GRID RELIABILITY: A STUDY OF FAILURES ON THE EGEE INFRASTRUCTURE[*]

Kyriacos Neocleous and Marios D. Dikaiakos
Department of Computer Science, University of Cyprus,
1678 Nicosia, Cyprus
{kyriacos,mdd}@cs.ucy.ac.cy

Paraskevi Fragopoulou and Evangelos Markatos
Institute of Computer Science, Foundation of Research and Technology-Hellas,
1385 Herakleion, Greece
{fragopou,markatos}@ics.forth.gr

Abstract The emergence of Grid infrastructures like EGEE has enabled the deployment of large-scale computational experiments that address challenging scientific problems in various fields. However, to realize their full potential, Grid infrastructures need to achieve a higher degree of dependability, i.e., they need to improve the ratio of Grid-job requests that complete successfully in the presence of Grid-component failures. To achieve this, however, we need to determine, analyze and classify the causes of job failures on Grids. In this paper we study the reasons behind Grid job failures in the context of EGEE, the largest Grid infrastructure currently in operation. We present points of failure in a Grid that affect the execution of jobs, and describe error types and contributing factors. We discuss various information sources that provide users and administrators with indications about failures, and assess their usefulness based on error information accuracy and completeness. Finally, we discuss two case studies, describing failures that occurred on a production site of EGEE and the troubleshooting process for each case.

Keywords: grid, reliability, dependability, EGEE, failure management

[*]This research work was supported in part by the European Commission under projects EGEE (Contract IST-2003-508833) and the Network of Excellence CoreGRID (Contract IST-2002-004265) of the Sixth Framework Programme of the European Union.

1. Introduction

Computing Grids are usually very large scale services that enable the sharing of heterogeneous resources (hardware and software) over an open network such as the Internet. A Grid is organised in Virtual Organisations (VOs) [11], collections of computational and storage resources, application software, as well as individuals (end-users) that usually have a common research area. Access to Grid resources is provided to VO members through the Grid middleware, which exposes high-level programming and communication functionalities to application programmers and end-users, enforcing some level of resource virtualisation [4]. VO membership and service brokerage is regulated by *access and usage policies* agreed among the infrastructure operators, the resource providers, and the resourse consumers.

Jobs submitted by users to a grid system sometimes fail to complete successfully and, in several occasions, the responsibility lies with the user to detect the failure of a job and resubmit it. We would like to examine in what extend[1] can this responsibility be moved from the end user to the grid system, with the ultimate goal of proposing a general solution that will satisfy all grid users, without posing performance penalties, nor resulting to increased CPU-time consumption. It is however difficult to address the problem of grid reliability without first determining, analysing and classifying the causes of failures. In this paper we study the reasons behind grid job failures, for gaining an insight on the most fruitful direction towards building a more reliable grid infrastructure.

In section 2 we present grid error types and contributing factors, while in section 3 we present the existing error information sources of the EGEE project that we have identified, and assess how useful they are, based on error information accuracy and completeness. To make things clearer, we felt it was necessary to include a section on actual grid infrastructure errors that occurred on the University of Cyprus EGEE production site, with an accompanied analysis and troubleshooting process; section 4 describes two such cases. We close by drawing some conclusions and proposing ways to improve the system.

2. Error types and contributing factors

This section briefly describes the factors that can disrupt a job from *starting* or *completing* execution:

(a) Hardware faults: if the job is running on the specified machine at the time of a hardware error, e.g. a machine crashes and has to be restarted, a hard drive burns etc, the job cannot be recovered and has to be resubmitted.

[1]with respect to the end-to-end argument as it applies to grid job reliability [3]

(b) O/S misconfiguration: this relates mainly to operating system services that are not properly configured. An example is implementing firewall changes which can lead to closing ports that are needed for inter-node communication.

(c) Network access disruption/misconfiguration: a factor that can lead to job failure (more accurately in this case, leading to CPU time being wasted), is a site losing Internet access. A user waiting too long to retrieve the job output may resubmit the job, rendering the previous one useless (redundant).

