Running a Production Grid Site at the London e-Science Centre

David McBride, Marko Krznarić,
John Darlington
London e-Science Centre
Department of Computing
Imperial College London
{dwm, marko, jd}@doc.ic.ac.uk

Olivier van der Aa, Mona Aggarwal,
Dave Colling
High-Energy Physics Group
Department of Physics
Imperial College London
{vanderaa, m.aggarwal, d.colling}@imperial.ac.uk

Abstract

This paper describes how the London e-Science Centre cluster MARS*, a production 400+ Opteron CPU cluster, was integrated into the production Large Hadron Collider Compute Grid. It describes the practical issues that we encountered when deploying and maintaining this system, and details the techniques that were applied to resolve them.*

Finally, we provide a set of recommendations based on our experiences for grid software development in general that we believe would make the technology more accessible.

1. Introduction

The **London e-Science Centre** (LeSC) at Imperial College London [21] serves two roles: the first, to perform research into large-scale distributed computer systems ("Grid" systems); the second is to provide a production high-performance computing service to the Department of Computing and other researchers at Imperial. These two activities are complementary; the practical understanding developed from running production computing facilities guides local research activity, whilst developments from that research can be tested and deployed on production systems.

The **Large Hadron Collider** (LHC) [17] is a new particle accelerator currently being constructed by an international consortium at CERN. After it goes into operation in 2007, it will generate on the order of 15PB of data per year — a quantity of data that simply cannot be processed in any reasonable time-frame at CERN itself.

The **LHC Compute Grid** (LCG) [20] project was created to solve this problem. Its goal: develop, distribute and deploy a standard software distribution on cluster systems at cooperating institutions around the world to build a new production Grid system sufficiently powerful to meet the demands of the LHC. Today, the worldwide LCG deployment currently spans some 136 sites in 36 different countries, offering to its users access to a total of nearly 14,000 CPUs and approximately 8PB of online storage.

This paper describes how we integrated the LeSC MARS cluster into the production LHC Grid. It describes the practical issues that we encountered when deploying and maintaining this system as well as the techniques that applied to resolve them.

2. Overview of the LCG Architecture

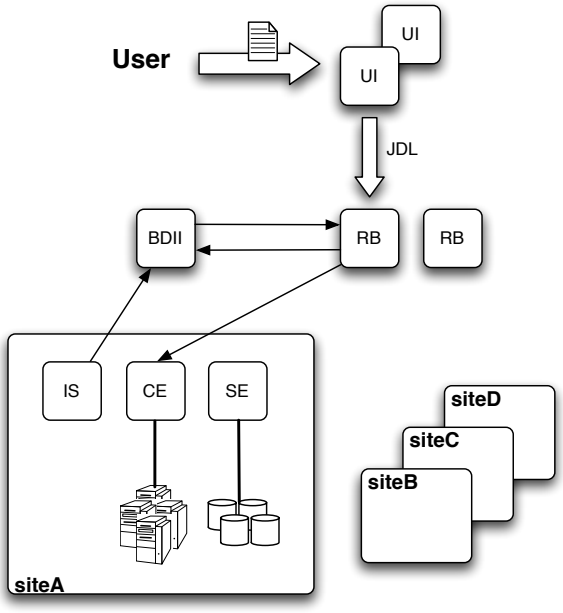

Figure 1. LCG Architectual Overview

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

The LCG software distribution is made up of many individual pieces of software drawn from external sources, such as the Virtual Data Toolkit [26] hosted by the University of Wisconsin and tools developed as part of the European DataGrid (EDG) [7] project. The organisation and function of the LCG software components is shown in Figure 1.

2.1. Computing Element (CE)

The function of a Computing Element (CE) is to provide a standard network-accessible interface to a site's local computer cluster. When the CE receives a properly authenticated job execution request, it will add a suitable job entry to the local cluster's batch queue, and report on the job's progress through the system during its lifetime.

2.2. Storage Element (SE)

A site Storage Element (SE) provides a standard network accessible interface to a large file storage system, whereby any properly-authenticated processes may retrieve or upload files from any location. Whilst many SEs simply provide access to a local online disk array, some sites also provide access to large high-capacity tape libraries.

