HEP APPLICATIONS EXPERIENCE WITH THE EUROPEAN DATAGRID MIDDLEWARE AND TESTBED

S. Burke, Rutherford Appleton Laboratory, Oxford, UK
O. Maroney, Bristol University, UK
F. Brochu, Cambridge University, UK
I. Augustin, F. Carminati, J. Closier, CERN, Geneva, Switzerland
D. Boutigny, J.J. Blaising, CNRS LAPP, Annecy, France
V. Garonne, A. Tsaregorodtsev, CNRS Marseille, France
D. Colling, Imperial College, London, UK
P. Capiluppi, A. Fanfani, C. Grandi, INFN Bologna, Italy
R. Barbera, INFN Catania, Italy
E. Luppi, INFN Ferrara, Italy
G. Negri, L. Perini, S. Resconi, INFN Milano, Italy
M. Reale, A. De. Salvo, INFN Roma 1, Italy
S. Bagnasco, P. Cerello, INFN Torino, Italy
O. Smirnova, Lund University, Sweden
K. Bos, D. Groep, W. van. Leeuwen, NIKHEF, Amsterdam, Netherlands
F. Harris, I. Stokes-Rees, Oxford University, UK

Abstract

The European DataGrid (EDG) project ran from 2001 to 2004, with the aim of producing middleware which could form the basis of a production Grid, and of running a testbed to demonstrate the middleware. HEP experiments (initially the four LHC experiments and subsequently BaBar and D0) were involved from the start in specifying requirements, and subsequently in evaluating the performance of the middleware, both with generic tests and through increasingly complex data challenges. A lot of experience has therefore been gained which may be valuable to future Grid projects, in particular LCG and EGEE which are using a substantial amount of the middleware developed in EDG. We report our experiences with job submission, data management and mass storage, information and monitoring systems, Virtual Organisation management and Grid operations, and compare them with some typical Use Cases defined in the context of LCG. We also describe some of the main lessons learnt from the project, in particular in relation to configuration, fault-tolerance, interoperability and scalability, as well as the software development process itself, and point out some areas where further work is needed. We also make some comments on how these issues are being addressed in LCG and EGEE.

INTRODUCTION

The EU-funded European DataGrid (EDG) project [1] ran from January 2001 to the end of 2003, with an extension to March 2004 for the final EU review, which was passed successfully. HEP applications were represented through Work Package 8 (WP8) of the project, initially with representatives of the four LHC experiments and subsequently including the US experiments BaBar and D0. There were also five experiment-independent people. Over the lifetime of the project all participants performed increasingly complex tests to evaluate the EDG middleware. In 2004 the LHC experiments have also performed data challenges on the LHC Computing Grid (LCG) system [2], which uses a substantial amount of EDG middleware. This paper reports on the experiences during this period, and indicates some of the lessons learnt.

EDG MIDDLEWARE AND TESTBED

EDG developed middleware in 5 main areas: job submission, data management, information systems, fabric management and mass storage interfaces. There was also a joint development with the DataTAG [3] project to produce VOMS [4], a VO-based membership and authorisation service. Basic services were taken from the Globus [5] and Condor [6] projects.

In addition, EDG ran an application testbed which operated continuously from November 2001 until after the formal end of the project in April 2004. The size of the testbed increased over time; it started with five core sites, and by the time of the final EU review in 2004 had grown to 21 sites in 8 countries, with 161 CPUs and 14 Tb of disk storage, plus four sites with tape-based mass storage systems (MSS). Most of those sites have since joined the production LCG system.

The middleware evolved through a series of releases, partly driven by the need to solve problems discovered by application groups. By early 2003, version 1.4 of the

middleware was reasonably stable, and a report on the status at that time was presented at CHEP03 [7]. In August 2003 EDG released version 2.0 with major new functionality in many areas, and this was upgraded to version 2.1 in October 2003. The final EDG release was 2.1.9 in January 2004. WP8 submitted a report on the final system as an EU deliverable in January 2004 [8].

Middleware Evolution from 1.x to 2.x

This section gives a brief summary of the middleware changes between the two major EDG releases.

