such as the time it takes to read files of certain sizes. The test will be run to pull across at least 12 configurations of 500MB size, and at higher loads as appropriate.*

An application called `data-grid-read-test` has been created, as described below:

Script name:	<code>data-grid-read-test</code>	
Purpose:	Simulates reading from the data grid multiple times.	
Arguments:	<code>-n</code>	Mandatory. The number of files to attempt to read from the grid.
	<code>-f</code>	Optional. The size of the files to read. An integer giving the size of the files to be read in megabytes. Defaults to 75MB.
	<code>-i</code>	Optional. The amount of time in seconds to wait between read request issues. If set to 0, all requests are spawned in the background in one go. If set to a non-zero number, the script will pause for that number of seconds between issuing requests. If this flag is not present, the script will wait for the previous request to terminate before issuing the next one.
	<code>-b</code>	Optional. If this flag is present, the requests are to be submitted as background jobs. This facilitates the simulation of multiple clients on a single machine.

This application selects files of the appropriate size, based on the information contained in their filenames (as outlined above). This implies that "read data" load testing should be carried out after "write data" load testing.

The `data-grid-read-test` script was not actually used. The Unix `time` command was used to time the execution of the `get-file-from-qcdgrid` executable. The case of reading twelve 500MB configurations was not carried out for reasons of disk space.

- R4** *The test suite should exercise the job submission system. The suite should allow the automated submission of jobs taking a specified amount of time at a given rate, by a certain number of users.*

The job submission system is a very recent addition to the QCDgrid software suite and, at the time of writing, is undergoing field testing with the UKQCD user group. Usage of the software is anticipated to be lightweight in the short to medium term and thus is not immediately affected by load/stress-related issues. Because of this, the QCDgrid development team have determined that this requirement be deprecated for the present time.

- R5** *The test suite should allow the server part of the metadata catalogue to be placed under various loads. This part of the suite should be flexible with respect to number of queries, expected size of data returned by query, rate of query and number of concurrent users. This part of the suite will leverage existing client side code.*

Two applications have been created to simulate read and write operations to the QCDgrid Metadata Catalogue. These applications are called `mdc-read-test` and `mdc-write-test`, respectively, with usage information provided below.

Script name:	<code>mdc-write-test</code>	
Purpose:	Simulates multiple reads from the metadata catalogue.	
Arguments:	<code>-d</code>	Mandatory. The number of directories containing metadata to be sent to the server.
	<code>-n</code>	Mandatory. The number of files to be present in each directory.

`mdc-read-test` creates client-side directories of test XML files. These test files consist of ensemble metadata. The files contain the name of one of the EPCC QCDgrid team members as the value of the

`markovChain/management/archiveHistory/elem/participant/name`

node, to facilitate the read use case. The script is biased so that one particular team member's name is selected about 75% of the time.

Script name:	<code>mdc-read-test</code>	
Purpose:	Simulates multiple writes to the metadata catalogue.	
Arguments:	<code><Name></code>	The name of the person to search for.

This script simply requests that all of the metadata submitted by a given user be read from the Metadata Catalogue.

Together, these two applications can be invoked to simulate different loads and stresses on the Metadata Catalogue.

R6 *A data generation tool for creating files of various sizes needs to be written or located.*

Creation of relevant test data has been integrated into each of the scripts individually.

R7 *The test system should examine how long it takes a piece of data to become available on the grid after being submitted.*

Simulations have been devised and measured that provide a realistic estimate of the time taken for a data file to become available on the grid (from the point at which it is submitted by a user). Details of this simulation and associated results are provided in Section 4.

R8 *A full report on the load and stress testing carried out should be written after testing.*

This document fulfils the requirement.

3 Methodology

Once the test scripts had been implemented, they were run in various combinations on the client machines to simulate a variety of load scenarios. A detailed record of the testing was kept as it was carried out, including command lines issued. Section 4 of this document describes the exact tests that were carried out, and the results of each. Each test is explained in terms of:

- Objective: what the test is expected to investigate or demonstrate;
- Description: the exact actions that were carried out to meet the objective;
- Observations and results: what happened during the test run, including quantitative and qualitative data.

4 Results

The following subsections describe the exact stress tests carried out between 1st – 21st February 2005.

4.1 Write 75MB Files, One Client

Objective

To investigate the performance of the data grid under low load. This test was intended to demonstrate that the data grid can operate at a basic level.

