

# LHC Grid Computing Project

## PRELIMINARY OBSERVATIONS ON LCG-2 BASED ON THE 2004 LHC DATA CHALLENGES

---

Document identifier:	<b>LCG-GAG-DC04</b>
Date:	<b>28/09/2004</b>
Authors:	<b>D.Adams (BNL) D.Barberis (CERN/Genoa), I.Bird (CERN), N.Brook (Bristol), S.Burke (RAL), P.Buncic (CERN), F.Carminati (CERN), R.Cavanaugh (UF), P.Cerello (INFN), F.Donno (CERN/INFN), D.Foster (CERN), C.Grandi (INFN), F.Harris (CERN/Oxford), L.Perini (INFN), A.Pfeiffer (CERN), R.Pordes (FNAL), A.Sciabà (CERN/INFN), O.Smirnova (CERN/Lund), J.Templon (NIKHEF), A.Tsaregorodtsev (IN2P3)</b>
Editors:	<b>F.Carminati (CERN), J.Templon (NIKHEF)</b>
Document status:	<b>Version v1.1 – Final</b>

---

**Abstract:** This document contains a preliminary report based on the experience acquired by the LHC experiments during the 2004 Data Challenges with the LCG-2 middleware.

## Table of contents

<b>1</b>	<b>INTRODUCTION .....</b>	<b>2</b>
<b>2</b>	<b>SHORT DESCRIPTION OF EXPERIMENT DC'S .....</b>	<b>3</b>
2.1	GENERAL REMARKS .....	3
2.2	ALICE .....	4
2.2.1	Strategy.....	4
2.2.2	Configuration.....	4
2.2.3	Software Installation .....	5
2.2.4	Statistics .....	5
2.3	ATLAS .....	7
2.4	LHCb.....	10
<b>3</b>	<b>PROBLEMS ENCOUNTERED.....</b>	<b>12</b>
3.1	EXPERIMENT SOFTWARE INSTALLATION .....	12
3.2	LOCAL SITE CONFIGURATION.....	12
3.3	INFORMATION SYSTEM AND MONITORING .....	13
3.4	WORKLOAD MANAGEMENT SYSTEM .....	13
3.5	DATA MANAGEMENT .....	16
3.6	MISCONFIGURATION PROBLEMS .....	17
3.7	MISCELLANEOUS OPERATION PROBLEMS .....	17
<b>4</b>	<b>REQUIREMENTS .....</b>	<b>18</b>
4.1	EXPERIMENT SOFTWARE INSTALLATION .....	18
4.2	LOCAL SITE CONFIGURATION.....	18
4.3	WORKLOAD MANAGEMENT SYSTEM .....	18
4.4	DATA MANAGEMENT .....	19
<b>5</b>	<b>GENERIC EFFICIENCY TESTS .....</b>	<b>20</b>

## 1 Introduction

LCG-2 has been heavily used during 2004 by the four LHC experiments for large distributed data challenges. LCG-2 has been in operation for 6 months and this represents a snapshot of the experiments' experiences at the end of August. This has been one of the first attempts to use a computational Grid at this scale for real production, and the feedback from the experiments is very important to guide the future evolution of the EGEE/LCG middleware in the little time left before real data will come.

These data challenges represent a significant achievement, as they are (to our knowledge) the largest-scale computational efforts on generic Grid infrastructures to date. The largest previous efforts were the participation of the European DataGrid testbed in the CMS Data Challenge (2002) and D0 Re-reconstruction (2003). The efforts described in this document exceed those by at least an order of magnitude in volume. They also have, as have all significant increases in scale before, exposed a new level of scalability problems. This is expected; however, as we will describe in detail below, there is a disappointing lack of convergence in the area of site configuration, which is still responsible for the lion's share of problems in the system.

We have collected here a summary of the main remarks and perceived shortfalls of the system divided by main components. We gratefully acknowledge the good will and competence of the LCG and EGEE personnel, who, amongst many difficulties, have made these tests possible and have

bravely sustained our continuous requests for more functionality and stability doing their best to meet them.

The focus of this document is to formulate requirements based on the current experience acquired by the experiments during their 2004 data challenges. We will describe briefly the data challenge setup for each experiment and then we will discuss the problems encountered with the LCG components that have been tried. From these we derive requirements that are addressed to the present LCG implementation as well as to the future middleware. An important input to this document is an internal EIS document (<https://edms.cern.ch/file/495809//Broker-Requirements.pdf>) that contains their perception of the experiment feedback. We found this document a very good source that we have used to complement the present document.

We expect that as the scale of the system and of the data challenges grows, new requirements may become apparent, and therefore a new and revised version of this document may be issued in the framework of a continuous requirement refining as it is in the mandate of GAG.

## 2 Short description of experiment DC's

### 2.1 General Remarks

All three experiments wound up implementing their production system on the LCG-2 software stack in a similar way. Each saw the LCG service as *one*, not *the only*, target system. Hence each experiment implemented its own systems for workload definition, bookkeeping, and data management. The general scheme is sketched below. It should be noted not all experiments have stress tested all elements of LCG and different usage strategies are useful in identifying shortcomings.

Jobs were defined and submitted to the experiment-specific production database (ATLAS, LHCb) or task queue (ALICE). A translator facility retrieves the jobs from the central database, translates the jobs to LCG Job Description Language (JDL), and submits them *via* the regular LCG job-submission tools. For ATLAS and ALICE, the job being sent to the Workload Management System (WMS) was completely specified at submission time. For LHCb, the job being sent to the WMS was an 'agent' job, which upon starting execution, contacted the LHCb production database and retrieved the 'real' job description to be executed.

The experiments made less use of the LCG data management commands. Each experiment had its own private file catalogue services; ATLAS and ALICE had a link from their private services to those of LCG, while LHCb did not.

Each of the three experiments relied on their own system for monitoring and bookkeeping.

The degree to which the experiments used the LCG facilities reflects rather well the degree of usability of those facilities. In general the WMS is much more robust than the Data Management System (DMS), and provides more of the needed functionality. The interface to the monitoring and

bookkeeping systems is so unfriendly, and provides so little of the desired functionality, that the experiments had to develop their private solutions. Here follows a short description of how the experiments have conducted their data challenges.

