European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

Failure Management in Grids: The Case of the EGEE Infrastructure

Kyriacos Neocleous *Marios D. Dikaiakos*
[kyriacos,mdd]@cs.ucy.ac.cy
Department of Computer Science
University of Cyprus
1678 Nicosia, Cyprus

Vivi Fragopoulou *Evangelos Markatos*
[fragopou,markatos]@ics.forth.gr
Institute of Computer Science
Foundation of Research and Technology-Hellas
1385 Herakleion, Greece

CoreGRID Technical Report
Number TR-0055
July 20, 2006

Institute on System Architecture

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Failure Management in Grids: The Case of the EGEE Infrastructure

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

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

CoreGRID TR-0055
July 20, 2006

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.

1 Introduction

Recent experimental studies have shown that jobs submitted by users to large-scale, multi-institutional Grid infrastructures often fail to complete successfully. For example, data collected and analysed by the WISDOM project [15], which submits tens of thousands of jobs to the EGEE infrastructure [2] in the context of a drug-design effort [13], indicate that only the 65% of submitted jobs executed successfully. In the case of a Grid-job failure it is up to the end-user or the Grid administrator to detect the failure, to identify its cause, to re-submit the job, and to try to fix the problem(s) that caused the failure. Detecting and managing failures is an important step toward the goal of a dependable Grid. This is an extremely complex task, however, as it currently relies on ad-hoc monitoring and manual intervention. Automating this task seems difficult due to intrinsic characteristics of the Grid environment: Grids are not administered centrally and, therefore, it is hard to access the remote sites in order to monitor failures; also, Grid systems are complex and extremely large; thus, it is difficult to acquire and analyze failure feedback.

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 Progamme of the European Union.

In our work, we investigate the reasons behind Grid-job failures, in order to gain an insight on how to build a more reliable Grid infrastructure. We concentrate on the problem of Grid reliability by focusing on jobs that fail to complete successfully, either providing no output or providing incorrect output. This study is conducted in order to unmask the root causes of such failures within the Grid system. We use the term *Grid reliability* as an indication of the extend to which Grid components behave in the way expected by their peers (Grid clients and services). In general, a reliable system is by no means an error-free system, failures would undoubtedly still occur. A reliable system must anticipate and be able to handle failures in various ways, such as by failure *detection, masking, tolerance,* and *recovery*. The handling of failures is particularly complex in large distributed environments such as the Grid, since a lot of components are involved and some may fail while others continue to function properly [20]. A reliable Grid system should fail in predictable ways; if a component fails, the rest of the system must be able to adapt to the changed conditions (such as the lack of a service that has crashed, or the erroneous/incoherent information provided by a faulty service) and maintain an acceptable state; if that is impossible, it should at least be able to recover from the failure and return to the last known correct state.

In the next section, we provide an overview of the architecture, the job-execution model, and the operations of the large-scale Grid infrastructure of EGEE [2]. EGEE is currently the largest Grid infrastructure in operation, comprising 180 sites worldwide with more than 30,000 CPUs, 5PB of storage, supporting over 80 Virtual Organizations. In Section 3, we present possible points of failure in EGEE and describe grid error types and contributing factors. In Section 4, we discuss various services of the EGEE infrastructure that provide error information about failures of Grid components, and assess their usefulness based on error information accuracy and completeness. Finally, in Section 5, we provide two short case studies, describing failures that occurred on the University of Cyprus production site of EGEE (CY01), with an accompanied analysis and troubleshooting process for each case. We close by drawing some conclusions and discussing future work.

2 Grid Computing and EGEE

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) [21], 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 [25]. VO membership and service brokerage is regulated by *access and usage policies* agreed among the infrastructure operators, the resource providers, and the resourse consumers.

The European project Enabling Grids for E-sciencE (EGEE) currently supports the largest grid infrastructure in the world, with more than 180 participating sites. EGEE uses the LHC Computational Grid (LCG) middleware [5], while the new generation gLite middleware [3] is already being deployed at several sites.

Within EGEE there exist several Virtual Organisations (VOs), as for example the Computational Chemistry VO. Users registered within a specific VO obtain credentials for single grid sign-on [16] that enables them to have access to the entire set of resources within (belonging to) that particular VO, despite the fact that such resources span different grid sites across different countries.

Users have access to a User Interface (UI) node for submitting jobs to the Grid, for requesting job status and resources information, and for obtaining the output from completed jobs. In brief, a grid job is usually a set of input files (the *input sandbox*) and an executable that processes the given input on a set of grid resources, according to the user requirements set forth in the Job Description Language (JDL) file that accompanies every grid job submission. The Job Description Language (JDL) is a user-oriented language for describing jobs [17] and the information obtained from a JDL file is taken into account by the grid Workload Management System (WMS) [23] components in order to schedule and submit a job. A job can have particular user-defined requirements for the resources it needs, such as computational capacity, physical memory capacity, the proximity (network latency-wise) of certain files that will be used as input, and the availability of specific application software. Grid jobs can be classified as CPU-intensive and data-intensive, depending on the type of work performed.

Jobs are submitted from the UI to a Resource Broker (RB), a central (global) grid service. The RB is a component of the distributed Workload Management System (WMS) of a grid infrastructure [24] which performs *matchmaking* by identifying a set of resources that satisfy the job requirements. The matchmaking is done based on data received by querying an Information Index, another central service that complements the WMS by providing up-to-date infor-

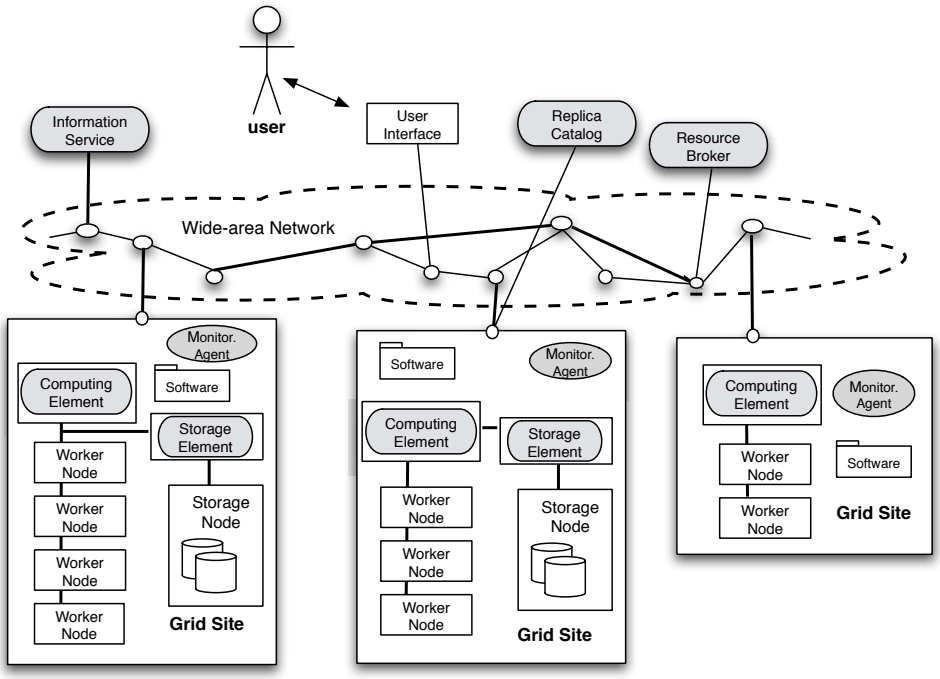


Figure 1: Grid architecture

mation about the state of grid resources, usually spanning several sites.

If the matchmaking is successful, the job is sent from the RB to the the matching Computing Element (CE) for execution. A Computing Element is at site level and it is comprised of the Grid Gate node and several Worker Nodes (WNs). The services running on the Grid Gate node are primarily responsible for authenticating users, accepting jobs, and performing resource management and job scheduling (the last two services comprise the *batch system*). The Worker Nodes are usually powerful machines in terms of processing power and memory capacity, and are responsible for executing jobs arriving at the site, as dictated by the batch system on the Grid Gate. If a job successfully completes execution, the result is then sent back to the Resource Broker and the user is able to access it from there using the User Interface. A UML diagram depicting the life cycle of a typical Grid job can be seen in Figure 2.