2.3. Information System Services

There are a number of Information System (IS) services operating at an LCG site, typically located on a separate Monitoring (MON) server. Their function is to collate information from the various services running at a site — such as the number of jobs currently running, pending, etc. and the amount of storage space available — and report that information to a set of centrally-accessible Information Index (BDII) servers.

2.4. Resource Broker (RB)

A job starts with an end-user, who submits a job-execution request to their local Resource Broker (RB) from a User Interface (UI) node. This request is expressed as a Job Description Language (JDL) document, and includes various important details such as the name of the executable to be run, the names of the data files that should be staged to or from the execution host, the job's memory requirements, and so forth.

The Resource Broker will validate the user's credentials and, if satisfied, will use the status information reported by the Information System services at each site to dispatch the job request to the most suitable target.

First, it excludes those site queues which do not satisfy the job's hard prerequisites. Out of those site queues remaining, it constructs a simple ordering using the Estimated Response Time (ERT) metric advertised for each queue.[1]

3. Adaptation of the Compute Element

The LCG software distribution was originally designed to support a single set of users: namely, the particle physicists who have just taken receipt of a several-hundred node compute and storage cluster — but who lacks the systems administrations experience required to deploy a new cluster infrastructure from scratch.

Thus, the LCG software distributions were designed accordingly — it provides everything that a LCG cluster would need, from the cluster management system on up through the stack, all pre-compiled and ready packaged for use on any of LCG's supported Linux distributions.[2]

However, this software suite is not well adapted for another less-common set of users: namely, those experienced systems administrators who wish to contribute some fraction of their existing production resources to the Grid effort. Here, the one-size-fits-all model can fail; even if the sysadmin extracts the extra services that she lacks from the official LCG distribution to layer on her existing system, she will most likely find that the interfaces used by the standard service implementations are incompatible with their equivilents already running on her production service.

Indeed, this was exactly the case at LeSC. The MARS cluster is made up of just over 200 64bit Opteron servers, each of which running a 64-bit/32-bit dual-arch revision of RedHat Enterprise Linux 3. On top of this, the GridEngine batch-job control software and a locally developed management infrastructure supported the day-to-day operation for hundreds of registered users.

Discarding all of our existing infrastructure so that we could participate in the LCG simply was not an option. Instead, it was necessary for us to break apart the LCG software distribution to modify (or replace outright) the implementation of the incompatible components so that it could be spliced into our existing cluster environment.

3.1. JobManager

The CE runs a Globus Gatekeeper. The Gatekeeper's function is to accept general-purpose RSL [23] job specifi-

[1] Strictly speaking, this ERT is only the RB's default queue-selection measure; it is possible to override this with much more sophisticated expressions in the JDL document for those jobs with unconventional run-time characteristics.

[2] Originally, LCG only provided binary packages for RedHat 7.3 — the only officially supported distribution at the time — and no readily-usable source-packages. This caused a lot of difficulties as our then-primary cluster, VIKING, was running another Linux distribution which was not binary compatible. Now that LCG have added support for RedHat Enterprise Linux 3, as used on MARS, we have bypassed this particular obstacle ... for now!

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

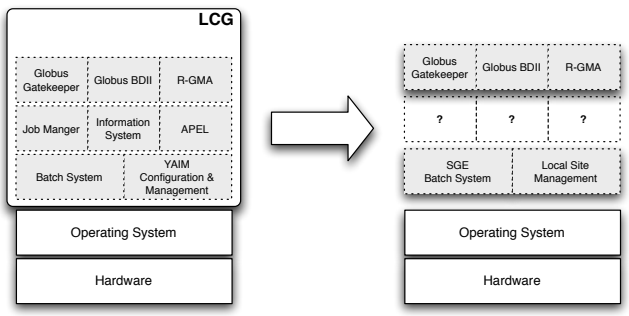

Figure 2. Adapting the standard LCG CE distribution to function on MARS.

cations, translate it into some suitable representation and submit the job into the local batch system. The batch-system specific implementations of these functions are provided by Gatekeeper plugins called JobManagers.