Description

Low load conditions were simulated by writing ten 75MB files to the data grid. This test ran the following command line on e3500:

```
data-grid-write-directory-test.pl -n 10 -s 75
```

This caused the `put-file-on-qcdgrid` executable to run ten times sequentially.

Observations and Results

The test took 478 seconds to run, which means that on average each file took 47.8 seconds to be transferred (a write rate of approximately 1.5MB per second). It should be noted that this figure includes the start-up time for the client executable, which has not been measured.

The verification script was run at various time intervals after the initial test. The following table shows that all the files were definitely present on the grid approximately half an hour after they were submitted, with the majority being present after 20 minutes.

Time after end of test (minutes)	Number of original files available
22	7
25	9
28	10

4.2 Write 75MB files, Multiple Clients

Objective

To investigate the performance of the data grid under moderate loads. This test was intended to show that the data grid can operate under higher than basic loads.

Description

Moderate load conditions were simulated by attempting to write thirty 75MB files to the data grid. Three client machines each executed the following command line simultaneously:

```
data-grid-write-directory-test.pl -n 10 -s 75
```

Observations and Results

This test operated correctly until the hard drive on the machine running the control thread (qcdgrid1) became full. It was noted that the control thread machine had only 3.1Gb available disk capacity, even when the data grid was empty. It is known that the Replica Catalogue occupies significant disk space, even when it is empty.

The consequences of the disk filling up were as follows:

- Only those files that had been transferred before the disk filled up ended up on the grid; requests issued by the clients after the disk filled up were not honoured by the system;
- The control thread process stopped responding to new requests such as qcdgrid-ping and qcdgrid-list.

It did not appear that any data had been lost – it seemed to be the case that any data that made it on to the grid before the server stopped had a replica catalogue entry. Nevertheless, it was decided that a follow-up test was needed to investigate the exact behaviour of the grid when the control node fills up.

In order to prepare for the second part of this test, all data was removed from the grid by calling qcdgrid-delete multiple times.

The following command line was run on each of three client machines:

```
data-grid-write-directory-test.pl -n 20 -s 75
```

This operation attempted to submit 20 files, each of size 75MB, from the three clients simultaneously, making a total of 60 files.

It was observed that the disk on the central node became full yet again, due to the size of the “Inbox”. It appeared that the particular configuration of the test grid favoured the central node as a first destination for all data placed on the grid.

It was observed again that control thread process stopped when the disk became full, and that only the write requests that had been issued at that stage had been carried out. It appeared that the system was losing some data in this instance: the file `mainnodelist.conf`, which contains information about all the machines on the grid, was found to be empty.

4.3 Write 1MB Files, Multiple Clients

Objective

To investigate the performance of the data grid when multiple clients attempt to write data to it, i.e. under high load conditions.

Description

To investigate high loads, initially ten client processes attempted to write 100 files each. The following command line was used to carry out the tests:

```
data-grid-write-directory-test.pl -n 100 -s 1
```

The command line was run three times each on `garnet` and `e3000`, and four times on `e3500`, simultaneously. Prior to running the test, some data was removed from the grid by removing a directory containing 75MB files using `qcdgrid-delete`, in order to free up some space. The available storage on the central node after this process was 2.4 GB.

In order to further investigate the results of this initial test, extra similar tests were run on both the live and production grids as described in the Observations and Results.

Observations and Results

The scripts were started at approximately 12:05pm. The scripts executed successfully on `e3500`. On `garnet` and `e3000` however, they appeared to fail immediately. The error message indicated that `grid-proxy-init` had not been run to initialise a client proxy – the author did not know that this needed to be done on a per-machine basis. This was not noticed for some time. The failing scripts were restarted after `grid-proxy-init` had been run. The time lag between starting the script successfully on `e3500` and doing so on the other machines was about 35 minutes.

The scripts were allowed to run to completion, during which time their progress was occasionally checked on. It was observed that about half of the files to be uploaded were present in the inbox on the central node about two hours and fifteen minutes after the scripts had started. Very little replication had taken place at this point. The following table summarises the total times taken for the client processes to run to completion.

As can be seen, the approximate average time to transfer a single small file was observed to be 6-7 minutes. It was observed that all the files found their permanent home on the grid about 24 hours after the start of the test, or about 12 hours after the end of the test. The system

appears to be stable when operating under many clients, as all the uploaded files made it on to the grid.