## **2.2 ALICE**

### **2.2.1 Strategy**

ALICE is using *AliEn* native sites and LCG resources to run its Data Challenge. LCG-2 is used from *AliEn* via a “keyhole” approach. The *AliEn* workload management sees the entire LCG system as a very large computing and storage element. Jobs going to LCG are first pulled by a special instance of the *AliEn* computing element that runs on a machine which is also configured as LCG User Interface; after pulling a job, the Job Definition Language (JDL) is made LCG-compliant and the job is submitted to the LCG resource broker (RB). From there on the job is managed by LCG, although it starts reporting to the *AliEn* server about its status as soon as it starts running on a LCG Worker Node. This approach has been extended also to the INFN Grid (*grid.it*).

A similar strategy is used for storage. Jobs executed on LCG write their output on LCG Storage Elements, but register it both in the LCG and in the *AliEn* Data Catalogue for future access. The double registration is required to make sure that the LCG Data Catalogue is self-consistent and the *AliEn* Data Catalogue contains the complete set of data generated by ALICE during the Data Challenge.

For each registered file, the LCG storage element (SE) name and GUID are stored in the *AliEn* file catalogue as physical name (PN). When a job requiring access to LCG-stored data is sent to the LCG-CE, *AliEn* resolves the logical name (LN) into the PN, which in this case are the LCG SE and GUID.

The idea is that, when an analysis job is submitted to *AliEn*, the *AliEn* Data Catalogue knows the physical location of data stored in LCG. Using this information, *AliEn* performs job splitting making sure that each job submitted to LCG request files that are all on the same LCG SE. The LCG RB will decide where to send the job based on the input LN's and this will test its matchmaking capabilities with respect to data location. This mechanism is in place and will be used in phase III of the ALICE Data Challenge due to start in October 2004.

### **2.2.2 Configuration**

ALICE started out with no dedicated RB or information index (BDII). LCG-2 core sites were seen through the default CERN RB, while *grid.it* resources were seen through a separate interface submitting to a RB in Catania and, later, to the *grid.it* default RB at CNAF. After some time, a dedicated RB and BDII were installed for ALICE by the LCG staff.

*AliEn* is installed as experiment software on remote sites (like *AliRoot*) and the shell script sent to the WN only executes it. *AliEn* services run on a dedicated interface site, which is at the same time an *AliEn* CE (it actually runs the CE, SE and ClusterMonitor services) and an LCG-2 UI. As already described, the

CE gets jobs from the *AliEn* master queue and forwards them to the LCG RB via a generated JDL file and job wrapper shell script.

For Phase I, all generated data was transferred to CERN CASTOR via AIOD, the *AliEn* I/O Daemon. In Phase II, the generated files are stored in the local LCG SE (via CopyAndRegister, on the default/close SE, executed by *AliEn*) and registered in the *AliEn* Data Catalogue using the LCG SE and GUID as physical file name (in the form `lcg://node.domain:8092/6c733811-d295-11d8-aae8-eeeda3a3374b`, where `node.domain` is the hostname of the SE where data were stored). A backup zip file of the files is again sent to CERN CASTOR via AIOD. **Software Installation**

ALICE relies on the fact that pre-compiled code including all required libraries is distributed. An installation perl script running on a LCG UI takes as input the shell scripts and associated JDLs for the installation of AliRoot and *AliEn*, queries the RB with *edg-job-list-match* and submits 2 jobs (AliRoot and *AliEn* installation) to each of the sites returned by *edg-job-list-match*. In case of more than a queue per site, the first one in the list of matching queues is taken. After the installation, AliRoot is validated by the installation job itself, which simulates a low multiplicity event. The installation jobs, if successful, publish the TAGS required by production JDLs. *AliEn* is tested directly with the production jobs. Also, scripts to test the environment are available. These jobs are typically run once per month or so.

#### 2.2.4 Statistics

The LCG functionality used by ALICE is different for phase I and phase II of our Data Challenge. In phase one, Data Management and Storage Services were not used; on the other hand, they are massively used in phase II. The following is a summary of LCG statistics for the ALICE Data Challenge. Fig. 1 and Fig. 2 show the contribution of involved sites to the number of successfully completed jobs for phase I and II, respectively.

##### *Phase I (completed)*

Successfully completed jobs	11k (out of 56k total)
Average job duration	7.5 hours
CPU used	57 MSI-2k hours
Number of jobs sent to LCG	14.5k (about 76% efficiency)
Storage:	26TB at CERN in CASTOR, seen as <i>AliEn</i> SE
Registration:	<i>AliEn</i> Data Catalogue

##### *Phase II (ongoing)*

The effort to work in stable condition (both as a function of time and as a function of sites) has not led to a satisfactory situation yet. However, the following is a summary of operations during last month on LCG (August 16 to September 14), as reported by our validation script. A total of about 14 k jobs were submitted, with the following result:

Jobs DONE	5990
Jobs ERROR_IB	1411 (error in staging the input from

	CASTOR)
Jobs ERROR_V	3090 (not enough virtual memory on the WN or AliRoot failure - about 4% average)
Jobs ERROR_SV	2195 (Data Management or Storage Element failure)
Jobs ERROR_E	1277 (typically NFS failures, so the executable is not found)
Jobs KILLED	219 (jobs that fail to contact the AliEn Server when started and stay QUEUED forever while they are Running - also forever - in LCG)
Jobs FAILED	330 (inconsistency between the status reported by AliEn and LCG; something wrong during the execution)
Jobs RESUB	851 (jobs which stayed in the LCG queue for a long time without being started, so they are cancelled and resubmitted)

The overall efficiency of LCG service (WMS + DMS + SE) can be expressed as:

$$\text{DONE}/(\text{DONE}+\text{ERROR\_SV}+\text{KILLED}+\text{FAILED}+\text{RESUB}) = 62\%$$

Where ERROR\_SV accounts for the DMS and SE inefficiency, while KILLED+FAILED is related to WMS inefficiencies. However, the overall efficiency of the system is given by:

$$4\% + \text{DONE}/(\text{Total} - \text{ERROR\_IB}) = 49\%$$

where 4% is the average fraction of actual AliRoot failures.