In the LCG software developer's model, the set of batch systems that may be installed is small and known — generally some variant of PBS [22] — and so they have provided customised JobManager implementations to support these specific batch systems. However, the LeSC MARS cluster is running a batch system that is not supported by LCG, namely Sun's GridEngine [13]. As a result we have had to develop and deploy our own JobManager implementation for GridEngine so that we could re-use the Globus Gatekeeper provided.

Fortunately, though Globus themselves do not distribute a GridEngine JobManager, we had developed one as part of our involvement in the Level 2 Grid deployment run by the UK Grid Engineering TaskForce (ETF) [19]. With only minor refinements we have found that it can provide the functional glue between the Gatekeeper and our batch system.

3.2. Information Systems

The purpose of the Information System processes on the cluster front-end machines is to periodically collect statistics from various local systems — such as the number of jobs currently waiting in each batch system queue, or the amount of disk space currently available — and publish them into a global hierarchy of information servers. This information includes an Estimated Response Time value for each of the site queues.

The information published by this system is vitally important; it is used by the Resource Brokers deployed at other sites to perform service discovery and to make scheduling decisions. Without a working information system, a cluster instance — even if otherwise fully functional — will effectively not exist.

The Information System processes on an LCG site all centre around the local BDII service - a simple LDAP server, backed by a Berkeley database, which stores the current state of the cluster as reported via various sensor processes running on the various site frontend machines.

Like the JobManager, the Information System operates through a plug-in system — in this case, the Generic Information Provider (GIP) [8] framework. The system operates by simply running a series of system-specific GIP plug-in scripts, called Information Reporters. Each Reporter script queries the local system it is responsible for and returns status data according to a standard format.

Because we were using an unsupported batch system, however, we needed to provide a GridEngine-specific GIP plugin that was capable of reporting cluster-state information in the expected form. Unfortunately, unlike the case of the Jobmanager there was no suitable pre-existing Information Reporter implementation that we could quickly press into service.

So, we naturally began work to develop our own implementation. However, these was an additional complication: the GLUE schema [12] that specifies the data structures to be used in Information System interactions assumes that a cluster manager's state can be expressed using a Queue-based model. In this model, a cluster's execution hosts are organised into a set of individual queues; a job, when submitted, specifies which particular queue it would like to be appended to. See Figure 3.

However, our local GridEngine installation had no such specific queues. (Indeed, prior to version 6, GridEngine didn't even support the concept of cluster-wide queues.) Instead, all cluster-local job scheduling operations are based on attributes attached to each individual job, such as their maximum wall-clock runtime, maximum memory usage, etc. To make matters worse, grid jobs submitted via a Resource Broker cannot be annotated with such information — the JDL language that end-users must use to express their job specifications is very limited and doesn't allow for the expression of such constraints. See Figure 4.

3.3. Virtual Queues

To work around this problem, we developed the concept of *virtual queues*. A virtual queue differs from a real queue in that it does not actually exist in the underlying batch system; indeed, a system which advertises virtual queues need not have any explicitly defined queues at all. The virtual queues themselves are defined in terms of job attributes; for example, the MARS cluster currently has a "10min" queue defined which includes all jobs with a maximum wall-clock runtime of 10 minutes or less. The queue also includes all of the cluster's execution hosts which are able to execute

