Journal of Grid Computing (2004) 2: 369–386

HEP Applications and Their Experience with the Use of DataGrid Middleware

S. Burke[1], F. Harris[2], I. Stokes-Rees[2], I. Augustin[3], F. Carminati[3], J. Closier[3], E. van Herwijnen[3], A. Sciaba[3], D. Boutigny[4], J.J. Blaising[4], V. Garonne[5], A. Tsaregorodtsev[5], P. Capiluppi[6], A. Fanfani[6], C. Grandi[6], R. Barbera[7], E. Luppi[8], G. Negri[9], L. Perini[9], S. Resconi[9], M. Reale[10], A. De Salvo[10], S. Bagnasco[11], P. Cerello[11], K. Bos[12], D. Groep[12], W. van Leeuwen[12], J. Templon[12], O. Smirnova[13], O. Maroney[14], F. Brochu[15] and D. Colling[16]
[1]*Rutherford Appleton Laboratory, UK*
[2]*Oxford University, UK*
[3]*CERN, Switzerland*
[4]*LAPP/Annecy, CNRS, France*
[5]*CNRS, Marseille, France*
[6]*INFN, Bologna, Italy*
[7]*INFN, Catania, Italy*
[8]*INFN, Ferrara, Italy*
[9]*INFN, Milan, Italy*
[10]*INFN, Romal, Italy*
[11]*INFN, Turin, Italy*
[12]*NIKHEF, Netherlands*
[13]*Lund University, Sweden*
[14]*Bristol University, UK*
[15]*Cambridge University, UK*
[16]*Imperial College, UK*

Key words: applications, EDG, EGEE, Grid, HEP, LCG, LHC, middleware

Abstract

An overview is presented of the characteristics of HEP computing and its mapping to the Grid paradigm. This is followed by a synopsis of the main experiences and lessons learned by HEP experiments in their use of DataGrid middleware using both the EDG application testbed and the LCG production service. Particular reference is made to experiment 'data challenges', and a forward look is given to necessary developments in the framework of the EGEE project.

1. Introduction

This paper overviews the evaluations of European DataGrid (EDG) middleware performed by the High Energy Physics (HEP) Applications Workpackage (Work Package 8 – WP8) throughout the lifetime of the DataGrid project from January 2001 to the end of March 2004. It also comments on the use of this middleware by the LHC Computing Grid (LCG) project up to September 2004 (the LHC is the Large Hadron Collider particle accelerator).

WP8 was composed of representatives from HEP experiments, starting with the 4 LHC experiments (ALICE, ATLAS, CMS, LHCb) and later augmented by effort from the US experiments BaBar and D0. In addition there were experiment independent experts, the so-called "loose cannons", who performed

generic application-oriented evaluations in addition to assisting experiments with their evaluations.

During 2001 WP8 worked on writing detailed requirements documents which later evolved into the HEPCAL [1] use case documents used by the HEP community as reference documents for middleware developments. Evaluations of the first middleware started in late 2001, and by July 2002 the project decided to form experiment/middleware task-forces to bring the middleware towards production quality. This commenced with ATLAS and in November 2002 evolved to intensive work with the CMS experiment. This pioneering work enabled the EDG middleware to be used for real physics production by CMS, and subsequently by the other LHC experiments later in 2003. The middleware continued to evolve, with new versions also being evaluated by LCG starting in summer 2003, and were used for real production by LCG starting in early 2004. At the time of writing this paper EDG middleware is being used in over 70 LCG sites distributed world-wide.

In Section 2 of this paper we give an overview of the nature of HEP computing, and introduce the HEPCAL use cases. In Section 3 we provide a qualitative appreciation of the various system components and relate these evaluations to the HEPCAL use cases. Section 4 describes the various quantitative evaluations performed throughout the history of the project up to current evaluations within the umbrella of LCG. Here we see that in addition to middleware issues there are major sources of problems for production systems in the areas of individual site configuration and validation and application software distribution.

Section 5 concludes the paper with a summary of the major lessons learned of relevance to the provision of large scale Grid systems.

2. High Energy Physics (HEP) Computing

Computing has always been important for nuclear and particle physics as many physics processes can only be studied statistically using large samples of data. The experiments which are currently being prepared for the LHC (Large Hadron Collider) at the European Laboratory for Particle Physics CERN in Geneva, will be looking for reactions which in some cases are expected to be detected only a few times per year. The samples of data to be processed off-line to find these events will total thousands of Terabytes. The associated computing is both CPU and data intensive, involving highly tuned "data mining".

The European DataGrid (EDG) project came at a time of the LHC Computing Grid (LCG) project preparation when Monte Carlo studies of the physics processes and the detector responses to those processes were very important. In a Monte Carlo simulation several steps can be recognised:
(1) event simulation (generation),
(2) detector simulation,
(3) detector response simulation (digitisation),
(4) background simulation.

In (1), event simulation, knowledge of physics processes has been encapsulated in code, and running such code produces lists of the particles produced when two protons collide, together with their properties.