(d) Security breaches/attacks: Takeover of grid nodes by an unauthorized user can result to corruption of job data, job termination etc. Such attacks are usually related to security holes of the operating system and grid middleware, weak root passwords and inappropriate firewall configuration.

(e) Middleware bugs/misconfiguration: attempts to correct problems or perform updates on a grid site can lead to job failure or a more general service disruption. This relates to grid site administrator errors, as well as to bugs in new releases of the middleware that introduce unwanted configuration. According to [10], a large number of service disruption occurrences is the result of a regular performance or security software upgrade that leads to configuration errors.

(f) User mistakes: such failures can result from (i) JDL file problems, for example the user may include an erroneous specification of job requirements that will result in the job failing to start; (ii) user software can cause errors during job execution leading to the job being terminated abnormally; and (iii) problems with user certificate proxies attached to the job, most commonly the absence of a valid proxy during submission, as well as the expiration of an originally valid proxy while the job is running.

3. Error information sources for EGEE

The following main sources can be used to retrieve information about errors on the EGEE testbed:

A. Site Functional Tests (SFTs) report web site
B. Grid Statistics (GStat) web site
C. GGUS and EGEE-SEE ticketing systems
D. CIC broadcasts and GOC entries for site downtime
E. Machine logs, diagnostic commands output, and databases

These sources are analysed below, accompanied by an assessment of the usefulness of each one of these sources, based on the accuracy and completeness of the error information provided. For a better understanding of the various levels on which monitoring is done, the reader can consult Figure 1.

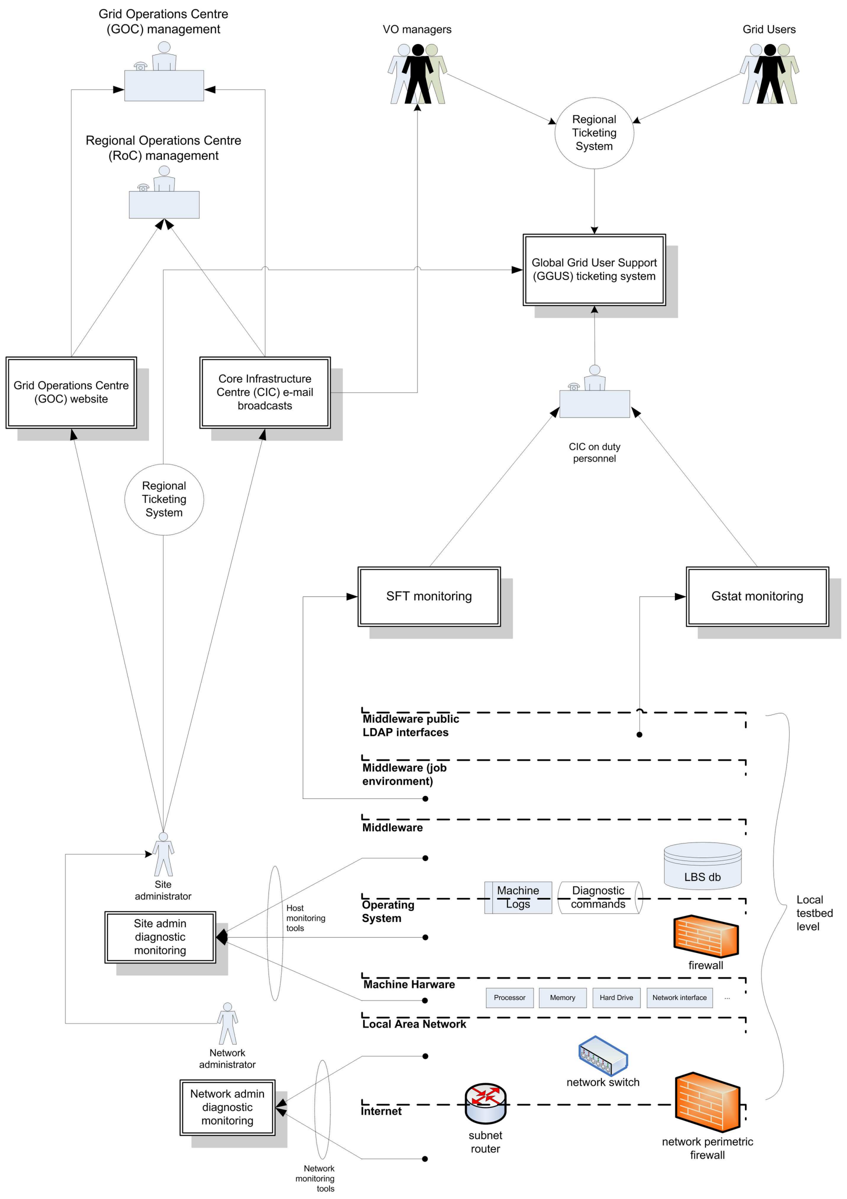
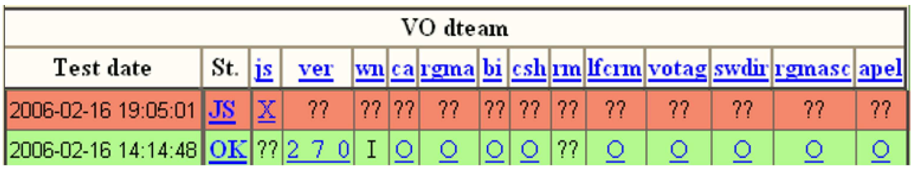