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

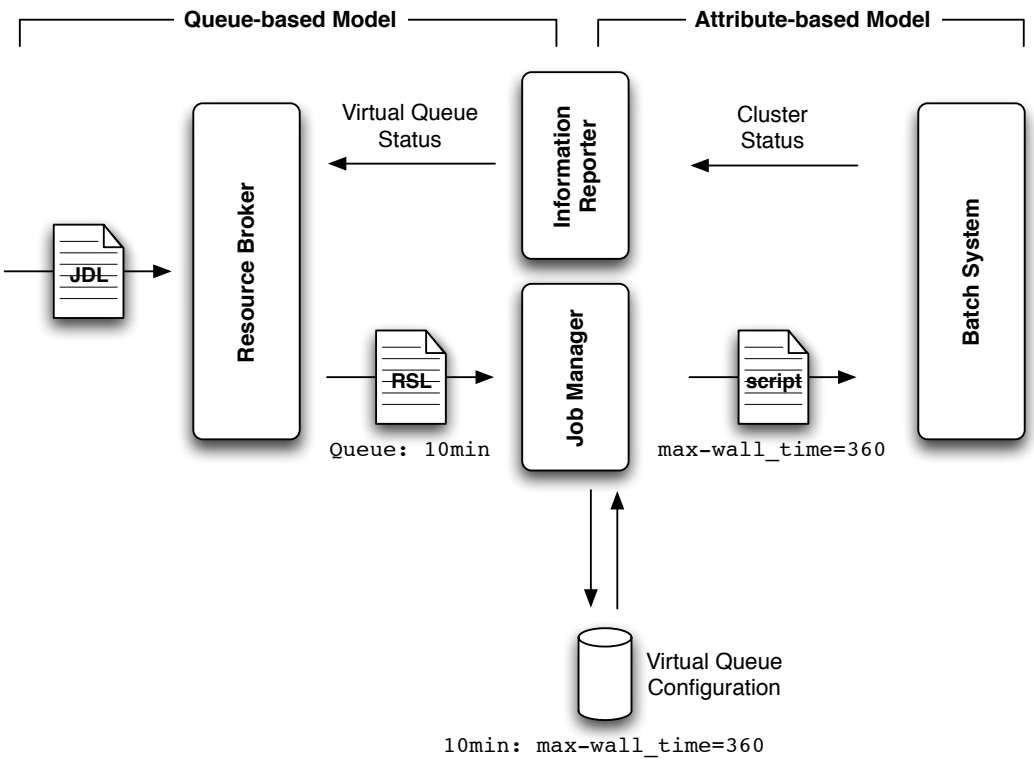

Figure 5. An attribute-based batch system running LCG jobs advertised using virtual queues.

such jobs.

Because grid jobs specify their job characteristics simply by indicating which specific queue they would like to be submitted to — as opposed to specifying the properties explicitly in the job's RSL specification — it is also necessary to extend the JobManager responsible for submitting the job. Happily, these modifications are relatively minor; all it needs to do is parse the set of virtual queues currently defined for the cluster and map the requested virtual queue name to the set of attributes that define that queue.

For example, when a Grid job arrives requesting the non-existent '10min' queue, it is simply submitted to the general job queue with a specified maximum wall-clock time of 10 minutes.

The disadvantage of advertising a cluster's state using virtually-defined queues is that the implementation of the Information Reporter must necessarily be more complex. For example, the ERT metric calculations are much more involved. Normally, the ERT for a queue is calculated as follows:

$$\text{ERT} = \frac{\Sigma_{i \geq 0} t_i}{2N} \ ,$$

where t_i is the maximum amount of execution time remaining for a job i and N denotes the total number of execution slots available.

At a conventional site, the Reporter can simply iterate through the state of each of the cluster system's real physical queues; with virtual queues, however, it is necessary for the script to calculate the job and host membership of each virtual queue manually before it can complete the calculation — a non-trivial exercise.

This can also give rise to some confusing-looking statistics; any given job, depending on its properties, can exist in zero, one or even all of the virtual queues simultaneously — which can violate assumptions made by some of the standard LCG monitoring tools. The GStat [15] tool, for example, used to calculate the total number of jobs currently running on our cluster by summing all of the running-job totals listed for each of our cluster's queues — leading to some wildly inflated numbers!

However, the advantages of advertising virtual queues greatly outweigh these inconveniences. Administratively creating or destroying even large numbers of virtual queues will have no impact on the cluster interface seen by our local grid users; similarly, our local cluster administrator is free to modify his local batch system configuration as necessary to meet the needs of our existing local production users without impacting incoming Grid jobs. The two concerns are