	Start Time	Total Run Time (seconds)	Approx. Run Time (hh:mm)	Average Time To Service One File (seconds)
Garnet				
Run 1	12:42pm	41538	11:32	415.8
Run 2	12:42pm	41493	11:32	414.93
Run 3	12:42pm	41442	11:31	414.93
E3000				
Run 1	12:45pm	41425	11:31	414.25
Run 2	12:56pm	40872	11:22	408.72
Run 3	12:57pm	40855	11:21	408.55
E3500				
Run 1	12:08pm	37268	10:21	372.68
Run 2	12:08pm	37281	10:20	372.81
Run 3	12:08pm	37271	10:22	372.71
Run 4	12:08pm	37258	10:22	372.58

Table 3: Time for each of ten clients to execute.

In order to investigate whether the times taken were due to scalability or data transfer rate issues, a single client test was run again. One hundred files of size 1MB were placed on the test grid from e3500. In this case the average time to place a single file on the grid was found to be 17.24 seconds, which is obviously substantially faster (note: this is not the time taken for the file to actually appear as available to read on the grid). It seems clear that the performance of the data grid system is affected when the number of clients rises.

In order to investigate the performance of the production data grid, a similar test was run. Two of the production grid machines (ukgrid0, located at Columbia, and edqcdgrid) each made five attempts to place one hundred 1MB files on the grid (for a total of 1000 files as above). On ukqcd0 the average time to place a file on the grid was 147.19 seconds. On

edqcdgrid, the average time to place a file on the grid was 166.91 seconds. Obviously this is better performance than on the test grid. At this point in time data regarding the performance of the production grid under low loads had not been produced.

A test simulating three users trying to upload one hundred 1MB files was also run. On the test grid, it took an average of 47.84 seconds to transfer a file. All processes were run on a single client machine (e3000). On the production grid, the three client scripts were also run on a single client machine (edqcdgrid), which also hosts the control node. The first run took 4887 seconds, the second run took 6155 seconds and the third run took 6190 seconds. The average time to transfer a single file was therefore 57.44 seconds, which is worse than on the test grid.

4.4 Read 100 x 1MB Files

Objective

To measure the performance when reading files from the data grid under low load conditions. This is intended to demonstrate that the data grid read functionality operates correctly under low loads.

Description

The following command line was used to read 100 x 1 MB files from the grid – the data to be read had previously been inserted into the data grid by the “Write Data” tests, above:

```
get-file-from-qcdgrid -R DIR-2005-2-3-12-56-30 ./data
```

Observations and Results

The command took 12 minutes and 13 seconds to execute, meaning that it took an average of 7.33 seconds to read a 1MB file from the grid.

4.5 Read 10 x 75 MB Files

Objective

To measure the performance of the system when reading files under low load conditions, using a typical file size.

Description

The `get-file-from-qcdgrid` command was used in a manner similar to the preceding test.

Observations and Results

The command took 7 minutes and 55 seconds to execute, meaning that it took an average of 47.5 seconds to read a 75MB file from the grid.

4.6 Read 100 x 1MB Files, Multiple Clients

Objective

To measure the performance of the data grid when multiple clients attempt to read files.

Description

This test simulated nine clients requesting the same directory containing one hundred files of size 1MB each simultaneously. The `get-file-from-qcdgrid -R` command was in a manner similar to the preceding test. The command was run three times on each of garnet, e3000 and e3500.

Observations and Results

The results are displayed in the following table. It can be seen that it took about 40 seconds to read a 1MB file from the grid when it is placed under such a load. This is about six times longer than when the grid is read from by a single client.

	Time to read 100 files (minutes)	Average time per file (seconds)
E3500		
Run 1	67.1	40.2
Run 2	67.2	40.2
Run 3	68.7	40.3
Garnet		
Run 1	67.1	40.2
Run 2	67.3	40.2
Run 3	67.4	40.2
E3000		
Run 1	67.6	40.2
Run 2	68.8	40.9
Run 3	68.6	40.9

Table 4: Time to read one hundred 1MB files under high load conditions.

4.7 Write 100 Folders Containing 100 Metadata Files

Objective

To investigate the performance and response of the metadata catalogue when placed under moderate loads.

Description

This test was then run using the following command line:

```
mdc-write-test -n 100 -d 100
```

This test submitted a total of 10,000 files to the metadata server.

Observations and Results

The data was uploaded to the server without error. The average time taken to submit a directory containing 100 files was 17.71 seconds.

4.8 Read Metadata**Objective**

To investigate the read performance of the eXist database server.