During job scheduling and execution, if any input files are necessary, they are either sent by the user during submission (included in the *input sandbox*), or they are already resident on a Storage Element (SE) and the user needs only to specify their location. This brings us to the central Data Management services: the Replica Catalog holds information about the location of various replicas of a file held at the Storage Elements of various sites, and the File Transfer Service is responsible for replicating files across different Storage Elements that are close to Computing Elements, as needed by various jobs.

In general, the *output sandbox* contains the result of a job after it has run on a CE, and contains a set of files that were specified by the user (e.g. a file that contains what would be the output of the console if a job was running on the user's computer). The entire set of output files from a completed job can either be transferred onto the RB (as part of the *output sandbox*) that the user will collect using the User Interface. Alternatively, the output files can be saved onto a Storage Element and registered with the Replica Catalog so the user can access them in the future (most probably these will be very large files of intermediate results that will serve as input to another job).

For more efficient project management, EGEE is divided into different federations/regions, and into each such federation resides a Regional Operations Centre (ROC) that is responsible for supporting and monitoring a set of EGEE-participating grid sites, the Resource Centres (RCs). These divisions into federations are usually organised geographically. As an example, the South East Europe (SEE) federation has the Regional Operations Centre based in Greece; Greece also has 10 participating production sites (RCs), and more production sites under SEE exist in

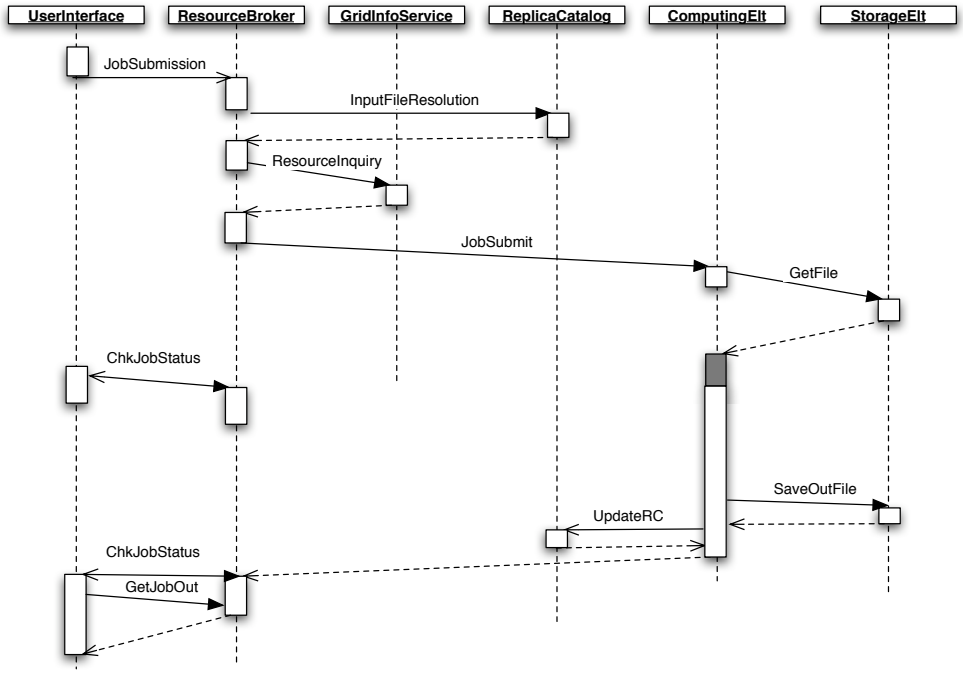


Figure 2: Life-cycle of a typical Grid job

Bulgaria, Cyprus, Israel, Romania, Serbia, and Turkey. Apart from ROCs and RCs there is also the EGEE-wide Grid Operations Centre (GOC), responsible for coordinating and monitoring the operation of the Grid infrastructure, and a total of four Core Infrastructure Centres (CICs) which provide monitoring and operational troubleshooting services, acting as second-level support to ROCs. It is also worth mentioning that usually there is one Certification Authority (CA) for every participating country (any country that contributes resources to the project). The CA is responsible for issuing X.509 certificates for grid users, hosts, and services. There are also optional Registration Authorities (RAs) at each site so the CA can delegate some of the management functions [16]. Figure 3 depicts the EGEE testbed administration hierarchy as described here.

3 Error types and contributing factors

This section describes possible failures that can occur on the Grid and can prevent a job from terminating successfully, or prevent the user from obtaining the output of a successfully completed job.

Examples of possible points of failure on a grid system:

(a) the Resource Broker, the grid node that is used to find an appropriate set of resources (at a grid site) to execute the job. This node holds the user's input sandbox when the job has been submitted, and also the output sandbox after the job has terminated, until the user retrieves this output (or until the sandbox expires, depending on how the system was configured). A failure at this point can result in delays for users to retrieve job status information or their job output, or even corruption or permanent loss of job output.

(b) part of the Computing Element of the site selected for job execution. The Computing Element consists of the Grid Gate node - often called CE, meaning 'CE head' - and the Worker Nodes, which are also CE nodes (see Figure 4 for an example). Points of failure here are the Grid Gate and the specific Worker Nodes that have been allocated for the job. It is worth mentioning here that Grid Gate unrecoverable failures are rare and we have not witnessed any on our site during the two-year lifespan of EGEE-I; on the occasions that the Grid Gate crashed or had to be restarted due

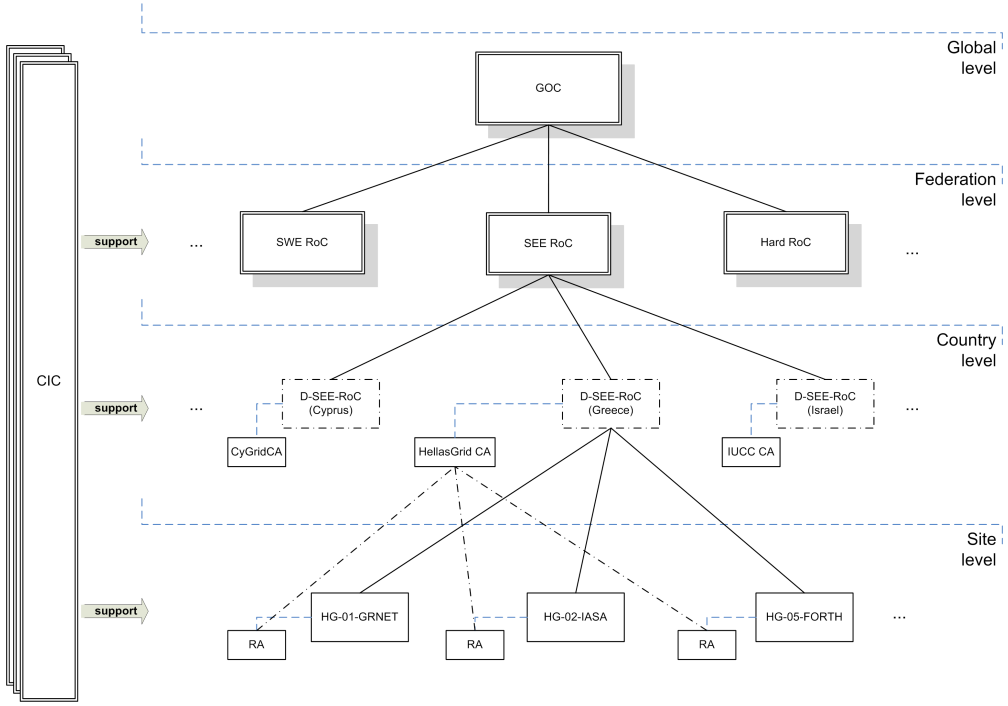


Figure 3: EGEE testbed administration hierarchy

to heavy load or abnormal CPU or hard drive temperatures, no job was lost as we noted upon restarting the machine (it recovered all job information from the WNs and started new *job manager* processes for each job). In contrast, WN failures are normally unrecoverable and resubmission is needed.