The job submission software changed fairly little from the viewpoint of a user, but internally was refactored substantially to deal with stability and scaling problems found in the first release. In addition some new features were added, for example to support interactive jobs, checkpointing and composite job definitions, but these were not fully integrated by the end of the project and hence were generally not tested by the applications.

The data management software was completely re-written. The first version used the LDAP-based Globus Replica Catalog, which was found to have serious scalability and performance problems, and a data management system called GDMP which was very difficult to configure and use. For cataloguing, the new system has separate file and metadata catalogues, based on MySQL or Oracle databases with a web-service front end. The system was designed to have the file catalogues distributed at each site with an indexing system to aggregate the information, but has so far only been deployed in a single-catalogue mode. The C++-based client-server GDMP system was replaced with client-only tools written in java.

For data storage, both MSS and disk-based, EDG developed Storage Element (SE) software implementing an early version of the web-service-based SRM standard [9], which was developed during the lifetime of the project by various partners including EDG. The so-called "classic SE", with a GridFTP server but no management software, was also developed as a fallback solution.

The information system changed completely, from the LDAP-based Globus MDS to a new system called R-GMA. This uses a relational data model with SQL-style querying, and an architecture where consumers of information find producers which can satisfy their queries by looking them up in a central registry, but with the information itself passed directly from producer to consumer. Adapters were also provided for compatibility to convert to and from the LDAP format.

The information schema also changed to the unified GLUE schema [10], aimed at helping to make different Grids interoperable. This was implemented in both relational and LDAP versions.

EDG continued to develop the LCFG-ng fabric management system, and this was used at most EDG sites. In addition it developed a new system called Quattor which was not used in EDG but is used to manage the LCG fabric at CERN and some other sites.

The VOMS VO management system was introduced in the EDG testbed as a prototype, but in general the old system based on an LDAP VO membership list and dynamic, anonymous accounts continued to be used.

HEPCAL Use Cases

In 2002 LCG produced a document giving a set of 43 use cases for basic use of Grids by HEP applications, known as HEPCAL [11]. This was based on previous work in WP8. In early 2003 the version 1 middleware was assessed against these use cases, and this was repeated for version 2. Use cases were put into one of four classes, as follows (the number of uses cases satisfied in version 2 is given, followed by the version 1 value in parentheses):

Fully implemented: 13 (6)
Largely implemented: 4 (12)
Partly implemented: 11 (9)
Not implemented: 15 (16)

It can be seen that most of the progress was in resolving problems with existing functionality, rather than in providing missing functions.

The missing functionality falls into three main areas. One concerns virtual data, i.e. storing a recipe for files which allows them to be materialised on demand, which was not part of the EDG workplan. The second relates to metadata, a concept which is still unclear on both the middleware and application sides. Finally, various use cases relate to things like authorisation, job control and optimisation, which were partly delivered but could not be integrated or tested in the time available.

LCG and EGEE

LCG has taken the job submission, data management and fabric management software from EDG, and continued to evolve them to improve performance and stability. R-GMA and the EDG SE were not sufficiently stable for production use, although R-GMA has recently been deployed in LCG for monitoring. As an information system LCG uses customised LDAP servers. Storage Elements currently use the classic SE interface until a production-quality SRM solution is available. VOMS is still under development.

Testing in EDG was fairly limited as time was short and the software was initially quite unstable. However, D0 ATLAS and LHCb performed some tests, in addition to generic testing by the experiment independent people. The LHC experiments have run major data challenges on LCG in 2004, and comments in this paper are based on this experience as well as that on the EDG testbed.

The EU-funded EGEE project [12] is a successor to EDG. Among other things it will take over the operation of the LCG system and expand it to non-HEP sites and applications, and will also produce new middleware.

EXPERIENCE AND LESSONS LEARNT

HEP applications have made enormous progress in the use of Grids, and are now routinely using LCG to run

hundreds of thousands of production jobs. We have generally had a good relationship with middleware developers and system managers, and problems have been progressively addressed as the middleware has evolved. Nevertheless, current systems are far from perfect, and some broad lessons have been learnt about which areas are especially prone to problems.