Grid Operations Centre (GOC) management
VO managers
Grid Users
Regional Operations Centre (RoC) management
Regional Ticketing System
Global Grid User Support (GGUS) ticketing system
Grid Operations Centre (GOC) website
Core Infrastructure Centre (CIC) e-mail broadcasts
CIC on duty personnel
Regional Ticketing System
SFT monitoring
Gstat monitoring
Middleware public LDAP interfaces
Middleware (job environment)
Middleware
Site administrator
Site admin diagnostic monitoring
Host monitoring tools
Operating System
Machine Logs
Diagnostic commands
LBS db
firewall
Local testbed level
Machine Harware
Processor
Memory
Hard Drive
Network interface
...
Local Area Network
Network administrator
network switch
Network admin diagnostic monitoring
Internet
subnet router
network perimetric firewall
Network monitoring tools

Figure 1. Error information sources: levels of monitoring



VO dteam														
Test date	St.	js	ver	wn	ca	rgma	bi	csh	rm	lfcrm	votag	swdir	rgmasc	apel
2006-02-16 19:05:01	JS	X	??	??	??	??	??	??	??	??	??	??	??	??
2006-02-16 14:14:48	OK	??	2 7 0	I	O	O	O	O	??	O	O	O	O	O

Figure 2. SFTs for site CY01 (University of Cyprus)

A. Site Functional Tests (SFTs) report web site. EGEE maintains a central "reporting web site" [6] (restricted access) for publishing test-job results for all sites of the infrastructure, primarily serving grid managers and administrators. From there, authenticated users can further access detailed reports for each site. SFT pages show the results of tests performed automatically every 3 hours, and the results of extra tests submitted by the administrators of Resource Centres (RCs) or responsible Regional Operations Centre (ROC) managers and administrators.

Test jobs are short jobs designed to check the health of the various grid components of a site. This testing is done using the DTEAM Virtual Organisation, which exists mainly for running such internal tests on the entire infrastructure. It is worth noting here that DTEAM jobs are typically short, around 10 minutes of CPU time, unlike production-VO jobs that usually take several hours to complete. An SFT consists of several subtests (see head of table in Figure 2); for example 'ver' checks the middleware version, and 'ca' checks the version of the Certification Authority RPMs (Linux software packages) installed. For a short description of each subtest executed during an SFT see [1].