(c) the Storage Element (SE) holding input files that are necessary for the job is another point of failure, in the cases that the particular SE crashes, has no network access, or its filesystem is corrupted. In such an event, if there are no replicas (copies) of the necessary files on different SEs and the user has no local copies to upload, we can talk about an unrecoverable failure; however it is usually the case that if a file is only on one SE, it has probably been generated by a job and the user can resubmit that job to generate it again. Still, this wastes CPU and end-user time.

Factors that can disrupt a job from *starting* or *completing* execution:

- **Hardware faults:** if the job is running on the specified machine at the time of an *unrecoverable* hardware error, e.g. a hard drive burns (most common), RAM or motherboard failures, power supply failure, etc.
- **O/S misconfiguration:** this relates mainly to operating system services that are not properly configured. One common example is implementing firewall changes on a site. This can lead to closing ports that are needed for site inter-node communication, or blocking site hosts from communicating with each other altogether. The Grid Gate may lose communication with a worker node (WN) running a user job, or a WN may lose communication with the site Storage Element (or with another site's Storage Element) and be unable to make necessary data transfers. A closely related issue is the inability to run MPI [8] jobs when 'inter-workernode' communication is blocked.
- **Network access disruption/misconfiguration:** a factor that can lead to job failure (or more accurately in this case, leading to CPU time being wasted), is a site losing Internet access. The firewall example mentioned above also applies to network misconfiguration, if the changes are implemented on the network 'perimetric' firewall. A user waiting for too long to retrieve the job output may decide to resubmit the job, rendering the previous one useless when the site recovers from network access failure, if we assume that the previous job was still running while the network was down.

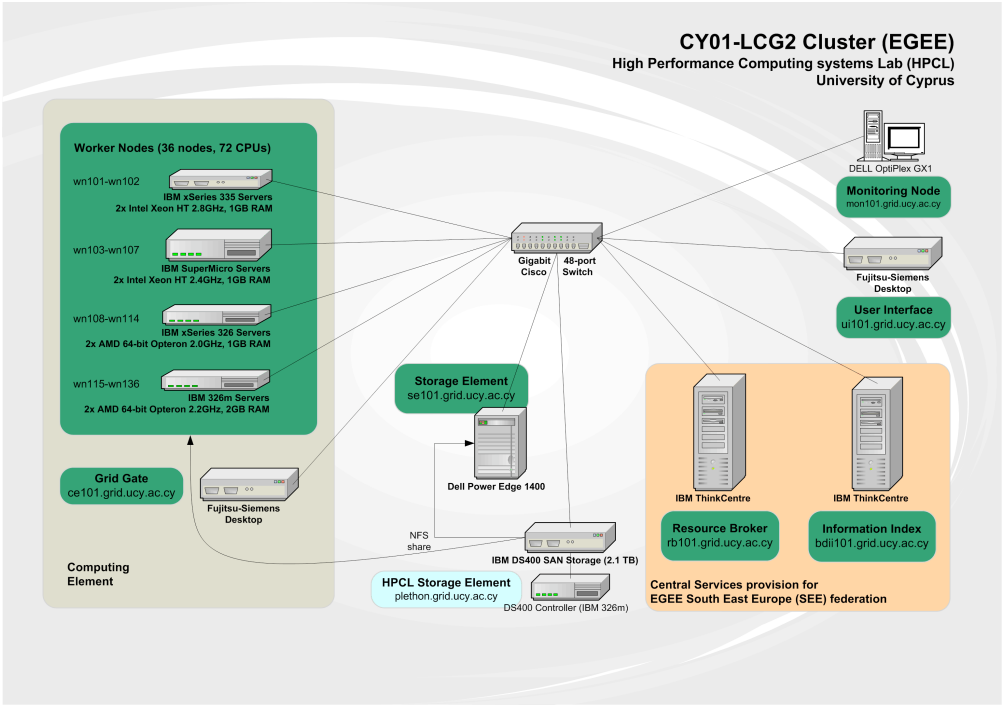


Figure 4: Grid infrastructure for site CY01

- **Security breaches/attacks:** Computing Element, Storage Element or Resource Broker takeover by an unauthorized user (commonly known as a 'cracker') can result to malicious acts like corruption of job data, job termination, sandbox deletion etc. Such attacks are usually related to security holes of the operating system and grid middleware, weak root passwords and inappropriate firewall configuration. Denial of Service (DoS) attacks may also disrupt job completion or temporarily prevent access to job output by cutting a site off the Resource Broker that has delegated the job and is waiting for the output. Note that discussing actual security incidents related to the EGEE infrastructure is not possible under most circumstances, since this could provide potential attackers with useful information.

- **Middleware 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 [18], a large number of service disruption occurrences is the direct result of a regular performance or security software upgrade that leads to configuration errors. Some examples of misconfiguration: setting too short a wallclock time for a job queue, and as a result jobs die before completion[1]; publishing wrong resource data and matchmaking results in accepting a job while no compatible resources exist to satisfy it, causing bandwidth loss and adding overhead to the overall time needed for serving the user; killing the wrong job by issuing a scheduler or resource manager command (as root on the Grid Gate node).

- **Middleware bugs:** grid job failures can result from bugs in middleware code; for instance, failures relating to the grid Workload Management System (WMS), including the components residing on the Resource Broker and the submitting User Interface nodes, observed in [13]. Further information can be given on this particular type of failures once the appropriate experiments are conducted on the EGEE grid infrastructure and the aborted jobs are analysed.

- **User mistakes:** such failures can result from (a) JDL file problems, for example the user may include an inaccurate specification of job requirements that will result in the job failing to start; (b) user software can

[1] this implies altering the queue wallclock time while jobs are running

cause errors during job execution leading to the job being terminated abnormally; and (c) 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. All the cases mentioned here are under user control and have nothing to do with the malfunctioning of other components of the system, so corrective action can only be assumed on part of the user, and not by any form of automatic job resubmission mechanisms.

4 Error information sources for EGEE

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

1. Site Functional Tests (SFTs) report web site.
2. Grid Statistics (GStat) web site.
3. GGUS and EGEE-SEE ticketing systems.
4. CIC broadcasts and GOC entries for site downtime.
5. Machine logs, diagnostic commands output, and databases.

These sources are analysed below, while at the end of this section we make an assessment of the usefulness of each one of these sources, based on the accuracy and completeness of the error information provided.

4.1 Analysis of error information sources

A. Site Functional Tests (SFTs) report web site. EGEE maintains a central "reporting web site" [9] (restricted certificate-based 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 that show the last few entries of the SFT results. 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,[2] 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 5); 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 5 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 there was a Job Submission error, and 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 [4]. One GStat page exists for each EGEE Resource Centre and these pages are open to the public, i.e. no certificate-based access applies here. From the main GStat web page[3] one can navigate to see the detailed pages that exist for every EGEE site.