This average number comes from very different performances of different sites, suggesting there is much room for improvement. In particular, ERROR\_E status always comes from big sites, where NFS problems are more likely. On the other hand, ERROR\_SV seems to be related to misconfigurations of the Data Management and/or Storage Element services. Problems are always reported, generating reactions that usually fix the situation, but not for a long time, as already-known errors tend to reappear on the same sites.

We also observe that several sites, suddenly, start to abort jobs. Clearly, when LCG problems are causing the failure of all the jobs going to a given site, that site is removed from the production. So, the reported value of efficiency could only be reached thanks to a continuous monitoring of the situation.

However, we would like to point out that, since the beginning of September, the performance has steadily improved, giving 80% as LCG service efficiency and 55% as global average value.

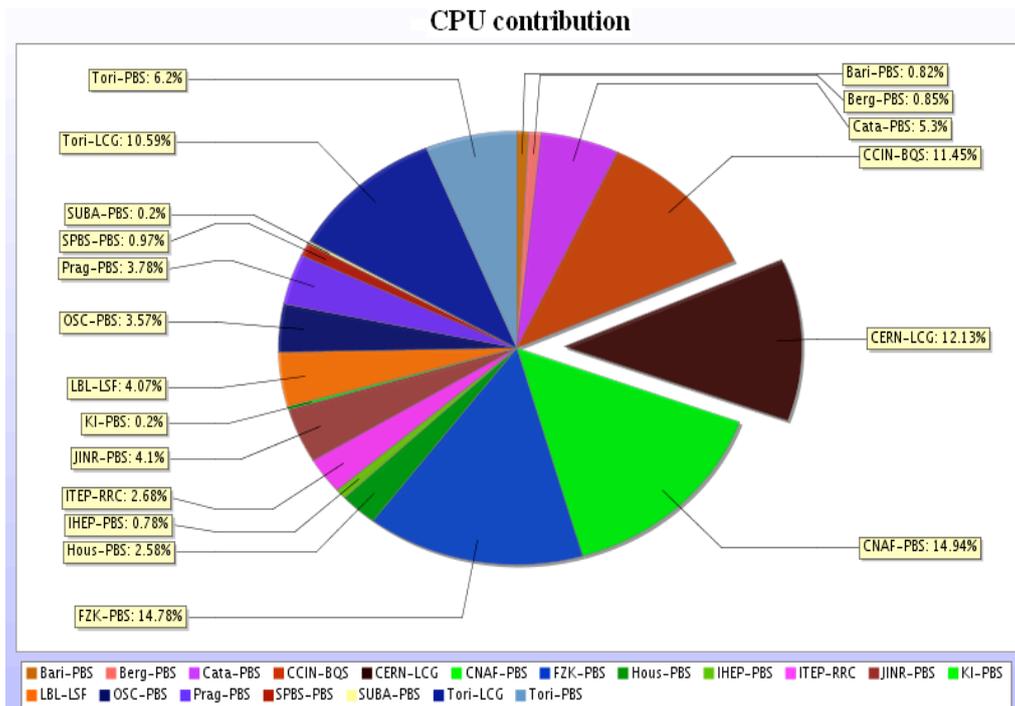


Figure 1: The contribution of involved sites to successfully completed jobs for phase I of the ALICE Physics Data Challenge 2004

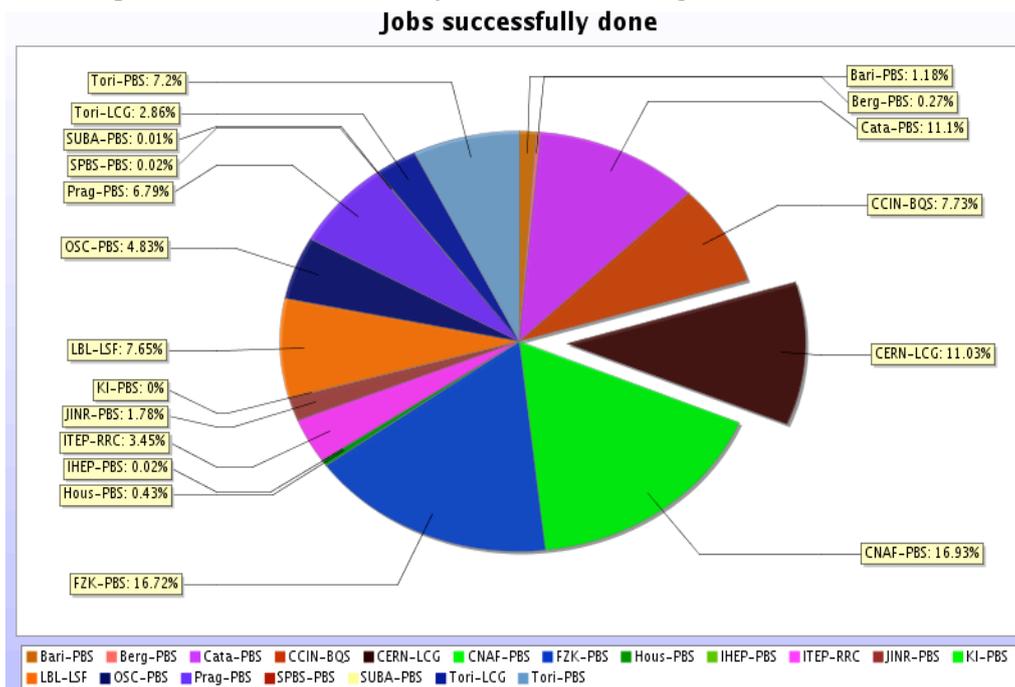


Figure 2: The contribution of involved sites to successfully completed jobs for phase II of the ALICE Physics Data Challenge 2004. Phase II is still unfinished.

### 2.3 ATLAS

For the ATLAS Data Challenge, all jobs are defined and stored in a central database. A supervisor agent (Windmill) picks them up, and sends their definition as XML message to various executors, via a Jabber server.

Executors are specialised agents, able to convert the XML job description into a Grid specific language (e.g. JDL for LCG). Four executors have been developed, for LCG (Lexus), Nordugrid (Dulcinea), GRID3 (Capone) and legacy systems (i.e. resources not available via Grid tools such as “classical” PBS clusters or a Condor pools), allowing the Data Challenge to be run on different Grids.