Description

The command line client that is part of the eXist distribution was used to time various XPath queries. This was found to be more convenient than using the test script. As a result of the test outlined in section 4.7, the database mostly contained ensemble metadata files containing the name “George Beckett” as their creator (about 75% of approximately 10000 files). The names “Daragh Byrne”, “James Perry” and “Ratna Abrol” were each present approximately equally in the remaining files. A small number of files (less than 10) contained the name “Chris Maynard” as their creator. This enabled queries with large, small and very small result sets to be run. The queries were run on `qcdgrid1`.

The queries had the following format:

```
/markovChain/management/archiveHistory/element/participant/name=[string(.)="%Name%"]
```

Observations and Results

Running the query with the name parameter equal to “Chris Maynard” found 8 results (matching XML documents) in 10.11 seconds.

Running the query with the name parameter equal to “James Perry” (a small expected result set) resulted in an out of memory condition on `qcdgrid1`. Increasing the amount of memory available to the eXist server (by increasing the value of the `db-connections/@free_mem_min` parameter in the server configuration from 5 to 15) allowed the query to be run successfully. The query ran in 10.905 seconds and returned 1241 results. When the same query was run using two copies of the client, the execution time was about 40 seconds for each client.

The query was run again with the name parameter equal to “George Becket” (a large expected result set). The query returned 6411 results in 14.818 seconds. When the query was run using two copies of the client, the execution time was about 51 seconds for each client. When the query was run using three clients, one encountered an out-of-memory condition and the other two took about 77 seconds to run.

4.9 Write 1 Folder Containing 50000 Metadata Files

Objective

To examine the response of the metadata catalogue when large numbers of files are written to it.

Description

The following command line was run

```
mdc-write-test -d 1 -n 50000
```

Observations and Results

This test was set to run overnight. The following morning the test script was still running. The server also appeared to be running. This test might have been expected to complete in 2.5 hours, given that it took approximately 0.5 hours to load 10,000 documents to the database.

In order to investigate how many files had been written to the server at this time, the eXist command line client was started. A command was issued to return all of the documents in the server and count them. This command failed with an out-of-memory warning message.

A second attempt was made to estimate the number of files loaded to the server at this point. eXist stores information about the stored documents in a file called `collections.dbx`. The following command was issued to count the references to XML files within this file:

```
strings collections.dbx | grep .xml | wc -l
```

From the result of running this command it was observed that 43,350 files were present in the database. There were about 10,000 files in the database before the start of the test, meaning that about 33,350 files had been loaded at this point.

In order to investigate whether files were still being loaded at this point, the above command was run after a 20 minute wait. There were still only 43,350 files in the database, meaning that the server had either stopped servicing requests, or was servicing requests very slowly.

5 Conclusions

5.1 Write Data Use Case

Performance figures for some typical write operations were collected. Typically, a 75MB file could be written to the grid in less than a minute when a single client was used. A 1MB file could be written in less than 10 seconds.

These figures were dramatically increased when multiple clients tried to write to the grid at the same time, increasing to about 40 seconds to write a 1MB file. Similar but less severe effects were noticed on the production grid system. Further investigation would be required to ascertain the cause of the slowdown. Possible reasons include:

- A processing bottleneck formed at the central node, as for this particular configuration of the grid the central node seemed to be serving as the only “inbox” as well as performing its co-ordination and replication duties. This is the most likely reason;
- The central node software itself fails to scale to multiple clients;

- Globus components on the central node fail to scale to multiple clients.

It should be noted that, while performance appeared to degrade as the system load scaled up, reliability was maintained, as all files were successfully placed on the grid.

Under moderate loads (3 clients), the production grid was found to perform slightly worse than the test grid, which was not expected, as the production grid had been seen to perform significantly better than the test grid at high loads. A possible explanation is that the production grid was being used at the time the test was carried out, placing it under further load. Further testing would be required to verify that this was the case.

Possible solutions to the scalability problem include:

- Ensuring that clients attribute a low preference to use the control node, achieved by appropriate configuration of the `nodeprefs.conf` file, to reduce the chances that data will be placed on it;
- Limiting the number of users (for example, three or less) allowed to write data to the grid at any one time – this may be possible because, as noted in [1], writing data to the data grid is a relatively infrequent operation;
- Investigating changes to the architecture of the control thread and the client software that might enhance its performance when writing data occurs. Such enhancements as caching grid status information on the client side, asynchronous data transfer and adding performance monitoring to the node location decision algorithms could help in this situation.