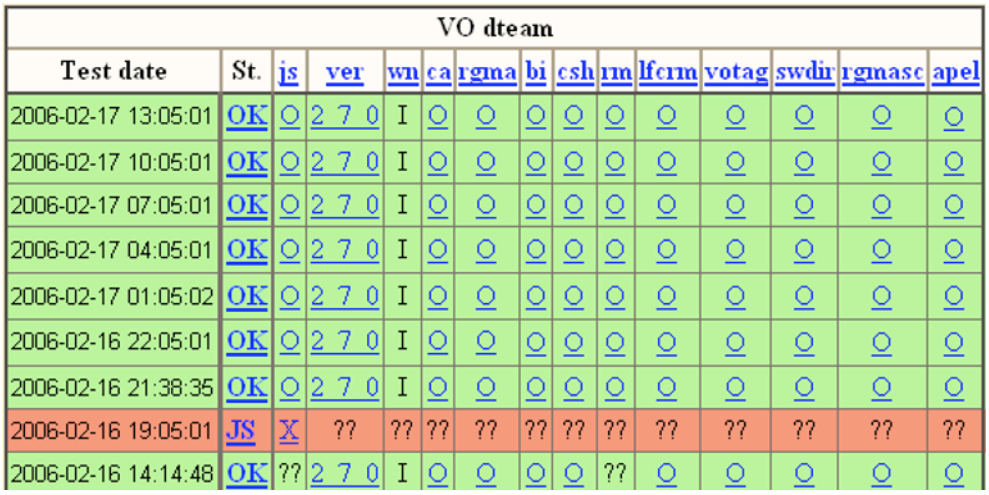
The most interesting point here is the graphs provided by GStat, showing error (alert) levels and various other metrics, usually going as far back as the last 12 months. From these graphs one can examine the stability of a site, and possibly how long an error lasted. The downtime announcements entered through the GOC portal are also listed here[4].

Figure 6 shows the number of local Grid Index Information Server (GIIS) entries that reveal site resources (hardware, services, supported software environments, policies, etc) for site CY01. A site's GIIS normally runs on the Grid

[2] there are plans to replace the DTEAM VO with the OPS (operations) VO

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

[4] see section D. 'CIC broadcasts and GOC entries for site downtime'.



VO dteam														
Test date	St.	js	ver	wn	ca	rgma	bi	csh	rm	lfcrm	votag	swdir	rgmasc	apel
2006-02-17 13:05:01	OK	O	2 7 0	I	O	O	O	O	O	O	O	O	O	O
2006-02-17 10:05:01	OK	O	2 7 0	I	O	O	O	O	O	O	O	O	O	O
2006-02-17 07:05:01	OK	O	2 7 0	I	O	O	O	O	O	O	O	O	O	O
2006-02-17 04:05:01	OK	O	2 7 0	I	O	O	O	O	O	O	O	O	O	O
2006-02-17 01:05:02	OK	O	2 7 0	I	O	O	O	O	O	O	O	O	O	O
2006-02-16 22:05:01	OK	O	2 7 0	I	O	O	O	O	O	O	O	O	O	O
2006-02-16 21:38:35	OK	O	2 7 0	I	O	O	O	O	O	O	O	O	O	O
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 5: SFTs for site CY01 (University of Cyprus)

Gate (the head node of the Computing Element) and collects information about all resources present at the site [19]. GIIS entries are requested by the GStat server by running an LDAP [6] search command every few minutes; the data returned to GStat is the reply from the GIIS of the corresponding Grid Gate. Below you can see the actual LDAP query used to obtain information from the GIIS at site CY01, and one of the entries returned as output (in this case, a supported type of software environment for the LHCb experiment [12]).

```
$ ldapsearch -x -H ldap://ce101.grid.ucy.ac.cy:2170 -b mds-vo-name=CY01-LCG2,o=grid


[...]

# VO-lhcb-Gauss-v24r6, ce101.grid.ucy.ac.cy, ce101.grid.ucy.ac.cy, CY01-LCG2, grid
dn: GlueLocationLocalID=VO-lhcb-Gauss-v24r6,GlueSubClusterUniqueID=ce101.grid.ucy.ac.cy,
 GlueClusterUniqueID=ce101.grid.ucy.ac.cy,mds-vo-name=CY01-LCG2,o=grid
objectClass: GlueClusterTop
objectClass: GlueLocation
objectClass: GlueSchemaVersion
objectClass: GlueKey
GlueLocationLocalID: VO-lhcb-Gauss-v24r6
GlueLocationName: VO-lhcb-Gauss-v24r6
GlueLocationVersion: Prod
GlueLocationPath: $VO_LHCB_SW_DIR
GlueChunkKey: GlueSubClusterUniqueID=ce101.grid.ucy.ac.cy
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 2

[...]
```

The data in Figure 6 consists of a number of ‘normal’ (up-to-date) entries indicated with the light-coloured line. The number of entries found varies from time to time due to the dynamic nature of the Grid (more specifically resulting from site configuration changes and changes of the software environment installed by the various VOs on the site). This means the number of normal entries can fluctuate and the site’s information system could still be considered error-free and up to date; on some occasions however, the entries abruptly drop to zero (or quite lower than the current value), perhaps due to some network fault that causes timeouts or even disconnections, or a failure of the GIIS daemons running on the Grid Gate. Such drops to zero can best be seen at the top-left graph, where the timeline is narrower. In this particular figure, the only drops to zero are shown on the top-right graph, but unfortunately due to the wider timeline and due to the fact that the error lasted for a very short time (perhaps a few minutes) the graph does not go all the way down to zero; even so, the two occurrences of a sudden fall followed by the immediate sudden rise are indications of such errors. In the bottom graphs, the timeline is even wider, 3 months for the bottom-left and a year for the bottom-right, so traces of such errors disappear completely.

The other type of GIIS entries found on the same graph (Figure 6) are so-called ‘old’ entries, meaning the information system of the site may not be up to date (the timestamp is older than 10 minutes). Such entries are shown on

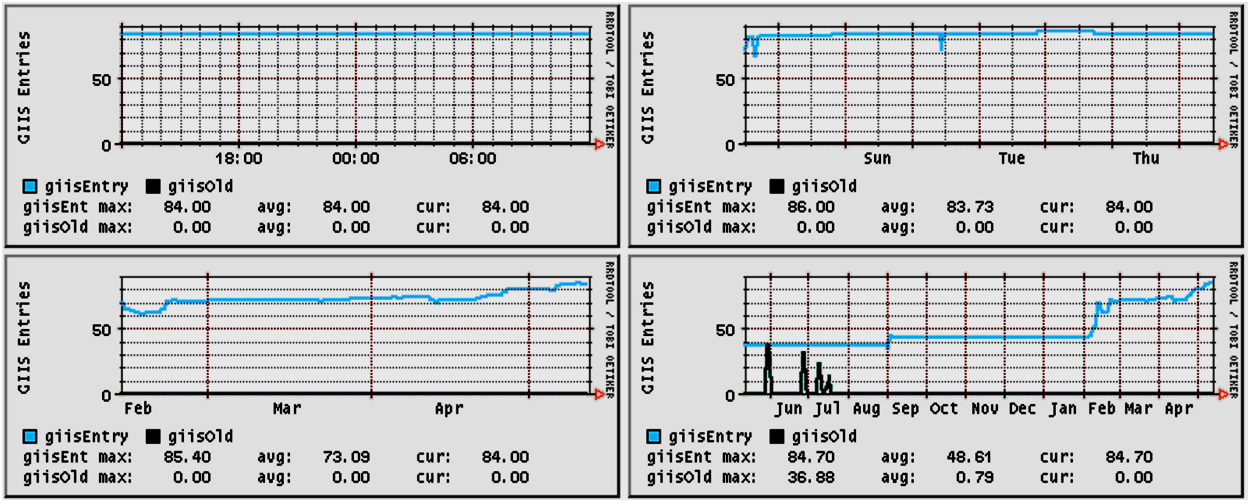


Figure 6: GStat GIIS entries at site CY01

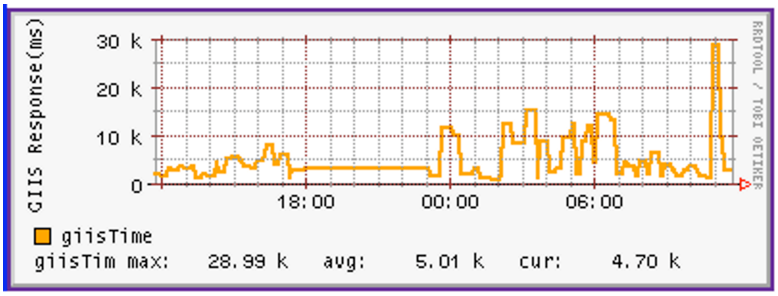


Figure 7: GStat GIIS response time for site CY01

the graphs with the darker line. For a site to pass testing, old entries must not exist, and the darker line should be at zero; an example of a problem with old entries can be seen on the bottom-right graph (between May and July), during a period in which the site suffered major network problems, indicated by the rise of the darker line above zero.