In Figure 2 we provide an example of an SFT report for EGEE site CY01 (University of Cyprus). The red entry (dark highlight) indicates an error, and the X indicates which component of the site has failed (in this case a Job Submission error). Clicking on the X hyperlink displays detailed job submission information, easing the troubleshooting process for site administrators.

B. Grid Statistics (GStat) web site. GStat is a Grid Information System monitoring application [2]. One GStat page exists for each EGEE Resource Centre and these pages are public. From the main GStat web page[2] one can navigate to see the detailed pages that exist for every EGEE site.

The most interesting point here is the graphs provided by GStat, showing error (alert) levels and various other metrics, usually going as far back as the last 12 months. From these graphs one can examine the stability of a site, and possibly how long an error lasted. Other points of interest in GStat pages are total and

[2]http://goc.grid.sinica.edu.tw/gstat/

per-VO CPU and job statistics, storage space reporting, as well as estimated and actual response time for each supported VO. All this information is given in the form of graphs except the latest values which are given as numbers.

It is also worth mentioning here the SmokePing network latency monitoring tool [7]. The website[3] maintained by ICS-FORTH provides network monitoring for the entire SEE federation. These metrics give additional insight to site administrators and they are particularly useful when combined with GStat measurements or SFT results, in order to narrow down the set of components that may be responsible for a failure.

C. GGUS and SEE ticketing systems. The third error information source under evaluation consists of the Global Grid User Support (GGUS) and South East Europe (SEE) ticketing systems. GGUS is the top-level ticketing system for EGEE, while 2nd-level ticketing systems exist for each federation of the project, such as the SEE ticketing system dedicated to the South East Europe federation. The ticketing systems in EGEE are very similar to the ticketing systems used in other organisations to efficiently manage tasks and requests.

As far as grid operational support is concerned, the ticketing systems are mainly used to report component failures as well as needed updates for sites. GGUS tickets are typically opened because of an error that appears in the SFT (Site Functional Test) reports or the GStat monitoring website; such tickets are opened by on-duty Core Infrastructure Centre (CIC) personnel.

D. CIC broadcasts and GOC entries for site downtime. Site managers are required to broadcast information related to site downtime events through the Core Infrastructure Centre (CIC) web site; this information is subsequently e-mailed to all affected parties. CIC e-mails often contain information related to the error that caused the site manager to set the site in maintenance mode; at other times, downtime events are associated with performance or security upgrades and are not related to errors. Site managers must also declare downtime in the Grid Operations Centre (GOC) website, in order to place the site's status in *maintenance mode* instead of *production*. Such entries are typically one short phrase, e.g. "CE hard drive burned," and also contain the start and end dates and times of the downtime event.

E. Machine logs, diagnostic commands output, and databases. The last error information source we will review consists of data found on the nodes of a grid site: (1) the machine logs, such as `/var/log/messages`, `/var/log/globus-gatekeeper.log`, (2) the output of various diagnostic commands executed on the machines that are involved in the error (while the error persists), such as `ps aux`, `diagnose -j`, `checkjob -v <jobID>`, and (3) the Logging and Book-keeping Service (LBS) database records found on the Resource Broker (RB), which can reveal detailed error information spanning

[3]http://mon.egee-see.org/cgi-bin/smokeping.cgi?target=World.Europe.SEE

many sites of many different countries, usually an entire region.

In the remaining of this section we will assess the various sources of information about grid errors discussed this far:

Site Functional Tests reporting and GStat monitoring: From our experience the SFT reports are usually accurate in indicating site problems. The only drawback is that production jobs run for much longer than test jobs, and this may cause some errors to escape the SFT testing; also, the frequency of the SFTs may not be as high as needed to catch all errors. For this reason we can also combine some monitoring information from GStat, although this is not easy to do automatically because the graphs are in image format and the data used to generate the graphs is not readily accessible.

Ticketing systems: if we rely on this information source, we should restrict the information we get from tickets, for example to get only the date the ticket was opened, the time it took for it to be resolved, number of persons involved for solving the problem, and the problem category (replica manager failures, job submission failures, R-GMA tests, etc).