When a LCG job is received by Lexus, it builds the corresponding JDL description, creates some scripts for data staging, and sends everything to a dedicated, standard LCG RB through a Python module built over the WMS API. The requirements specified in the JDL let the RB choose a site where ATLAS s/w is present and the requested amount of computation (expressed in  $\text{SpecInt2000} * \text{Time}$ ) is allowed. An extra requirement is outbound connectivity, necessary for data staging.

The actual transformation is wrapped in a script that performs various tasks:

- Check the ATLAS s/w installation on the WN
- Download and install pacman
- Download and install the pacman package for the required transformation
- Set up the ATLAS s/w environment
- Stage in the input files
- Perform the transformation
- Stage out the results

For data management, a central server (Don Quijote) offers a uniform layer over the different replica catalogues of the three Grid flavours. Thus all the copy and registration operations are performed through calls to Don Quijote.

The software distribution is implemented using the tools provided by LCG, customized as foreseen in the tools.

### **Statistics**

The Phase 1 of the ATLAS DC2 started in the first week of July and is not yet completed; here the status at September 7<sup>th</sup> is presented

#### *Phase I (as at September 7<sup>th</sup>)*

Successfully completed jobs	36k
Typical job duration	1 day
CPU used	390 MSI-2k hours

#### *Failures and Efficiency (From August 1st to September 7<sup>th</sup>)*

The following estimations are done starting from August 1<sup>st</sup> because in the preceding part of the DC2, the ATLAS treatment of the failures was not yet detailed enough.

Preliminary Observations on LCG-2 Based on the 2004 LHC Data Challenges  
Grid Application Group

In this period a total of about 47.5 k jobs were submitted, with the following result:

Jobs DONE (OK)	29303
Jobs ER_MIS	750 (failures due to misconfigured sites)
Jobs ER_WLMS	1600 (failures in the WLMS or related LCG services)
Jobs ER_DMS	4350 (failures in the Data Management or Storage Element or related LCG services)

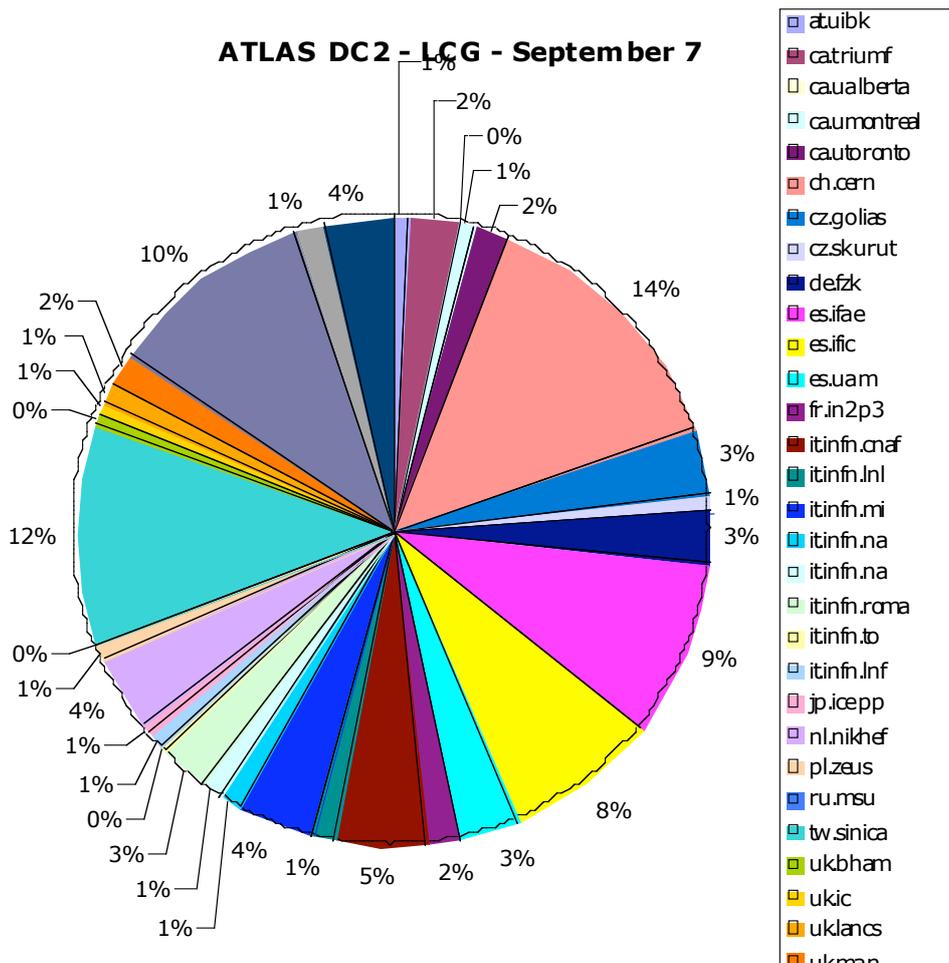
So, the overall efficiency of LCG service (WMS + DMS + SE + Sites) can be expressed as:

$$\text{DONE}/(\text{DONE}+\text{ER\_MIS}+\text{ER\_WLMS}+\text{ER\_DMS}) = 81\%$$

However, the overall efficiency of the system is given by:

$$\text{DONE}/(\text{Submitted}) = 62\%$$

This last figure however includes failures that are due to the ATLAS production system or DQ (about 7000) and failures that are difficult to trace back (about 4500).



## 2.4 LHCb

LCG-2 is being used in the LHCb production Data Challenge 2004 (DC04) through the LHCb production system *DIRAC*. *DIRAC* is also used on so-called *DIRAC native sites* where LCG-2 is not installed or in parallel with LCG resources.