Other points of interest in GStat pages are total and 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 [10]. The website [11] 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. For example, if GStat shows increased response time for GIIS as in Figure 7, the SmokePing graph can indicate whether this is a general problem with the site's network as shown in Figure 8, this was indeed the case. Notice that the network latency shows an order of magnitude increase during the same interval that the GIIS response time displayed a similar increase. Suppose on the other hand that GIIS showed high response time (above the usual range of 2-15 s) but SmokePing graphs showed normal latency (usually around 40 ms)[5]: in such a case, the fault origin would most probably lie on the CE, due to heavy load, abnormal CPU or hard drive temperature etc.

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

[5] from our own observations these are the normal values during error-free periods

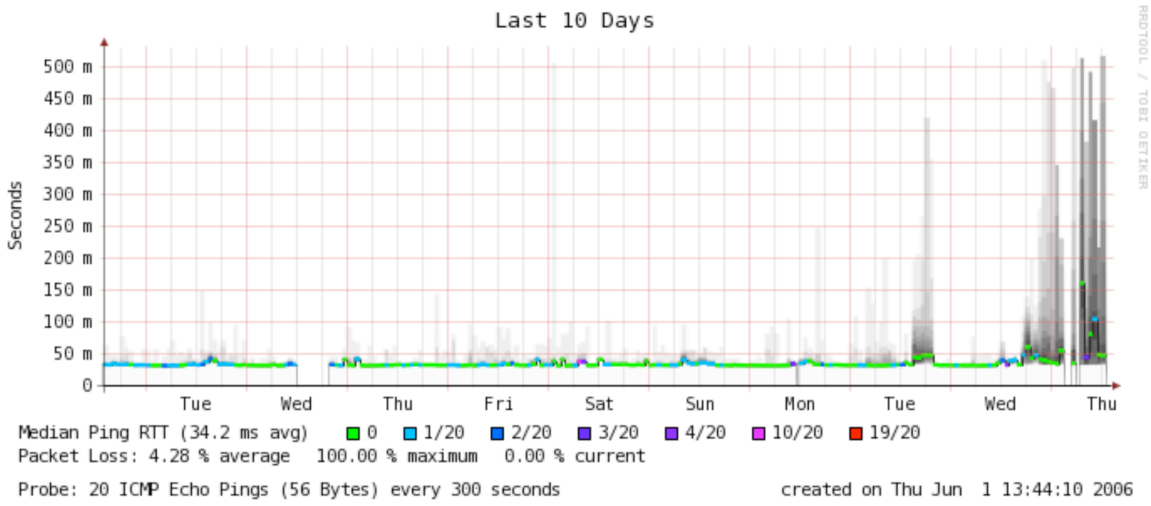


Figure 8: SmokePing network latency for site CY01

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. A ticket corresponds to a task-request and has various attributes such as the name and e-mail of the ticket submitter (initiator), the name of the responsible federation, the name and e-mail of the person who is assigned to the task, the ticket status (open, pending, solved, etc), and a log that shows the reason for opening the ticket, the work done, and how the problem was solved. To give an example, all tickets that were created for site CY01 from the beginning of the EGEE project in April 2004 up to the end of January 2006 are shown in Figure 9, while a more detailed view from one of these tickets can be seen in Figure 10.

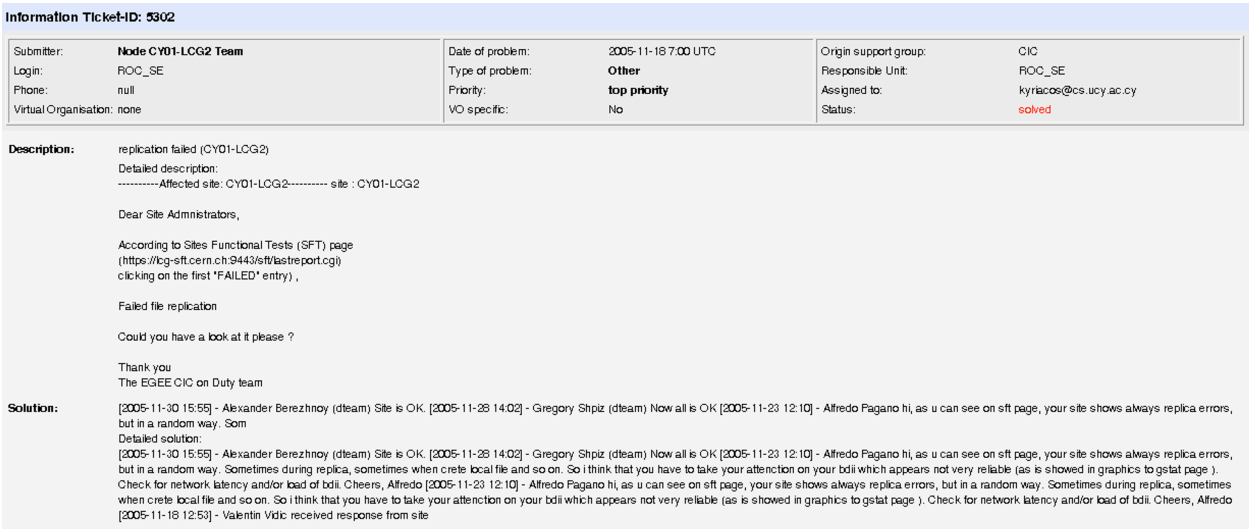
NEW SEARCH: Support Unit - User - **Keyword** Involved Supporter - Ticket-ID Keyword: CY01 in all Tickets

8 tickets found. Criteria: searchstring=CY01 status=all

Ticket-ID	Virt. Org.	Resp. Unit	Status	Date	Info
5302	none	ROC_SE	solved	2005-11-18	replication failed (CY01-LCG2)
5046	atlas	ROC_SE	solved	2005-11-02	replication failed
4927	atlas	ROC_SE	solved	2005-10-21	SEE VO user with SEE-GRID CA certificate can not s...
3412	atlas	ROC_SE	solved	2005-06-29	JS - Cannot read JobWrapper output...
2607	none	ROC_SE	solved	2005-05-24	still using lxn1178 instead of lcg-bdii.cern.ch
2405	none	GlobalGridUserSupport	solved	2005-05-04	Testing to make sure R-GMA is working correctly on...
2233	none	ROC_SE	solved	20-04-05 2	unstable CE information
1879	none	ROC_SE	solved	n/a	Replication from lxn1183.cern.ch to the default SE...

Figure 9: GGUS Search Interface – Searching for tickets by keyword

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. Once a GGUS ticket is opened, it is also visible to the affected federation's ticketing system, and intermediate updates are done there, but everything is also visible in the GGUS system, including the solution, the full log, and the ticket closing time and date. For this reason, a combination of both global and regional helpdesks is not necessary to make more sense of this type of error information, i.e. we only need to access the GGUS entries.



Information Ticket-ID: 5302

Submitter:	**Node CY01-LCG2 Team**	Date of problem:	2005-11-18 7:00 UTC	Origin support group:	CIC
Login:	ROC_SE	Type of problem:	**Other**	Responsible Unit:	ROC_SE
Phone:	null	Priority:	**top priority**	Assigned to:	kyriacos@cs.ucy.ac.cy
Virtual Organisation:	none	VO specific:	No	Status:	solved

Description: replication failed (CY01-LCG2)
Detailed description:
----------Affected site: CY01-LCG2---------- site : CY01-LCG2

Dear Site Administrators,

According to Sites Functional Tests (SFT) page
(https://lcg-sft.cern.ch:9443/sft/lastreport.cgi)
clicking on the first "FAILED" entry) ,

Failed file replication

Could you have a look at it please ?