General Issues

It has become clear that running testbeds on a reasonably large scale with real users is essential to developing a robust system. Many problems emerge which are not seen in the closed environments in which developers test their code, and users often do things not expected by the developers. Also, Grids are likely to be highly heterogeneous, and middleware needs to be sufficiently flexible to cope with a wide range of different systems. Testing scalability is also important, in many cases failures were only seen once the size of the system grew beyond a few sites.

Developers should not be overambitious; many features in the EDG middleware were not fully integrated by the end of the project. Software integration and configuration can be a major time-consumer, often taking longer than the initial code development. Middleware is often flexible and can be configured in many ways, but finding a working configuration is then correspondingly difficult. This can be made harder with the traditional style of configuration via large numbers of parameters in text files, which are often obscure in their effects. It is also harder if the middleware does not give clear error messages if the configuration is not suitable.

Following on from this, middleware needs to be adaptive and fault-tolerant. In a large Grid there will always be misconfigured sites, failed machines, full disks and services down, so middleware needs to regard exceptions as a part of normal operation. If errors do occur, tracing problems through several layers of middleware distributed over many sites requires consistent error logging and remote diagnostic tools.

From the start of EDG one goal was to avoid single points of failure. In practice, services were deployed with a single instance with the intention of evolving to a distributed and/or replicated model, but in practice this has so far failed to happen. The lesson here is that distributing services is not straightforward and requires a dedicated effort to achieve.

A similar situation has been seen with security. Since secure services tend to subtract from usability rather than adding to it they are not seen as a priority, and in most cases we started with insecure services with the intention of adding security later. However, this has yet to happen; security is a difficult and specialised area and again requires a lot of effort. In general HEP does not regard security as a high priority, but HEP systems are still vulnerable to attack and this is likely to become a more critical issue as the Grid becomes larger and more widely known.

EDG developed many tools for particular functions, as described above. However, some things were not covered by any work package, and hence have left gaps in the LCG system. An overall architecture, informed by user requirements and use cases, is needed to ensure that all areas are covered and that different services can interact to achieve the desired behaviour.

Deployment

Middleware needs to be easily deployable on a wide variety of sites by system managers with limited time and who may not be Grid experts. Configuration managers like LCFG and Quattor can help, but cannot be used everywhere, and manual configuration has so far been very difficult and error-prone. It is also hard to validate a site as correctly configured, and a working site often ceases to work properly after some time. Problems which may affect only a few batch worker nodes in a large system are particularly difficult to identify, and can result in the "black hole" syndrome where one node fails jobs immediately and hence attracts further jobs. In the current LCG system, site-specific problems are the major cause of job failures.

Interoperability between Grids is desirable, and the GLUE schema is a step towards it. However, going beyond simple job submission is difficult, particularly for anything which requires client software pre-installed on batch worker nodes, or needs specific versions of compilers or libraries. Dynamic installation of a software environment may be needed in future if non-dedicated resources are to be used.

Another issue which remains unresolved is access to the WAN from worker nodes. Many sites would like to deny such access, but currently both the middleware and application software require it, and there has so far been little movement on either side.

Job Submission

The Resource Broker (RB) in EDG 2 is much more stable than the earlier version, and has been further improved by LCG. Failures due to the RB itself are now around the 1% level. However, job submission remains a fairly slow process, taking several seconds per job, and since jobs can only be submitted one at a time it can be difficult to submit a large number of short jobs.

There are also still some problems with scheduling, as the RB relies on items in the information system which are not always calculated in a meaningful way. In general this no longer results in major problems, but can lead to a non-optimal distribution of jobs.

The current implementation of the RB uses a "push" model where jobs are dispatched to local batch queues as soon as they are submitted. For some purposes a "pull" model where sites collect jobs from a central queue may be more appropriate. EGEE is currently developing such a solution.

Error reports remain hard to interpret, and the absence of a simple link between the job ID in the RB and the

local job ID in a batch system also makes it difficult to trace the reasons for failures.