The paradigm used on LCG-2 by *DIRAC* is the following:

1. The production manager enters jobs into the *DIRAC* WMS, independent on whether they will end up on a *DIRAC* native site or a LCG-2-site. Typically these jobs are simulating and reconstructing events. The jobs need about 12-48 hours Wall clock time depending on the type of events being simulated and computing power of the WN.
2. A cron job is submitting *DIRAC* agents to the one of 4 RBs (2 at CERN, 1 at RAL and another at CNAF). The jobs consist of a small shell script.
3. When jobs are scheduled to LCG sites on a Worker Node (WN), the script first downloads (using http) a *DIRAC* tarball and deploys a *DIRAC* agent on the WN. A *DIRAC* agent is configured and executed. This agent is the same as those running on *DIRAC* native sites. It requests the *DIRAC* WMS for a workflow to be executed. If any job is matched to the originating site<sup>1</sup>, the workflow is downloaded on the WN and executed there.
4. Software is normally pre-installed with the standard LCG software installation procedures. If the job lands on a site where software is not installed, then installation is performed in the current user's work directory only for this particular job.
5. All data files as well as logfiles of the job are produced in the job current directory. Typically the amount of space needed is around 2 GB plus an additional 500 MB if the software needs to be installed.
6. The bookkeeping information (file "metadata") for all produced files is uploaded for being inserted into the LHCb Bookkeeping Database (BKDB)
7. At the end of the reconstruction, the DST file(s) are transferred by GridFTP to the SEs specified for this site, usually an associated Tier1 centre and Tier0-CERN (only CERN if Tier1 is CERN).
8. Once the transfer is successful, the replicas of the DST file(s) are registered into the LHCb-AliEn file catalogue and into the replica table of BKDB. This was done for redundancy and in order to acquire experience. Both catalogues can be accessed via the same *DIRAC* interface and can be used interchangeably.

Before production is allowed on a site it is tested with production-like jobs.

The LHCb DC started in May 2004 with running mostly on the *DIRAC* native sites. Progressively, more and more LCG sites were and added to the LHCb

---

<sup>1</sup> The *DIRAC* WMS has an acceptance site mask as well as a matching mechanism for resource requests

DC pool of sites. The LCG share in the total volume of produced data grew from 11% in May to 73% in August. In the end of this period up to 3000 jobs were executed concurrently on LCG sites. The total of 43 LCG sites were executing (at least one) LHCb jobs with major contributions being from CERN, RAL, CNAF, NIKHEF, PIC, FZK. A total of 211k jobs was submitted to LCG, 26k were cancelled by LHCb after 24-36 hours in order to avoid the expiration of the proxy. Of the remaining 185k, 113k were regarded as successful by the LCG. This is an efficiency of ~61%. A breakdown of the performance is given in Table 1. A further breakdown of these 113k jobs was made and are summarised in Table 2. The initialisation errors included missing PYTHON on the worker node, failure of DIRAC installation, failure to connect to DIRAC server and failed software installation. The application error is a misnomer as it includes errors not only with the LHCb software but also hardware and system problems during the running of the application. The errors while transferring or registering the output data were usually recoverable. The bottom line is that LCG claimed 69k jobs produced useful output datasets for LHCb but according to our own accounting system there was 81k successful LCG jobs that produced useful data. This is interpreted that some of the LCG aborted jobs did run to completion and some jobs that were marked as not running did actually run unbeknown to the LCG system.

	Jobs(k)	%Submitted	%Remaining
Submitted	211	100.0	
Cancelled	26	12.2	
Remaining	185	87.8	100.0
Aborted(not run)	37		20.1
Running	148		79.7
Aborted Run	34		18.5
Done	113		61.2
Retrieved	113		61.2

Table 1: LCG performance as measured by LHCb

	Jobs(k)	%Retrieved
Retrieved	113	100.0
Initialisation error	17	14.9
No job in DIRAC	15	13.1
Application Error	2	1.8
Other Error	10	9.0
Success	69	61.2
Transfer Error	2	1.8
Registration Error	1	0.6

Table 2: LHCb analysis of output sandbox

LHCb experienced a series of problems that could take several weeks to be fully understood and solved. An intermediate report that listed a snapshot of the problems as of the end of June was produced; many of the issues raised

had solutions implemented by the GDA team. The DC on LCG was at that point interrupted for 2 weeks in order to fix the most burning problems.

### **3 Problems encountered**

There is a general problem related to the information flow between LCG and the experiments. Problems are reported and, usually, when solved, users get just a green light without an explanation of what caused the problem. Since it is likely - it happened several times - that a problem shows up again at a different site, this lack of feedback slows down the reaction as the information about fixes is not shared.

#### **3.1 Experiment software installation**

The tools for installing and maintaining application software on remote sites are still somewhat rudimentary. The system is built around the concept of a 'software manager' who has special permissions on Grid sites. These permissions allow the manager to install software in an experiment-dedicated (shared) file system as well as to publish run-time environment tags on the site's CE's. Scripts are provided to both install the software (running commands on a LCG UI) and to publish the run-time tags. ALICE and LHCb decided to use a different installation approach and send "installation jobs" that use the LCG script to publish the software TAGs. ATLAS is using a customised version of the LCG installation script, while it is publishing TAGs with the standard script.

The following issues have been identified as limitations of the current approach.

- a. The lack of "roles" severely constrains the abilities of software managers. These people would like to 'turn on' the software manager role and submit high-priority installation jobs, then turn off the role and start to submit normal user jobs conforming to the site's normal VO priority. In the current system, software managers are forced to accept either the ability to submit a limited number of concurrent high-priority jobs, or the ability to submit large numbers of normal user jobs, but not both. Most software managers accept the latter, which means that their installation jobs receive no special priority and may take days in some cases before they run.
- b. Since only the experiment's Software Manager certificate is allowed to write onto the shared software area, a "generic" job cannot trigger an installation.
- c. Many failures come from loss of visibility of the NFS file system either during software installation or during run. As noted previously, this happens relatively more often on larger sites and hence indicates a likely scaling problem warranting further investigation.

#### **3.2 Local site configuration**

General experience was that site local configuration was responsible for most of the problems that lead to job failure. In general individual sites are of a

heterogeneous nature e.g. worker nodes with different processing power and this created difficulties with publishing information to the Grid. A few examples:

- a. Missing or mis-configured experimental software areas
- b. Missing or mis-configured user working directories
- c. Inconsistent and/or missing information published in the IS
- d. Tag management tools unavailable from WN
- e. A large clock skew preventing any operation of the WN
- f. Intermittent authentication problems
- g. Missing generic software (e.g. PYTHON).
- h. Environment variables not (properly) set.