Thank you
The EGEE CIC on Duty team

Solution: [2005-11-30 15:55] - Alexander Berezhnoy (dteam) Site is OK. [2005-11-28 14:02] - Gregory Shpiz (dteam) Now all is OK [2005-11-23 12:10] - Alfredo Pagano hi, as u can see on sft page, your site shows always replica errors, but in a random way. Som
Detailed solution:
[2005-11-30 15:55] - Alexander Berezhnoy (dteam) Site is OK. [2005-11-28 14:02] - Gregory Shpiz (dteam) Now all is OK [2005-11-23 12:10] - Alfredo Pagano hi, as u can see on sft page, your site shows always replica errors, but in a random way. Sometimes during replica, sometimes when crete local file and so on. So i think that you have to take your attenction on your bdii which appears not very reliable (as is showed in graphics to gstat page). Check for network latency and/or load of bdii. Cheers, Alfredo [2005-11-23 12:10] - Alfredo Pagano hi, as u can see on sft page, your site shows always replica errors, but in a random way. Sometimes during replica, sometimes when crete local file and so on. So i think that you have to take your attenction on your bdii which appears not very reliable (as is showed in graphics to gstat page). Check for network latency and/or load of bdii. Cheers, Alfredo [2005-11-18 12:53] - Valentin Vidic received response from site

Figure 10: GGUS ticket details (partial view)

Note that federation-level tickets can be opened also, and in such cases GGUS has no corresponding entries, but these tickets mostly relate to needed updates (and not failures) that each federation has chosen to handle internally.

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. An e-mail sent through CIC can be seen in Figure 11. 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. Frequently the downtime announcements follow a series of SFT failures and a resulting ticket prompting the site to fix the errors; at other times the administrator declares the site down before the operations support has the time to open a ticket.

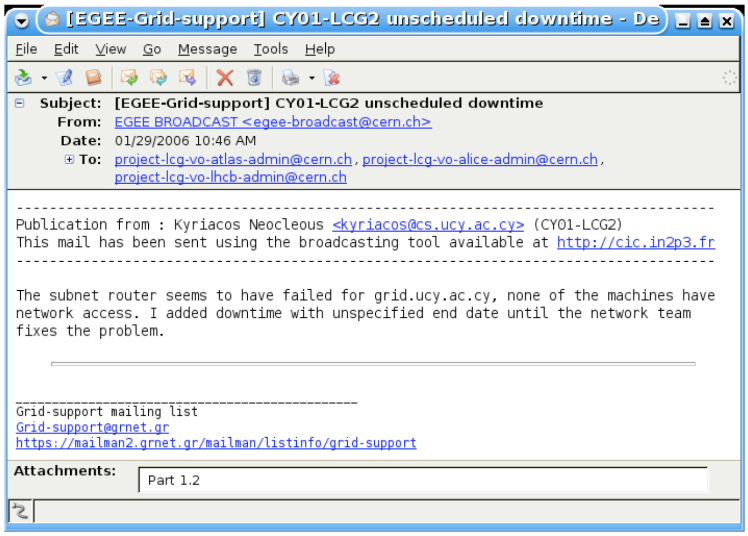


Figure 11: CIC e-mail broadcast

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 many sites of many different countries, usually an entire region.[6]

Next we provide an example of an error logged to the CE Grid Gate file `/var/log/messages`. This file is only accessible using the root account (the one normally held by site administrators). [7] This particular error is logged by the Grid Resource Allocation Manager (GRAM) *gatekeeper* service. Normally this service is responsible for accepting connections, authenticating the remote user, and mapping the remote credentials of that user onto a local user ID (a VO *pool account*). The error message below (the last line of output) indicates denial in accessing the requested computing resources due to a problem with mapping of the user's remote credentials onto a local account. Note that submitting IP addresses and certificate subject have been altered for privacy reasons.

```
Nov 29 13:53:49 ce101 GRAM gatekeeper[26331]:
Got connection 195.1.1.256 at Tue Nov 29 13:53:49 2005
Nov 29 13:53:49 ce101 GRAM gatekeeper[26331]:
Trying to use original user proxy ...
Nov 29 13:53:49 ce101 GRAM gatekeeper[26331]:
Authenticated globus user: /C=GR/O=HG/OU=grid.gr/CN=SomeName
Nov 29 13:53:49 ce101 GRAM gatekeeper[26331]:
LCMAPS failed user mapping.
```

An example from the Logging and Book-keeping Service (LBS) database of the SEE federation Resource Broker can be seen next. This logging was the result of submitting a simple job from the User Interface, in order to observe what happens in LBS if a job failure occurs. The next output is a part of the answer to a query on database 'lbserver20', table 'states' (formatted for improving readability). This selection contains the Job Description Language (JDL) information used to submit this job, with added JDL information from WMS components. [8]

```
requirements  = ( other.GlueCEStateStatus == "Production" );
RetryCount  = 3;
edg_jobid  = "https://rb101.grid.ucy.ac.cy:9000/15r0w6bgsc59Hst3S5CArg";
Arguments  = "-al";
OutputSandboxPath  = "/var/edgwl/SandboxDir/...2f15r0w.../output";
MyProxyServer  = "myproxy.grid.auth.gr";
JobType  = "normal";
Executable  = "/bin/ls";
CertificateSubject  = "/C=CY/O=CyGrid/O=HPCL/CN=Kyriacos Neocleous";
X509UserProxy  = "/opt/edg/var/spool/edg-wl-renewd/b68edbe495655235978d28dd555516d7.0";
StdOutput  = "std.out";
OutputSandbox  = { "std.out","std.err" };
InputSandboxPath  = "/var/edgwl/SandboxDir/...2f15r0w.../input";
LB_sequence_code  = "UI=000003:NS=0000000003:...LRMS=000000:APP=000000";
VirtualOrganisation  = "see";
rank  =  -other.GlueCEStateEstimatedResponseTimeType = "job";
StdError  = "std.err";
DefaultRank  =  -other.GlueCEStateEstimatedResponseTime;
InputSandBox  = {  };
edg_sync_running_event  = false
```

Observing the entire output of this query (not shown above) at closely separated time intervals, we found that the data is being updated during the lifetime of a job: the first time the query was run, it was showing a different Computing Element (CE). Obviously the job failed at that particular CE for some reason, and it was migrated to CE node001.grid.auth.gr without user intervention. While table 'states' only keeps the latest [most current] job state information, the LB database keeps all information regarding the life cycle of a job (tables of interest are 'events', 'short_fields' and 'long_fields'). The LBS database actually keeps a state machine view for each job [23]. In general there are two methods for obtaining this job state machine information. The first one is by querying the LB database

[6] More often than not there are secondary RBs in a federation to provide some sort of manual failover and ease the load on the primary RB and Information Index (BDII).

[7] Most log files useful for this study are only accessible with the root account.

[8] Components such as the Job Adapter supplement the user-submitted JDL with more information [23]. To give an example, the path of the output sandbox on the Resource Broker (the 'OutputSandboxPath' variable seen below) is added in the JDL by the Job Adapter).

as the owner of a job by issuing the 'edg-job-get-logging-info' command; depending on the verbosity level set (0..2), this command can show a high level summary or all LBS entries related to the job in question. The second method for obtaining job state machine information is by directly querying the LB database on the RB host through the mySQL interface.