R-GMA

R-GMA appears to be a promising technology, for application monitoring in particular, as it uses a standard relational data model with SQL-style queries and makes it easy for users to define their own tables.

However, the time available for testing R-GMA in EDG was fairly limited. Some tests of application monitoring were performed by the CMS and D0 experiments, and were generally successful. R-GMA has only recently been deployed in LCG, so further testing is required to make a full assessment.

Information systems

In the EDG 2 testbed R-GMA was used as the information system. This appeared to work satisfactorily, but could only be tested on a fairly small and lightly-loaded system over a limited time, so strong conclusions cannot be drawn.

LCG has developed a system of LDAP servers backed by databases, the so-called BDII. This has had some problems with performance under heavy load, but in general has been satisfactory.

Some problems have been encountered with the GLUE schema, e.g. that it does not reflect the scheduling policies at many sites. GLUE is not directly under the control of LCG and has so far proved difficult to evolve, but an effort in this direction is currently underway. The lesson here is that schema evolution is difficult, so it is important to make the schema flexible enough to cope with a wide range of situations. There is also a need to have a clear definition of things like measurement units, e.g. some sites have published time limits in real time and others in units normalised to CPUs of some standard power.

Errors in the published information can lead to incorrect scheduling decisions, so there is a need to have some checking in the information providers. Default values need to be defined in a "safe" way, such that they will tend to result in too few jobs going to a site rather than too many.

Data Management

The EDG replica management tools are fairly intuitive and have worked well. However, the choice of java for clients resulted in the commands being very slow, typically taking several seconds, and LCG has since re-implemented them in C++.

The tools also have somewhat limited functionality. They operate on only one file at a time, and as client-only tools there is no provision for queuing transfers or retrying at a later time after a failure. There are no transactions or consistency checks, so failures can leave the system in an inconsistent state. In general there is a need for a higher-level data management system with a client-server architecture.

The replica and metadata catalogues have been tested up to a few million entries in LCG, without significant problems. However, the web service interface can be very slow to return large amounts of data from queries due to the XML encoding overhead.

The SRM protocol is generally considered to be the way forward for management of Grid-enabled storage, but production-quality implementations are rare so far. EDG provided an SE with a partial SRM implementation, but while this worked reasonably well it had many problems with configuration and stability, and was essentially a prototype.

VO Management

The LDAP-based VO membership system used in EDG has worked well, but has very limited functionality, for example there are no subgroups and a user cannot belong to more than one VO. VOMS appears to be a promising solution, but has not yet been tested in a production environment. Security tends to come last, but such tools will be needed urgently as Grid usage increases. VOs themselves will also need to develop experience in how they want to manage the system.

SUMMARY

Over the last three years a great deal of experience has been gained in the use of Grids by HEP experiments. The EDG middleware has evolved substantially, and much of it is deployed in the LCG production system, which is being used extensively for real work. However, many problems remain, and a lot of work is still required. EGEE is working towards this goal, but it needs to take account of the experience gained with existing systems. For the LHC experiments in particular, 2007 is no longer very far away, and a fully-working Grid is needed soon.

ACKNOWLEDGEMENTS

The authors wish to thank the EU and our national funding agencies for their support. We would also like to acknowledge the active cooperation of our EDG colleagues in the middleware and testbed work packages, as well as the substantial support of the Project Office.

REFERENCES

[1] http://eu-datagrid.web.cern.ch/eu-datagrid/
[2] http://lcg.web.cern.ch/LCG/
[3] http://datatag.web.cern.ch/datatag/
[4] http://grid-auth.infn.it/
[5] http://www.globus.org/
[6] http://www.cs.wisc.edu/condor/
[7] http://arxiv.org/pdf/cs.DC/0306027
[8] https://edms.cern.ch/file/428171/3/DataGrid-08-D8.4-0127-3-0-161203.pdf
[9] http://sdm.lbl.gov/srm-wg/
[10] http://www.cnaf.infn.it/~sergio/datatag/glue/
[11] https://edms.cern.ch/document/375586/1.3
[12] http://egee-intranet.web.cern.ch/