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

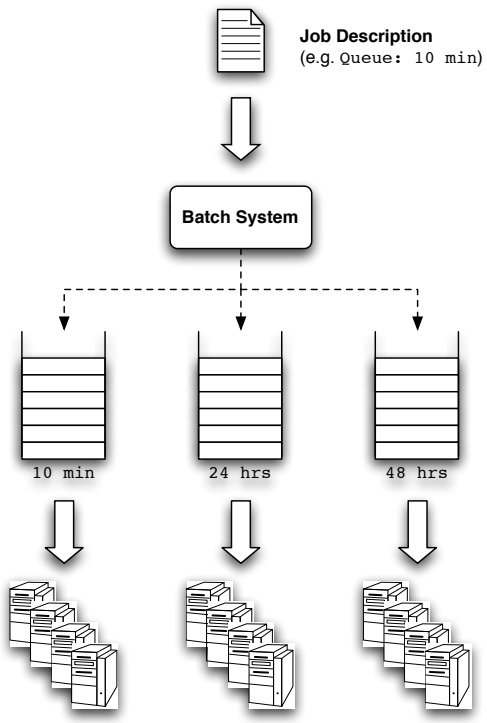

Figure 3. The Queue Model as used in the GLUE Schema

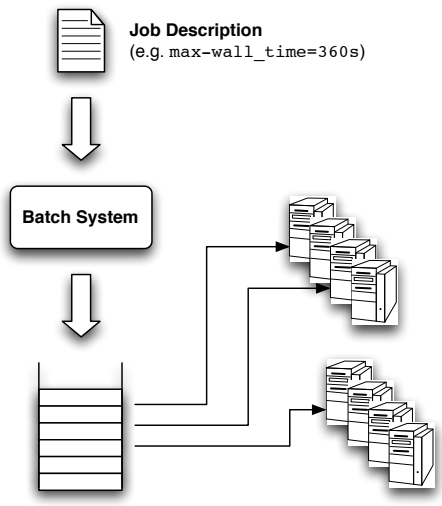

Figure 4. The Attribute Model, as embodied by the LeSC GridEngine installation.

entirely seperated, allowing a great deal of operational flexibility.

3.4. Accounting

There exists an accounting system in the standard LCG software distribution called APEL [1]. The function of the software is simple; it periodically reads through the batch system's accounting logs and matches them against the various global Grid user and group idenitfiers using the Globus gatekeeper's logging output. It then uploads this information to a central data store.

Unfortunately, the original implementation of the APEL software was very much specific to the PBS batch system as used by the standard LCG software distribution; no support existed for any other batch systems such as the GridEngine installation on MARS. (Indeed, APEL itself stands for "Accounting using PBS Event Logs"!)

As the LCG software distribution matured, APEL — though it kept its name — was re-engineered to support batch systems other than PBS. Earlier this year the APEL developers, in concert with CESGA, have completed a GridEngine-specific variant of APEL which, with minor modifications, is now deployed and operational at LeSC as well.[3]

4. Adaptation of the Storage Element

As part of our contribution towards the global LCG effort, we would like to provide access to some of our storage capacity in addition to our computing facilities. The storage services in LCG are based on the Storage Resource Manager interfaces.[4] The two SRM implementations in use at most LCG sites today are dCache [4] and Disk Pool Manager (DPM) [5]. Both of these SRM implementations are typically deployed on Linux storage servers; however, at LeSC, our primary storage arrays are all attached to Solaris mainframes. dCache is closed-source, and binaries for Solaris are not readily available which makes it unusable for a Solaris server deployment. The source for DPM, however, is publically available under a free license.

So we have been working on rebuilding DPM from source on our Solaris servers so that they can act as Grid-accessible storage systems.

[3] There exists, unfortunately, more than one GridEngine-specific JobManager implementation by different LCG sites — and more unfortunately, the CESGA test LCG installation used to develop the GridEngine APEL implementation used another variant that generates different log output to our own. As a result, the current official Java-based APEL implementation doesn't operate correctly out-of-the-box.

[4] There are actually two major versions of the SRM interface specification; at present, SRMv1 is the interface in current active use — though a migration to using the newer SRMv2 specification is planned.

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

This work is still ongoing; there have been various difficulties in building not just the DPM software itself, but its many dependencies — some of which are not as mature as would be desired. Until we can bring our own native SE online, we are using the SRM services provided by the neighbouring High-Energy Physics group.

A full log of the work as it progress (along with the 20+ patches generated so far!) can be found online at [6].

5. Monitoring and Maintenance

The monitoring of any infrastructure is vital to its continued productive operation; The LCG and the sites from which it is composed are no different. The LCG developers have thus developed a set of monitoring tools that continusouly test the availability and correct functionality of site services. Some of these tools generate automatic alarms, which are registered in the form a problem ticket in the Globus Grid User Support (GGUS) request-tracking system [10].