To better understand the usefulness of LBS in this study, Grid job with id = J5j8J2jHvDia35eevMj3yg will now be analysed, since this was another one of the cases of initial failure and automatic resubmission to a different Computing Element. In the LB database, table 'events' shows all events associated with this job. Attribute *event* is the serial number of the event, *prog* is the Grid component handling the event, and *host* registers the Fully Qualified Domain Name (FQDN) of the machine hosting the service responsible for handling that event. As one can infer from the data below, the job was submitted from the CY01 User Interface (ui101.grid.ucy.ac.cy) to the South East Europe Resource Broker (rb101.grid.ucy.ac.cy), going through the various WMS components (e.g. Network Server, Workload Manager, Local Resource Management System), until it terminates (event 31).

```
mysql> select jobid, event, prog, host from events where (jobid="J5j8J2jHvDia35eevMj3yg");
+------------------------+-------+-----------------+----------------------+
| jobid                  | event | prog            | host                 |
+------------------------+-------+-----------------+----------------------+
| J5j8J2jHvDia35eevMj3yg |     0 | UserInterface   | ui101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     1 | UserInterface   | ui101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     2 | NetworkServer   | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     3 | NetworkServer   | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     4 | UserInterface   | ui101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     5 | WorkloadManager | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     6 | WorkloadManager | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     7 | WorkloadManager | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     8 | WorkloadManager | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |     9 | JobController   | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    10 | JobController   | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    11 | JobController   | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    12 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    13 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    14 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    15 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    16 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    17 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    18 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    19 | WorkloadManager | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    20 | WorkloadManager | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    21 | WorkloadManager | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    22 | WorkloadManager | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    23 | JobController   | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    24 | JobController   | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    25 | JobController   | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    26 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    27 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    28 | LRMS            | wn04.grid.acad.bg    |
| J5j8J2jHvDia35eevMj3yg |    29 | LRMS            | wn04.grid.acad.bg    |
| J5j8J2jHvDia35eevMj3yg |    30 | LogMonitor      | rb101.grid.ucy.ac.cy |
| J5j8J2jHvDia35eevMj3yg |    31 | LogMonitor      | rb101.grid.ucy.ac.cy |
+------------------------+-------+-----------------+----------------------+
```

Part of the output from a query on the LBS database, table 'short_fields' is shown below (some irrelevant data has been stripped out). This tabled consists of four fields: the *jobID* attribute uniquely identifies a job; the *event* attribute is a serial number to tie together several attribute-value pairs that belong to the same event, the *name* and *value* fields (shown below). In this case, the job was at some point successfully submitted (event 13) to the Local Resource Management System (LRMS) on the Computing Element of *buggySite* (the name of the problematic site has been changed for obvious reasons). Event 15 indicates the error and event 16 indicates that there will be a resubmission. The job then ends up to a different Computing Element, at the Grid site of the Bulgarian Academy of Sciences, and starts running on a Worker Node as it can be seen from event 28, to finally terminate successfully as indicated by the last event.

```
mysql> select * from short_fields where jobID="J5j8J2jHvDia35eevMj3yg";
+------------------------+-------+--------------+----------------------------+
| jobid                  | event | name         | value                      |
+------------------------+-------+--------------+----------------------------+
| ...                    |    .. | ...          | ...                        |
| J5j8J2jHvDia35eevMj3yg |    13 | DESTINATION  | LRMS                       |
| J5j8J2jHvDia35eevMj3yg |    13 | DEST_HOST    | ce01.buggySite.org         |
| ...                    |    .. | ...          | ...                        |
| J5j8J2jHvDia35eevMj3yg |    13 | REASON       | Job successfully submitted |
| J5j8J2jHvDia35eevMj3yg |    13 | RESULT       | OK                         |
| ...                    |    .. | ...          | ...                        |
| J5j8J2jHvDia35eevMj3yg |    15 | EXIT_CODE    | 1                          |
| J5j8J2jHvDia35eevMj3yg |    15 | REASON       | Cannot read JobWrapper ... |
| J5j8J2jHvDia35eevMj3yg |    15 | STATUS_CODE  | FAILED                     |
| ...                    |    .. | ...          | ...                        |
| J5j8J2jHvDia35eevMj3yg |    16 | RESULT       | WILLRESUB                  |
| ...                    |    .. | ...          | ...                        |
| J5j8J2jHvDia35eevMj3yg |    28 | NODE         | wn04.grid.acad.bg          |
| J5j8J2jHvDia35eevMj3yg |    28 | SEQCODE      | UI=000003:NS=00...0000     |
| ...                    |    .. | ...          | ...                        |
| J5j8J2jHvDia35eevMj3yg |    30 | NODE         | ce01.grid.acad.bg          |
| ...                    |    .. | ...          | ...                        |
| J5j8J2jHvDia35eevMj3yg |    31 | EXIT_CODE    | 0                          |
| J5j8J2jHvDia35eevMj3yg |    31 | REASON       | Job terminated successfully|
| ...                    |    .. | ...          | ...                        |
| J5j8J2jHvDia35eevMj3yg |    31 | STATUS_CODE  | OK                         |
+------------------------+-------+--------------+----------------------------+
```

This information can also be seen using 'edg-job-get-logging-info'. The output of this command provides information that results from a join operation on several LB database tables (events, short_fields and long_fields). Here we again only provide the interesting entries:

```
> edg-job-get-logging-info -v 2 https://rb101.grid.ucy.ac.cy:9000/J5j8J2jHvDia35eevMj3yg

[...]

 Event: Running
- host                   =    rb101.grid.ucy.ac.cy
- level                  =    SYSTEM
- node                   =    ce01.buggySite.org
- priority               =    asynchronous
- seqcode                =    UI=000003...
- source                 =    LogMonitor
- src_instance           =    unique
- timestamp              =    Thu May 11 09:06:48 2006
- user                   =    /C=CY/O=CyGrid/O=HPCL/CN=Kyriacos Neocleous

        ---


 Event: Done
- exit_code              =    1
- host                   =    rb101.grid.ucy.ac.cy
- level                  =    SYSTEM
- priority               =    asynchronous
- reason                 =    Cannot read JobWrapper output ...
- seqcode                =    UI=000003...
- source                 =    LogMonitor
- src_instance           =    unique
- status_code            =    FAILED
- timestamp              =    Thu May 11 09:08:14 2006
- user                   =    /C=CY/O=CyGrid/O=HPCL/CN=Kyriacos Neocleous

        ---

 Event: Resubmission
- host                   =    rb101.grid.ucy.ac.cy
- level                  =    SYSTEM
- priority               =    asynchronous
- reason                 =    unavailable
- result                 =    WILLRESUB
- seqcode                =    UI=000003...
- source                 =    LogMonitor
- src_instance           =    unique
- tag                    =    unavailable
- timestamp              =    Thu May 11 09:08:15 2006
- user                   =    /C=CY/O=CyGrid/O=HPCL/CN=Kyriacos Neocleous

[...]
```

```
 Event: Running
- host                    =    wn04.grid.acad.bg
- level                   =    SYSTEM
- node                    =    wn04.grid.acad.bg
- priority                =    asynchronous
- seqcode                 =    UI=000003...
- source                  =    LRMS
- timestamp               =    Thu May 11 09:10:03 2006
- user                    =    /C=CY/O=CyGrid/O=HPCL/CN=Kyriacos Neocleous

[...]

 Event: Done
- exit_code               =    0
- host                    =    rb101.grid.ucy.ac.cy
- level                   =    SYSTEM
- priority                =    asynchronous
- reason                  =    Job terminated successfully
- seqcode                 =    UI=000003...
- source                  =    LogMonitor
- src_instance            =    unique
- status_code             =    OK
- timestamp               =    Thu May 11 09:13:28 2006
- user                    =    /C=CY/O=CyGrid/O=HPCL/CN=Kyriacos Neocleous
```

4.2 Assessment of sources and next steps

In the remaining of this section we will assess the various sources of information about grid errors discussed in 4.1. For a better understanding of the various levels on which monitoring is done you can consult Figure 12 at the end of this section.