The behaviour of the system under low disk space on the central node was also investigated. This situation has occurred in the production data grid before, and similar results have been noticed. The control thread has been observed to stop when disk space on the central node runs out. Excepting the contents of `mainnodelist.conf`, which are easily replaceable, no catastrophic data loss occurs in this situation – the replica catalogue appears to be reliable and does not appear to have lost data once restarted. There is a possibility that clients may think that data has been uploaded to the data grid when, in fact, it has been rejected due to the control node being down.

The following action for dealing with the low disk scenario is recommended:

- Ensure that the control thread is run on a machine whose system partition will not fill up. In practise, this has been ensured by mounting the data storage directories on the central node on a separate partition to the system partition, and by ensuring that the system partition is not used for any purpose that might fill it;
- Ensure that client tools alert users that their data may not have been written to the grid;
- Alter the control node software to send an email warning to the grid administrator when it notices that disk space is low.

It is possible that this issue will be investigated further during a later stage of the project.

5.2 Read Data Use Case

It was observed that 1MB files could be read in less than 10 seconds, and 75 MB files in less than 50 seconds. When placed under a load of nine clients, the time to read a 1 MB file was observed to be about 40 seconds, or a factor of about six greater.

It appears that load affects the “Read Data” use case much less than the “Write Data” use case for this particular configuration of the grid. The most likely explanation for this is the fact that the data is replicated across several machines. In this case, the role of the central node is simply to present the client with details of the machine from which to download the data. The client then interacts with one of a number of machines, which reduces the overall load on the central node.

5.3 Write Metadata Use Case

It was observed that the metadata catalogue dealt with low data volumes well. Writing ten thousand pieces of metadata did not appear to present a problem, with an average time to write one hundred files of 17.71 seconds.

When attempts were made to write large amounts of data, problems occurred. It appeared that the server failed after a certain number of requests had been serviced (about 33,350). The root cause for this failure is unknown, but is most likely to be related to the amount of memory consumed by the server.

It is unclear whether the server will need to deal with such heavy loads in production use, since metadata will generally be read more than it is written. Should the observed problems become an issue, further investigation into appropriate eXist configuration should be carried out. In any event, it is recommended that no more than 10,000 metadata files be loaded at one time.

5.4 Read Metadata Use Case

It appears that the tested configuration of the eXist server has difficulties servicing a number of simultaneous queries. This would cause problems for the operation of the production grid if the volume of requests were to become in any way large, since performance degradation and request failures have been observed when as few as three simultaneous queries have been serviced. However, one should be aware that the test was not entirely analogous to the way the real grid system operates, as the queries were carried out using the relatively heavyweight eXist client rather than via the Web application. Further investigation is required to ascertain the extent of and solution to this problem.

5.5 Overall Conclusions

This report documents our observations of the QCDgrid software under a variety of loads and stresses, designed to simulate the current and predicted future usage patterns for the UKQCD Grid infrastructure. Overall, the observations have confirmed that the QCDgrid software is reliable and predictable even when under high stress. Several issues have been raised by the activity:

- scalability of data grid when multiple users attempt to write data to it;
- behaviour of the control thread in the situation that the local disk capacity is exhausted;
- reliability of the eXist database under high load.

These issues have been noted and will be monitored/addressed as appropriate, within the future activities of the project.

6 References

- [1] D.J. Byrne, *Work Package 1.2: QCDgrid Stress Testing Requirements*, QCDgrid Project Deliverable (November 2004).
- [2] NeSCForge, *The Quantum Chromodynamics Grid Project*, QCDgrid Project Software Repository. On-line resource at <http://forge.nesc.ac.uk/projects/qcdgrid/>.
- [3] QCDGrid: Probing the building blocks of matter with the power of the Grid, QCDgrid Project Homepage available on-line at <http://www.gridpp.ac.uk/qcdgrid/>.
- [4] UKQCD Collaboration, GridPP Project Logbook, available on-line at: http://www.gridpp.ac.uk/qcdgrid/documents/UKQCD_LogBook.pdf (December 2004).
- [5] UKQCD Collaboration, Project Homepage available on-line at <http://www.ph.ed.ac.uk/ukqcd/>.
- [6] Wikipedia definition of load testing http://en.wikipedia.org/wiki/Load_testing
- [7] Wikipedia definition of stress testing http://en.wikipedia.org/wiki/Stress_testing