GOC and CIC downtime: Data is easy to gather and easy to separate between "downtime due to errors" and "downtime due to standard maintenance tasks". However, these sources may present important drawbacks: the most important one is that the site manager or administrator publishing this information may be covering up for other types of failures. Furthermore, some downtime may not be announced due to negligence or lack of motivation, since more downtime will be accounted for that site (on some occassions, short failures may pass unnoticed). This source is possibly both inaccurate and incomplete.

Machine logs, diagnostic commands output, and databases: The last category of error information sources is to be found at the lowest levels of the grid nodes themselves (refer to Figure 1, layers *Middleware, Operating System,* and *Machine Hardware*). The machine logs and the LBS database do not rely on human intervention for their production, and we can therefore consider them the most accurate and complete error information sources from the ones examined here. Processing these logs automatically needs some extra work, possibly writing the right parser for each log type, since they are in non-standard formats. On the other hand, obtaining diagnostic command output of real value is trickier, since this will only be of use if it is obtained at the right time, i.e. while the error persists and perhaps even before a subsequent change of the machine state that can hide the initial error information.

For the error analysis, based on the above observations, we propose to rely primarily on the information obtained from grid nodes, but also try to associate this information with entries found on other relevant sources. This could provide a more complete view of errors, and enable us to discard some cases from the analysis if we find contradicting data between different sources.

4. Case studies

In this section we will present some of the more interesting case studies involving error detection, analysis and correction. These studies were conducted between December 2005 and February 2006 on the University of Cyprus EGEE grid site (CY01), in parallel to standard maintenance operations. The analysis was aided by diagnostic commands and log information, and the output of the diagnostic commands was recorded while the failures persisted.

Case study 1: DTEAM VO jobs queued indefinitely. As mentioned in section 3, the DTEAM VO is used for testing the EGEE infrastructure. Standard test jobs are automatically submitted every three hours to all EGEE sites, and the results are published on a web site (the Site Functional Tests report - SFTs). Other DTEAM jobs can also be submitted manually by site administrators, for running non-standard tests on the infrastructure.

At one point, there was a series of DTEAM jobs queued on site CY01 that for some reason (unknown at the time) failed to start. This was a mixture of SFT-related jobs and DTEAM jobs coming from other sites of the federation that were testing a grid service. By the time this was noticed by CY01 site administrators, the number of jobs had reached 30 (the normal is usually 1 to 2 such jobs), and they were all in status ‘queued’, while there were enough free resources to execute 4 of them immediately. This caused several SFT entries to fail.

This problem persisted for several days, and we had to deal with it at first by manually forcing the queued jobs to run on idle processors. An example of manually starting a job, given a processor (node) and job ID, is shown below:

```
[root@ce101 root]# runjob -f -n wn101.grid.ucy.ac.cy 78075
job '78075' started on 1 proc
```

It was then discovered that the problem was caused by an erroneous job scheduler and resource manager configuration (site administrators’ responsibility). These components are somewhat complicated, and their configuration non-intuitive, especially in the case of Maui [5, 12], the middleware component responsible for handling job scheduling on a large number of EGEE sites. The need for reconfiguring Maui and the underlying resource manager (Torque [9]) became evident, but this involved more than a few hours of work, so we had to make a quick workaround to fix the problem, mimicking our actions of manually starting queued jobs: this was a fairly simple script that read the output of the job queue every 4 hours, and detected which queued jobs belonged to DTEAM; the script was then forcing jobs to start execution (whenever possible, based on the free resources), while logging the output of the force-run command (`runjob -f`). Part of the log can be seen below:

```
Sun Feb 12 16:26:12 EET 2006, Attempting to start queued dteam jobs:
job ‘79105’ started on 1 proc
```

After a few days of tuning Maui and Torque, such a problem rarely appears but when it does, the workaround still handles it successfully. DTEAM jobs are still likely to be queued (not indefinitely but for several hours) for various other reasons; by monitoring our site over an extended period of time, we observed that DTEAM VO members may at any point submit a large number of jobs on the site, and the result is that the job scheduler avoids starting some of them as a result of the fair share policy implemented (i.e. the ‘maximum running jobs’ limit set for testjobs is exceeded and Maui does not allow more DTEAM jobs to run before others terminate).