5.1. GStat

GStat (short for Grid Statistics) is a tool for monitoring the data published by a site's information system. It operates by periodically querying each site's BDII (Berkely Database Information Index) service and records the state of the site over time. It verifies that the information being reported by each individual site is self-consistent, properly formatted, and satisfies some basic sanity checks. For example, it will check that the number of jobs currently reported as queued in the site's batch system is not significantly larger than the total number of batch servers currently operational.

It also monitors site status over time; if the system observes a large variation in the number of services advertised by the site, it will flag up a warning — under such circumstances it is likely that either the information-gathering mechanisms themselves are intermittently failing or that services at the site are repeatedly crashing and restarting. All the site information gathered by GStat is accessible from a central web server [15], and has been invaluable as a diagnostic tool for identifying site configuration problems.

5.2. Site Functional Tests

Site Functional Tests (SFT) are used to test the functionality of the core LCG services running at each site. It does this by regularly sending test jobs to the site Gatekeeper. Each test job runs a standard battery of tests to exercise all of the facilities that a real end-user job may need to access. The tests include:

- Job Submission Status: Could the test job actually be successfully submitted via a Resource Broker, or did some error occur?
- File Replication Tests: A small test data file is created on the worker node and copied to the local SE; the file is then registered with the LCG File Catalog (LFC). The job then requests that the file be replicated to another remote SE, typically at CERN.
- CA Tests: The test verifies that the latest versions of each of the LCG Certification Authority's certificates is installed. Also, the certificate revocation lists are checked for freshness.
- VO tags: Each LCG Virtual Organisation may publish "tags" — simple strings — in the site's information system output. These are commonly used by a VO to indicate which version's of the VO's specialist software has been installed. This test simply checks that the interface for setting and checking VO tags is operating correctly.
- VO Software Volume: Each VO can install their software in a special dedicated volume on each cluster; this test simply checks this volume is accessible.

Each test can return different status results, ranging from "OK" to "CRITICAL", indicating a critical site failure. Critical site failures result in a ticket being automatically raised against a site; if the ticket is not resolved (or at least acknowledged) quickly, the site is at risk from being removed from the production grid until the fault has been corrected. Statistics regarding the number of SFT failures at each site are kept for further analysis on the Communication Interface for Central Operation (CIC) portal [3].

5.3. Grid Load

Monitoring the number of jobs in any given state in the whole grid is important for understanding how the system as whole is operating. The function of the GridLoad tool is to monitors all of the LCG Resource Broker and keeps track of all of the jobs' status over time in a local relational database. This database is then polled every five minutes and used to populate Round-Robin Database (RRD) files, from which plots of job states over time can be readily generated with RRDTool [25]. The user interface on the GridLoad webserver [16] allows for a variety of different views of the data, such as by VO or by CE. This facility has been useful for monitoring the job-abort rate at sites, and also for spotting CE gatekeeper problems quickly. Further details of this system are available in [2].

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE

5.4. Security

The LHC Grid's constituent resources present an enormously attractive target for attackers. The machines participating in the Grid tend to be homogenous; they all trust a global authentication infrastructure; they have access to very large quantities of storage capacity and network bandwidth - in addition to the massive processing power available. Moreover, the individual cluster machines are frequently run and maintained by inexperienced administrators, who may not be able to distinguish a normal operational load from a compromised system — especially if more sophisticated attacks are employed.

Monitoring of a cluster's current state is important; it allows you to see what the cluster's "normal" state actually looks like so that anomolies, when they occur, can be identified as such. Unfortunately, although the above tools exist for monitoring the load of a cluster or of the Grid as a whole, visualisation tools that show what individual jobs are actually doing on a cluster are not widely available. This is a topic of further investigation.

6. Recommendations for future development

The original aim of LCG was to provide a grid infrastructure to support a single application — the Large Hadron Collider. However, as the system has matured and expanded to encompass many high-perfomance computing sites around the world, it is now moving beyond the single-application focus and being deployed as a general-purpose grid computing platform — gLite [9].

As it continues to grow, we expect that interest from academic and commercial organisations will result in them wishing to join the growing system. However, they will be enjoined from contributing their resources if doing so would require that they are forced to scrap their existing infrastructure.