### **3.3 Information System and Monitoring**

While the experiments observed some problems that could be attributed to the Information System, no specific cases have been reported. The reason is that the Information System was not used directly by the experiments, only via other services, such as the WMS. It is thus possible that some of the WMS problems described below should rather be attributed to the Information System. However, the experiments had neither tools nor the expertise to distinguish them. The ability to query jobs by class e.g. by distinguishing name will also be extremely useful.

Monitoring services were largely insufficient: the most serious deficiency being absence of tools to monitor individual job performance/status. A problem is that it is impossible for a person other than the job owner to monitor the status of the production jobs. Similar problems effectively forced all the experiments to introduce other ways of monitoring, e.g., by the means of the central production database, which is far from being an optimal solution.

Access to relevant log files is virtually impossible. This makes debugging problems extremely difficult and time consuming.

### **3.4 Workload Management System**

Our experience with the WMS is that we encountered problems mainly in the areas of performance and workload sharing. The WMS needs a system to achieve a reasonable distribution of jobs. Right now, the RB can rank on a single number. The default number is the Estimated Response Time (ERT), which in principle is the correct thing an experiment would want, but in practice is quite inadequate for two reasons:

- a. A weakness of the algorithm for computing it.
- b. The lack of ability of the information system to express various values of the ERT, corresponding to the various user groups (e.g. experiments or VOs) using the system.

Multiple numbers (one per experiment or VO) are needed for the ERT since

each experiment generally has a different priority or quota on each site. Hence a site could be full for LHCb but still have CPUs available for CMS; the ERT for LHCb will be larger than for CMS. When a site can publish only a single ERT number per queue, this number is generally accurate for only one VO, incorrect for the rest.

These problems lead to a large number of troubles for the actual execution of the data challenges. The solution converged to by LCG was to implement a dedicated queue per VO, which is a workaround for b. above: there is only one VO per queue so one does not need to express multiple numbers for the queue. Also limiting to one queue reduces the problem space for point a. to a reasonable level.

These solutions should be viewed as workarounds, as single queues per VO will ultimately not work, especially if VO-internal priorities are introduced as required in HEPCAL-prime. The WMS needs to be able to make a reasonable distribution of jobs without manual intervention by the users and site administrators. Within the single-number ranking model, this seems to require at least fixing the information system to be able to present multiple ERTs per queue, and work on the CE side to present reasonable ERT numbers in a multi-VO, multi-user groups environment with diverse system policies.

The main problems encountered with the LCG WMS are

- a. It is still the case that we are lacking adequate tools for submitting, retrieving and checking the status of a large number of jobs. Most commands work only serially, i.e. one instantiation per job. The ones that do allow a list of jobs as input are often not robust (e.g. retrieve non terminated jobs). Capability of treating bulk operations will be necessary.
- b. Response from WMS could be improved: typically it is 5" for all operations, 15" for jobs to be scheduled (but this is partially piped). ATLAS experienced up to 30" response time with very heavily loaded brokers. This is particularly painful due to the lack of bulk operations. As soon as a few hundred jobs are in the WMS, it becomes extremely slow. The load on the WMS node goes up to 100%.
- c. Currently the job distribution is often very uneven. Due to the problems with ERT mentioned above, the site ranking does not evolve in a way that reflects the actual occupancies and remaining capacities of the sites. As an initial workaround, experiments have investigated with different "private" ranking algorithms.
- d. After arriving at the execution site, jobs often sit waiting in the queues for a relatively long time (sometimes tens of hours), even though there are free CPUs listed for the site. This is a side effect of the ERT problem mentioned above, being related to an experiment's CPU quota being saturated coupled with the impossibility of expressing this fact in the current information system schema. Another observed problem with the experiment workarounds is that smaller sites never get jobs until the largest ones are filled or almost so.

- e. Jobs are often reported as “Running” by LCG while they never perform the first operation. This problem is hindering use of the full computing power available. In at least few occasions it has been diagnosed as a LRMS problem, solved restarting the PBS scheduler on the Computing Element.
- f. LCG does not enforce Normalised CPU Units as time unit and has no facility for dealing with heterogeneous clusters. Hence depending on the site, MaxCPUTime has different meanings. The GLUE schema used for publishing this information again allows only one number for floating-point capability, and there is ambiguity between sites as to how this is dealt with – does a site pick the average computing power per node? The power of the slowest node? The fastest? The experiments developed several partially successful workarounds, but a unified solution is desired.
- g. When a job exceeds its queue’s WallClockTime or CPUTime, it is impossible to retrieve the stdout/stderr of the job and therefore to distinguish this from more serious problems
- h. Jobs aborted by “proxy expired”. Many jobs (submitted with a valid 3 days proxy) got aborted much before the 3 days were elapsed. The WMS was re-using old proxies, discarding the proxy attached to the job. A work-around is to set the maximum number of jobs sharing a proxy to 1. On suggestion of LCG people ATLAS started using MyProxy, and this seemed to solve the problem.
- i. Jobs are unduly cancelled by the WMS. Aside from concerns about why so many jobs are cancelled, there are two other problems. First is that while the jobs are officially reported to the WMS as cancelled, killed, or sometimes even cancelled by user, the jobs actually continue to run on the worker node and possibly even register output. In this case one cannot blindly rely on the ‘cancelled’ status of the WMS as the job may have well done its work as assigned. The second problem is that the WMS by default will resubmit one of these jobs. In case the job has already done significant work (like registering output) it may not be appropriate to simply resubmit, as it could generate duplicate entries in tables and ambiguity in choosing the correct output.
- j. WMS lost control of the status of the jobs: in fact the WMS seemed stuck, not submitting, nor responding to any request (from e.g. the UI machine). Usually the problem could be fixed without loss of jobs by contacting the WMS administrator to restart the service. A similar phenomenon was observed but related to very rare problems with the L&B server preventing some components from pushing events regarding job status changes. This resulted in jobs blocked forever in the waiting status. Although the job outputs could be retrieved via the Grid, it was impossible to manually advance the supervisor state machine.
- k. There is a bug in the WP1 API that causes a crash after several days of uninterrupted running. The problem was a file descriptor leak, now fixed. Some smaller memory leaks still exist, but the system is restarted

often enough for upgrade, database maintenance and so on, to prevent them to become an issue.

1. It happened a few times that jobs stay in "Running" status forever, even when successfully completed according to reports by site managers and information in the logging system.