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, but this is not easy to do automatically because the graphs are in image format and the data used to generate the graphs are 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: While easy to gather and easy to separate between "downtime due to errors" and "downtime due to standard maintenance tasks" without the need to automate the process (such entries are infrequent), these sources 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 occasions, 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 are to be found at the lowest levels of the grid nodes themselves (refer to Figure 12, 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

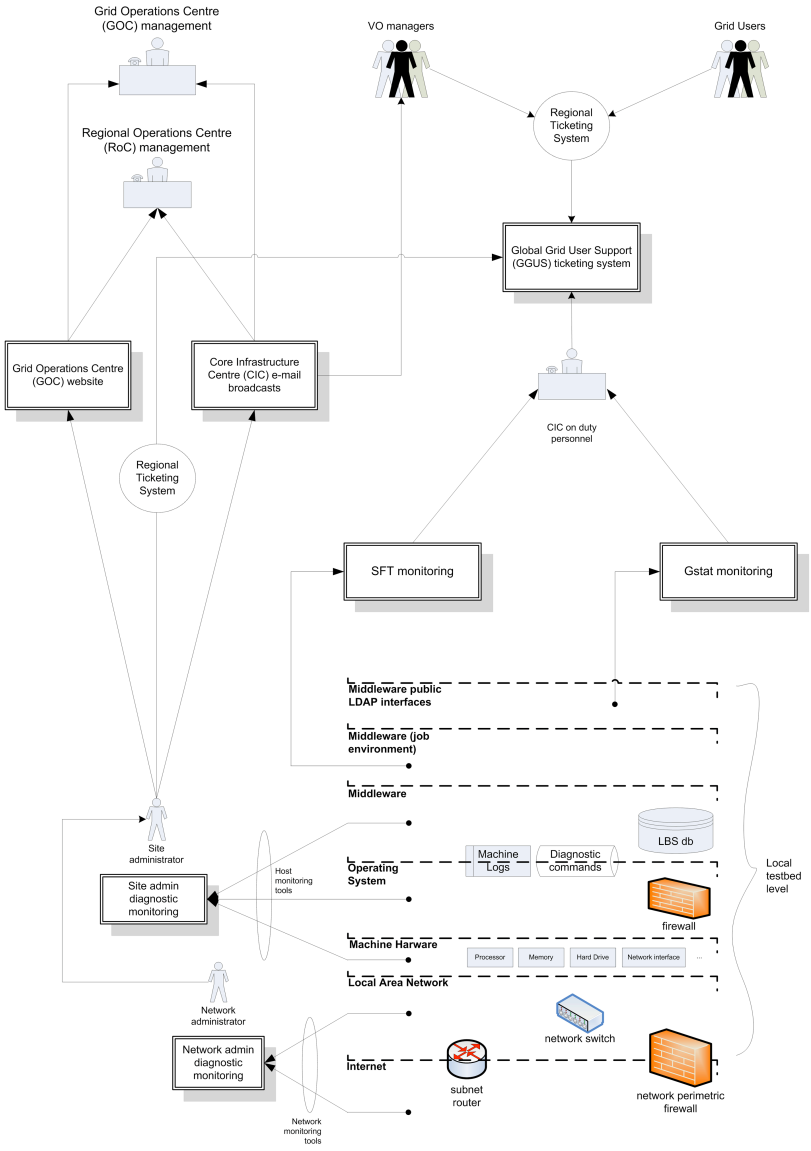


Figure 12: Error information sources: levels of monitoring

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.

5 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 2, 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 and the 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 [7, 22], 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 [14]) 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
job '79106' started on 1 proc
job '79107' cannot be 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 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`[9], notice production job of the LHCb experiment [12], with ID 74896.ce101. It appears to be running (Status [S] = Running [R]).

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

```
[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. After restarting the failed node, the job was 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 started, 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

checking job 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]
...
Reservation '74896' (-00:54:19 -> 2:23:05:41  Duration: 3:00:00:00)
PE:  1.00  StartPriority:  1
```

As it turned out 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'
```

```
[root@ce101 root]# showres -n
reservations on Thu Feb  2 15:44:22

            NodeName  Type  ReservationID  JobState  Task     Start    Duration ...
wn111.grid.ucy.ac.cy   Job          75110   Running     1  -6:21:20  3:00:00:00 ...
wn113.grid.ucy.ac.cy   Job          75112   Running     1  -4:50:44  3:00:00:00 ...
2 nodes reserved
```

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.ce101.JB and 74896.ce101.SC), 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 'understand' 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.

6 Conclusions

Detecting and managing failures is an important step toward the goal of a dependable grid. However, this is currently an extremely complex task due to the complexity, the scale, and and the multi-institutional span of Grid infrastructures. In this paper, we examined the problem of failure detection and management in the context of EGEE, the largest Grid infrastructure in operation world-wide. We identified the sources that provide information about errors on the EGEE Computing Grid, and assessed these sources in terms of their usefulness, accuracy and completeness of the error information provided. Moreover, we presented and analyzed the error types that can lead to Grid job failure. We presented in detail two case studies of Grid errors by describing the problem symptoms, the root cause of the failure, and the troubleshooting process that was used to resolve the problem.

The experiences described in this paper show that manual failure management in large-scale infrastructures such as EGEE is a tedious and cumbersome process. Furthermore, 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 route 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, George Tsouloupas and Demetris Zeinalipour-Yazti 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 Feb. 2006).

[2] Enabling Grids for E-SciencE project. http://www.eu-egee.org/.

[3] gLite Middleware. http://glite.web.cern.ch/glite/ (accessed June 2006).

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

[5] LCG Middleware. http://lcg.web.cern.ch/LCG/activities/middleware.html (accessed June 2006).

[6] Lightweight Directory Access Protocol, open source implementation, website. http://www.openldap.org (accessed June 2006).

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

[8] MPI: A Message-Passing Interface Standard. http://www.mpi-forum.org/docs/mpi-11.ps (accessed June 2006).

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

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

[11] SmokePing results for EGEE site CY01.
http://mon.egee-see.org/cgi-bin/smokeping.cgi?target=World.Europe.SEE (accessed June 2006).

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

[13] The WISDOM (Wide In Silico Docking On Malaria) Data Challenge, general statistics. http://wisdom.eu-egee.fr/malaria/grid_stat.php?menu_grid=general (accessed June 2006).

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

[15] WISDOM: Initiative for grid-enabled drug discovery against neglected and emergent diseases. http://wisdom.eu-egee.fr/ (last accessed June 2006).

[16] Internet X.509 Public Key Infrastructure – Certificate and Certificate Revocation List (CRL) Profile. http://www.ietf.org/rfc/rfc3280.txt (accessed March 2006), 2002.

[17] Job Description Language: Attributes Specification.
http://edms.cern.ch/document/590869/, May 2006.

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

[19] Stephen Burke, Simone Campana, Antonio Delgado Peris, Flavia Donno, Patricia Mendez Lorenzo, Roberto Santinelli, and Andrea Sciaba. gLite 3.0 User Guide. https://edms.cern.ch/document/722398/, May 2006. Document Status: PRIVATE.

[20] G. Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design*, chapter 1: Characterization of Distributed Systems. Addison-Wesley, 3rd edition, 2001.

[21] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15(3):200–222, 2001.

[22] Sophie Lemaitre, Jeff Templon, Steve Traylen, Markus Schulz, and Davide Salomoni. Maui Cookbook. http://grid-deployment.web.cern.ch/grid-deployment/documentation/Maui-Cookbook.pdf (accessed May 2006).

[23] F. Pacini. gLite Workload Management System service.
https://edms.cern.ch/document/572489/, May 2006.

[24] D. Thain and M. Livny. *Grid 2: Blueprint for a New Computing Infrastructure*, chapter 19: Building Reliable Clients and Services. Elsevier, Morgan Kaufmann, 2nd edition, 2004.

[25] M. Xu, Z. Hu, W. Long, and W. Liu. *Grid 2: Blueprint for a New Computing Infrastructure*, chapter 14: Service Virtualization: Infrastructure and Applications. Elsevier, Morgan Kaufmann, 2nd edition, 2004.