However, gLite's implementation must also continue to mature if it is to be adopted as the global standard for grid computing activities. Hence, we propose the following recommendations, tackling a number of issues addressed in this paper:

- Source code for LCG releases made more readily accessible, e.g. as source RPM packages.
- Better seperation-of-concerns between LCG components.
- Implementation of support for other batch systems as standard. (In fairness, LCG developers are now working on adding GridEngine support into the standard distribution.)
- Improvement of LCG auto-configuration scripts to be more fault-tolerant and system-agnostic.
- We would recommend that a stronger emphasis is placed on developing stronger security protections into LCG.
- Support for the specification of job attributes in JDL.

7. Conclusions

We have demonstrated that the principle of adapt-the-application, whilst often requiring more effort than a more typical dedicated-cluster installation, is workable and maintainable in practice — and almost certainly far less disruptive to our existing users than replacing most of LeSC core systems would have been. Indeed, apart from an increase in the number of jobs arriving on MARS, they have seen no changes at all.

In addition, these adapations have been made public for others to re-use, deploy and develop to satisfy their own operational needs. Indeed, there has been a lot of interest in our work from other existing LCG sites and other production cluster operators who, before now, were either prevented from deploying LCG at all or are keen on replacing the standard PBS cluster manager with a more flexible alternative.

For those interested in using LCG or gLite with Sun's GridEngine, further information and source code is available at [18].

8. Acknowledgements

This work was funded by the UK Particle Physics and Astronomy Research Council (PPARC) as part of the GridPP project [14].

References

[1] Accounting using PBS event logs (APEL). http://goc.grid-support.ac.uk/gridsite/accounting/.

[2] M. Aggarwal, D. Colling, B. MacEvoy, G. Moont, and O. van der Aa. A statistical analysis of job performance within the LCG grid. In *Computing in High Energy and Nuclear Physics*, 2006. http://www.gridpp.ac.uk/tier2/london/Data/lcgstat-chep06.pdf.

[3] Communications interface for central operation (CIC). http://cic.in2p3.fr/.

[4] dCache. http://www.dcache.org/.

[5] Disk Pool Manager (DPM). https://uimon.cern.ch/twiki/bin/view/LCG/DpmAdminGuide.

[6] Building Disk Pool Manager on Solaris. http://www.gridpp.ac.uk/wiki/DPM-on-Solaris.

[7] European DataGrid Project (EDG). http://eu-datagrid.web.cern.ch/.
[8] Generic Information Provider (GIP). http://lfield.home.cern.ch/lfield/gip/documentation.html.
[9] gLite. http://glite.web.cern.ch/glite/.
[10] Global Grid User Support (GGUS). http://www.ggus.org/.
[11] Globus Toolkit v2 (GT2). http://globus.org/toolkit/docs/2.4/overview.html.
[12] GLUE Schema. http://glueschema.forge.cnaf.infn.it/.
[13] GridEngine. http://gridengine.sunsource.net/.
[14] GridPP. http://www.gridpp.ac.uk/.
[15] GSTAT. http://goc.grid.sinica.edu.tw/gstat/.
[16] IC-HEP GridLoad Monitor. https://gfe03.hep.ph.ic.ac.uk:4175/cgi-bin/load.
[17] Large Hadron Collider (LHC). http://lhc.web.cern.ch/.
[18] LCG on SGE. http://www.gridpp.ac.uk/wiki/LCG-on-SGE.
[19] Level-2 Grid. http://tyne.dl.ac.uk/ETF/public/Deployment/Level2/.
[20] LHC Compute Grid (LCG). http://lcg.web.cern.ch/LCG/.
[21] London e-Science Centre, Imperial College London, UK. http://www.lesc.ic.ac.uk/.
[22] OpenPBS. http://www.openpbs.org/.
[23] Resource Specification Language (RSL). http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html.
[24] R-GMA. http://www.r-gma.org/.
[25] RRDTool. http://oss.oetiker.ch/rrdtool/.
[26] Virtual Data Toolkit (VDT). http://vdt.cs.wisc.edu/.

Proceedings of the Second IEEE International
Conference on e-Science and Grid Computing (e-Science'06)
0-7695-2734-5/06 $20.00 © 2006 IEEE