Overall we can say the following

- a. The WMS is reasonably reliable. However there is no way to submit a large number of jobs in a short amount of time because it does not have provisions for submitting many jobs with a single operation and it does not sustain the load when several single jobs are submitted in rapid succession. This provides an effective limitation to the maximum number of jobs that one can introduce into the system via a single RB. The under utilisation of the RB can be partially explained by the previously discussed ranking problems. The experiments had to use several RB's to solve the load problem. In other words, a RB can serve a finite amount of resources, which is roughly defined by the ratio between the average job duration and the time required for submission.
- b. The WMS is missing essential functionality particularly in the area of workload optimization. Users should not care at all about ranking, and the system should provide proper job distribution.<sup>2</sup>
- c. The configuration of the WMS seems to be very delicate in general. This is almost wholly related to the design of the CE information provider, which is not robust against site misconfigurations. A wrong configuration produces apparently valid information system values that result in large numbers of jobs being attracted to the site, that will subsequently not run at all, wait in the queues for days, or abort and fail quickly, attracting more and more jobs and failing them - the *black hole scenario*. A possible workaround for the "sink hole" effect is to leave a sleeping job on the misconfigured WN; this is only valid though if the job starts to run.

### **3.5 Data Management**

These are only preliminary remarks, since we have less experience with the Data Management than we have with other components of the LCG-2 middleware such as the WMS, and our understanding is not yet consolidated (ALICE uses Data Management and SE in phase 2).

- a. For each site, a "default SE" is defined for the replica manager commands. One or more Close SEs are defined for each CE within the information system. The purpose and definition of these are not

---

<sup>2</sup> Note however that the worst problem experienced by ATLAS generating CPUs under-utilization was related to the ORACLE DB used for production jobs, which behaved so badly and with so slow response time that for about 10 days the speed of the DC2 on all the 3 GRIDs used by ATLAS, was reduced by almost a factor 3. This problem is obviously outside the LCG realm but is mentioned here because it affected ATLAS DC more than any other external software problem.

entirely clear, and much confusion resulted from a lack of clarity about the normal model for using these SEs with the DMS and WMS commands. The workaround provided by LCG, an SE environment variable *per site per VO*, ultimately does not scale and furthermore it makes sense only for sites with multiple SEs.

- b. There are two sets of LCG data management tools, starting with prefixes “edg-” and “lcg-”. There was some confusion about which to use, and neither of them were adequate to the scale of the data challenges. ALICE started with edg-rm commands and recently moved to lcg- commands, following the suggestion of the LCG support. “edg-rm” commands often stayed hanging, thus stopping the job execution for hours. After moving to lcg-\* commands, ALICE observed that the lcg-cr command (the most used) is hard to debug as, when failing, it always returns the same message (“invalid argument”). Also, the fraction of failures is not uniform for different sites.
- c. Sometimes the job is running in a local directory that doesn’t have enough space (typically 2 GB). Probably a system of reservations and quotas is necessary for local space on WN’s.

### **3.6 Misconfiguration problems**

Here we list problems due to site misconfigurations that have been observed to introduce a certain overall fragility in the system.

- a. Experiment Software Area configuration had a lot of problems in the beginning. The area is usually NFS-exported to the WNs; it happened several times, particularly for sites with many CPUs, that the area was unreachable from all or some of a given site’s WNs (e.g. because of NFS crashes). This caused the site to appear as perfectly good, but all jobs going there to fail very quickly, so that the site continues to get lots of jobs and waste them (the dreaded “black-hole” effect). NFS mounts and the availability of that area from the WNs should be closely monitored. Furthermore, not all sites have consistent policies for writing on it, or for publishing tags, so some sites need special care to install software there. Other sites (e.g. CERN) only allow installation on AFS, which cannot be performed as a job.
- d. Many Data Management tools and utilities are very slow in providing response. This however was not critical, as the job execution times were typically much longer.
- e. On several occasions, Storage Elements ran out of storage space, which caused jobs either failing, or being aborted by system administrators in need of urgent maintenance.

### **3.7 Miscellaneous operation problems**

- a. LDAP of globus-mds servers stop on the site. No jobs can be submitted to that CE while the condition persists. Furthermore (and more seriously) is that the information system contains some important information about the SE, needed by the data management

tools for proper functioning; hence as long as the site's info system is down, data management commands involving the site will fail. Fortunately this problem is not very frequent. Usually solved by contacting the site admin until he restarts the thing.

- b. Sometime globus-mds is stuck in a state and always is reporting the same status.
- c. Jobs return an empty output sandbox. There is no way to figure out what happened to the job.
- d. Response from site administrators sometimes less effective and prompt than central support. Note that this can be also considered a measure of how good central support is. However the point here is that the reaction of central and remote support should be synchronised.
- e. Relevant log files with appropriate information should be available for efficient debugging of problems.

## **4 Requirements**

### **4.1 Experiment software installation**

- a. The experiments believe that a software distribution and installation model that does not necessarily use regular batch jobs would be a better solution to this problem. The main requirement here can be summarized as: Install just once and have the software automatically replicated or removed by the system where needed. The outline of such a solution is outlined in HEPCAL-prime 3.7.
- b. The software installation model should allow some customization, such as specific pre-, post-installation and validation scripts.

### **4.2 Local site configuration**

- a. Better control of the local configuration and stability;
- b. Middleware, EIS and GDA should aim for more "passive" designs where stability is intrinsically built-in. For example we mentioned above the problems with the ERT and other information published by a computing element node. The information provider publishing these values should be written so that by default a site has zero CPUs, maximum wall clock time of one minute, and estimated response time of  $2^{16}$  seconds. The input needed from the configuration files and LRMS response should be rigorously checked, and the default values filled in with those from the calculation only after passing all checks. In this way, a misconfigured site would not pull any jobs as it will report having no resources and a very large (and very suspicious looking) ERT. These design principles could be used in many other areas as well.

### **4.3 Workload Management System**

- a. Bulk operations for job submission, monitoring and management

should be supported.