To sum up, the main cause of the problem here was the lack of proper resource manager and job scheduler configuration. The symptoms were treated first due to the urgency of the matter, while the subsequent fine-tuning of the resource manager and the job scheduler configuration addressed the root cause of the problem.

Case study 2: Active Worker Node dies. In this case study, a Worker Node crashed while a job was running on it, causing the job to be completely lost and also creating a second problem with resource allocation. In the following output obtained from `qstat`[4], notice production job of the LHCb experiment [8], with ID 74896.ce101. It appears to be running (Status [S] = Running [R]).

```
[root@ce101 root]# qstat
Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
73933.ce101        STDIN            atlas004              0 Q atlas
74896.ce101        STDIN            lhcb002        00:33:58 R lhcb
```

However, the output of `diagnose -j` (Maui job scheduler command) shows a problem with this job (notice the last two lines):

```
[root@ce101 root]# diagnose -j
...
74896               Running DEF    1 DEF  3:00:00:00 1    1  lhcb002     lhcb
...
WARNING:  active job ‘74896’ has inactive node wn107.grid.ucy.ac.cy
allocated for 1:18:17:03 (node state: ‘Down’)
```

After checking to see what was the problem with worker node wn107, we could not connect to the machine remotely nor ping; the machine was also inaccessible from its console. The WN had crashed due to hard drive overheating.

[4]command of PBS Torque resource manager used to show the status of batch jobs

After restarting the failed node, the job appeared to be exiting (Status = E) from the queue but this state persisted for several minutes. This can be seen below from the new output of `qstat`:

```
[root@ce101 root]# qstat
Job id           Name             User             Time Use S Queue
---------------- ---------------- ---------------- -------- - -----
73933.ce101        STDIN            atlas004              0 Q atlas
74896.ce101        STDIN            lhcb002        00:33:58 E lhcb
```

The output of `diagnose -j` erroneously showed that the job was running normally. The only difference from the previous message is that the warning on 'wn107 being down' was no longer present, which means that the job scheduler was updated with the information that the WN was on-line again, and the server daemon (pbs_server) of the resource manager on the CE could connect to the torque client (pbs_mom) on the restarted WN. This job had to be killed manually, since the data from it being executed on wn107 had been lost when the machine died, and it was certain that the job could not recover. The new output of `diagnose -j` (after restarting the failed WN) can be seen below:

```
[root@ce101 root]# diagnose -j
...
74896                Running DEF    1 DEF  3:00:00:00 1    1  lhcb002     lhcb
-  1:18:33:49   [NONE] [NONE] [NONE]    >=0    >=0    NC0    [lhcb:1] [NONE]
```

Another related problem was that the worker node that had crashed was later reserved and could not be utilized for a fresh job, despite the fact that its CPUs were idle. This can be seen from the output of other Maui commands, such as `showres -n` and `checkjob <jobID>` (the latter is shown next):

```
[root@ce101 root]# checkjob 74896

State: Running
Creds:  user:lhcb002  group:lhcb  class:lhcb  qos:DEFAULT
WallTime: 1:18:42:46 of 3:00:00:00
...
NodeCount: 1
Allocated Nodes:
[wn107.grid.ucy.ac.cy:1]
```

The job stayed in the queue with 'exiting' status, despite the attempts to delete it (`qdel`), suspend it (`mjobctl -s`), and similar modifications with Torque and Maui commands. All such attempts failed because the job was at a state that could not accept modifications. Next, the reservation that was made on wn107 was removed manually using Maui command `releaseres <jobID>`, so at least the node was free to serve another job.

```
[root@ce101 root]# releaseres 74896
released Job reservation '74896'
```

The job was later removed from the queue (after spending more than several hours in 'exiting' status, even persisting after a restart following a middleware upgrade) by manually deleting the resource manager job-specific files from the Grid Gate (node ce101, `/var/spool/pbs/server_priv/jobs/74896*`), and restarting the resource manager.

To sum up case study 2, the problem originated due to a middleware bug that did not allow the job scheduler to detect that one of the worker nodes had crashed and a job was lost, so manual modifications by the site administrator were necessary in order to clear the failed job and allow the restarted worker node to be utilised by new jobs.

5. Conclusions and Future Work

Detecting and managing failures is an important step toward the goal of a dependable grid. The experiences described in this paper show that manual failure management in large-scale infrastructures such as EGEE is a tedious and cumbersome process, due to the complexity, the scale and the multi-institutional span of Grid infrastructures. Furthermore, it can be asserted that current middleware systems do not provide adequate support for handling failures and for supporting Grid dependability. Therefore, we need to develop tools that will support system administrators and end-users to identify failures of Grid components and to investigate their root causes. These tools should provide a higher-level representation of failures, integrating information from the variety of error-information sources presented earlier. Furthermore, they should ease the troubleshooting process undergone by grid system administrators by automating diagnostic and corrective functions, and helping them cope with the complexity of error-information provided by underlying monitoring systems through proper abstractions and uniform user-interfaces. Also, we need to develop systems and algorithms for processing the information collected by the various failure-information sources in order to support the automatic identification and prediction of failures, in order to improve the dependability of the Grid's operation.

Acknowledgements

The authors wish to thank Chryssis Georgiou, Demetris Zeinalipour-Yazti and George Tsouloupas for their helpful comments and suggestions, Nicolas Jacq for insights on the results of the WISDOM data challenge concerning grid reliability, as well as Fabrizio Pacini and Zdenek Salvet for clarifications on the internals of the Workload Management System of EGEE.

References

[1] *Description of Site Functional Tests*. http://lcg-testzone-reports.web.cern.ch/lcg-testzone-reports/sftestcases.html (accessed February 2006).

[2] *Grid Statistics (GStat) description*. http://goc.grid.sinica.edu.tw/gstat/filter_help.html (accessed June 2006).

[3] D. Thain and M. Livny. Building Reliable Clients and Services. In *Grid 2: Blueprint for a New Computing Infrastructure* (I. Foster and C. Kesselman, eds.), Elsevier, Morgan Kaufmann, 2004.

[4] M. Xu, Z. Hu, W. Long and W. Liu. Service Virtualization: Infrastructure and Applications. In *Grid 2: Blueprint for a New Computing Infrastructure* (I. Foster and C. Kesselman, eds.), Elsevier, Morgan Kaufmann, 2004.

[5] *Maui Administrator's Guide*. http://www.clusterresources.com/products/maui/docs/mauiadmin.pdf (accessed May 2006).

[6] *Site Functional Tests for EGEE sites*. https://lcg-sft.cern.ch/sft/lastreport.cgi (accessed June 2006).

[7] *SmokePing network latency measurement tool*. http://oss.oetiker.ch/smokeping/ (accessed June 2006).

[8] *The Large Hadron Collider beauty experiment*, homepage. http://lhcb.web.cern.ch/lhcb/ (accessed June 2006).

[9] *Torque Administrator's Manual*. http://www.clusterresources.com/torquedocs21/ (accessed May 2006).

[10] Aaron Brown. Coping with human error in IT systems. In *ACM Queue magazine*, http://www.acmqueue.com, November 2004.

[11] I. Foster, C. Kesselman and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In *International J. Supercomputer Applications*, Volume 15, number 3, pages 200-222, 2001.

[12] Sophie Lemaitre et al. *Maui Cookbook*. http://grid-deployment.web.cern.ch/grid-deployment/documentation/Maui-Cookbook.pdf (accessed May 2006).