- b. Response time of the WMS to accept a job submission request should be improved and should gracefully degrade under load. This requirement may be partly satisfied by proper support for bulk operations.
- c. WMS should ensure fair share and efficient usage of resources. Usage of available resources should be maximised, with minimal waiting time in the queue without user intervention on ranking algorithms.
- d. Some experiments require being able to control some of the parameters of job submission (ranking, destination CE and so on), but this should be the exception rather than the norm, and the default rank should give the best results under normal conditions.
- e. The logic of assignment of jobs to a site should be such that *black holes* are avoided. If a site starts failing all jobs submitted, it should be excluded from the possible choices.
- f. It should be possible to specify the resource requirements of a job in normalised units.
- g. It should be possible for a job specify the amount of space needed on the Worker Node.
- h. The middleware should provide a way for jobs to gracefully terminate if they have exceeded the allowed CPU or Wallclock time. Jobs should properly save output. Output sandboxes should be retrievable by the users.
- i. Error reporting should be improved. One example is the “no compatible resources” error coming from the job submission tool. This error could mean that the user has specified a non-existing CE in her JDL; that the CE should exist, but is down; or that the information system is empty or unreachable. In case of errors, the first caught error is usually much more meaningful than the one returned by the wrapper of the wrapper of the service.
- j. Monitoring should be improved. One should be able to easily get a list of jobs, sorted and/or selected by user-definable criteria (default being time submitted to the WMS), in a tabular format with clear, concise status codes. It should be able to have the monitoring on a component level e.g. CE level, Job level, RB level, WN level. Perhaps one could take a lesson from the old VMS command:

```
job-status --since=yesterday --by_owner=[ATLAS,prodman]
```

#### **4.4 Data Management**

- a. A job running on a CE must have a simple way to determine which the default SE is for that CE. It must be possible to indicate a different SE, but this should be the exception and not the norm.
- b. When scheduling a job requesting input, also sites that do not hold a copy of the requested data should be considered while deciding where

to send the job, taking into account file pre-staging costs with respect to a site where the input data is already present.

- c. If a job requests the output to be saved on permanent storage, it should be the RB responsibility to find a site with a reasonable SE to hold the job output or to find out a list of possible candidate SEs. The related information should be available to the job.
- d. It should be possible to have the following use case.
  - a. The user specifies in the JDL only the input LFNs (or GUIDs or Collections) and the size of the output data.
  - b. The system then finds the SE's which hold the input and the SEs on which the output files can be saved. It then finds a CE that can "conveniently" access both a SE with the input files and a SE where the output data can be stored.

If not all files are on the selected SE, they should be all replicated on a given SE before the job is submitted to the selected CE. (This is in fact explained in HEPCAL-prime in the Data Transformation use case).

## 5 Generic Efficiency Tests

We report here the results of some overall system efficiency tests done by S. Burke. All the experiments consider these results a good reflection of their experience. For these tests, a test job has been submitted to LCG once an hour; this summarises the outcome for the ten-week period starting at 13:00 UTC on Monday 5th July 2004.

The job is targeted at an input file replicated to almost all SEs, with a Rank of 1 so it should go randomly to matching CEs. It checks that it can read the input file with `rfio`, registers a file to the close SE and then replicates it to a randomly-chosen remote SE, and finally copies it back to the WN. The analysis is largely done using the registered file names to avoid having to wade through thousands of log files. The job is submitted with a cron job on the RAL UI via the RAL RB.

The job only runs for a few minutes, so it does not test proxy renewal or stability over long periods. It is submitted under the `dteam` VO and hence does not test the configuration for other VOs. It does not require significant resources in terms of disk space, memory or I/O, or any non-standard software to be installed on the WNs. It currently uses `edg-rm` rather than `lcg-*` for replica management.

One job an hour for ten weeks should be 1680 jobs. However, 28 were not submitted because the UI was rebooted, and for some reason the cron job does not restart automatically. The broker rejected a further 131. 16 of those were sporadic failures, but there were also three periods of a day or two when all or most submissions failed. One of those was due to an incorrect host certificate on the myproxy server, one was probably a side effect of bugs in a modified `qstat/qsub` wrapper and one may have been related to the upgrade to version 2.1 which happened during the period (the latter two occurred at weekends and the first while the sysadmin was on holiday).

Of the 1521 submitted jobs, 195 were aborted. 169 of those were "cannot plan", i.e. a BDII failure. These again happened in bursts when the BDII was unstable. The last of these was in the week starting August 16th; it is claimed that the BDII stability should be improved now. 18 had proxies expire due to a proxy re-use bug, which has since been fixed, the other 6 were miscellaneous failures. One more was still listed as scheduled, but in fact had failed to download the input sandbox.

Of the 1325 completed jobs, 65 had a non-zero exit code, indicating that no file registration even to the local SE was possible. The reasons have not been analysed in detail, but appear to be due to a wide variety of local configuration and operation problems.

Of the jobs that ran at least at the level of being able to write an output file to the close SE, there were 293 where the input file read failed. Most of these were either because rfio wasn't working or because the file had been deleted from the SE by the sysadmin, although some may have been because the SE was down or the disk was unmounted. rfio is supposed to be supported but is not widely used and hence is not regarded as a priority, probably a higher efficiency would be seen with gridftp. Deletion of files on an SE, e.g. if it gets re-installed, is an operational issue: in general sysadmins should try to avoid losing files, and should announce the fact if it's unavoidable.

There were also 144 cases where the replication to a remote SE failed. That may be because of an information system problem (typically a missing CESEBind), or one of the many things that can cause gridftp to fail. In production use it would generally be possible to try a different SE if one fails, so this is a measure of the general stability of SEs in the system rather than a real inefficiency.

In 22 cases a job had both input and output failures. This might suggest that the number of fully successful jobs would be  $1325 - 65 - 293 - 144 + 22 = 845$ . However, some jobs run successfully in terms of registering files but are seen as failures by the broker, usually because of the infamous "Maradona" error (job output not retrievable). The jobs then get resubmitted, so effectively they may run twice or more. Overall, 67 jobs ran twice in this sense, 9 ran three times and one ran four times. Some of those had input or output file failures counted above. The final count of jobs that ran only once and showed no errors is 826, although this may include jobs that were resubmitted (and hence had no retrievable output sandbox) but where the resubmitted job did not run. This would lead us to conclude an overall efficiency around 60%. This number is very close to what the LHC experiments find. A closer scrutiny of the different factors however shows that they contributed differently to the percentages found, and therefore it seems premature to draw any conclusion.