In (2), the detector simulation step, all these particles are tracked through the whole volume of the detector. This is by far the most compute intensive step of the calculation, involving tracking thousands of particles with tiny steps through very large detectors. A particle makes a small step in 3-dimensional space and for each step a possible change of direction is calculated if a magnetic field is present and if the particle is charged. Moreover, as a function of the material in which the particle progresses, multiple scattering and energy loss are calculated, and at each step the probability is calculated that the particle may decay into other particles, or annihilate, or be absorbed, or interact with the material to produce more particles.

In (3), the last step of the simulation, the detector response to the traversal of all the particles is determined. This means that a detailed layout of all active elements, their position and their read-out electronics has to be known. The output from this step in the program sequence must be data that looks just like the real data which will be collected with beams colliding at the centre of the detector. Such data can then subsequently be used to test the data reconstruction programs as if it were real data from the detector. This is important for the development of reconstruction algorithms and to determine the reconstruction efficiency.

In (4), to make the data for testing the reconstruction more realistic, extra data has to be added to each event. This extra data comes from different classes of *background* to the *signal* event of interest. This background arises from, for example, other possible events in the same proton–proton bunch collision, or from noise in the detectors.

At all steps of this program sequence data has to be fed to the running code, both through data files

and experiment specific databases accessible to the running program. Event data is generally processed sequentially, with each processed event being written out before the next event is read. This general scenario is implemented in different ways in each experiment which has its own data model.

The raw physics data, itself totalling several Petabytes, will be generated and stored at CERN. The processing of this data will be distributed world-wide. The models for this processing and associated data distribution are still under evaluation, but certainly these models include replication of large samples of reconstructed data throughout the collaborations. Typically the raw data will be reprocessed a few times over the life of the experiment as reconstruction algorithms are improved.

The amount of data coming from the reconstruction is less than the raw unprocessed data, but is still very significant in size. However, while the unprocessed data is likely to be needed only a few times, the processed data, which is used for physics analysis, is used over and over again by many physicists and at many places. Moreover, whereas the reconstruction is a relatively controlled process which will be operated by a limited number of specialised people, analysis operates around the clock on a world-wide basis according to the work patterns and pressures of individual scientists and engineers. Hence there is a need for the data used by these analyses to be even more distributed than the raw, unprocessed data.

Databases containing experiment-specific data will probably have to be replicated at various places for efficiency reasons as they are accessed frequently: for reconstruction, for analysis and for Monte Carlo simulations. Plans for replicated databases exist but were not tested within the EDG project.

For many years the HEP community has been running large distributed computing infrastructures, but generally each site is configured in substantially different ways. The Grid paradigm [2] offers an opportunity to have a homogeneous view of a world-wide virtual computing system, and potentially access to world-wide resources, some of which may not be owned by the community.

The HEP computing user expects to see a distributed computing infrastructure, including hardware resources and the corresponding software tools and services, which allow optimal execution of computational tasks, with appropriate access to the distributed data. The Grid is assumed to provide proper authentication and authorisation, transparent access to resources, and management of the necessary databases.

2.1. *HEP Computing and the Grid – the Model and the Use Cases*

The 4 LHC experiments (ALICE, ATLAS, CMS and LHCb) have all developed detailed planning for computing leading up to the LHC startup in 2007. This includes large scale *data challenges* taking place in 2004. As an example, Table 1 summarises the characteristics of the data challenge performed in spring 2004 by CMS. Note that CPU times are normalised to CPUs with a power equivalent to a Specint 2000 rating of 1000. Real times assume that the simulation and digitisation phases use machines with an average power of 500 SI2000, and 700 SI2000 for reconstruction, which is a rough average for the resources expected to be available.

The data challenge is logically divided into three data processing phases (*Generation & Simulation, Digitisation* and *Reconstruction*), followed by a data distribution phase. To reach a reconstruction event rate which corresponds to 25% of the LHC startup rate in 2007, 50 million events must be processed in one month. This work is executed in an organised production mode. There will be future evaluations of the performance of the Grid in supporting *chaotic* use by thousands of users.

For convenience the data are split into several streams, so the objects for a single event are spread across several files (e.g., 16 for the reconstruction output). The number of events per job is adjusted to keep file sizes below 2 Gb.

The number of CPUs dedicated to the different steps of the challenge has been calculated assuming 75% efficiency during the allocated time, i.e. it is assumed that 25% of jobs will fail in some way and need to be re-run. This is just an ad-hoc guess, the real efficiencies can vary substantially depending on circumstances (see Section 4 for further discussion).

From the characteristics of the programs, the interaction frequencies with the Grid Workload Management System (WMS, see Section 3.7) and the Replica Manager (see Section 3.6) can be extracted. For example, for the reconstruction phase about 2700 jobs per day are submitted to the WMS, 0.4 files per second registered with the RM (0.5 interactions per second are needed if considering also the lookup frequency), and 4 MB/s are transferred to each Regional Centre.

It should be noted that this challenge used components both from Grid projects and from CMS, the latter including a system for scheduling file transfers.

Table 1. Parameters of the CMS data challenge 2004.

		Gen./Sim.	Dig.	Reco.
Time allocated	months	5	2	1
CPU time per event	sec	160	24	12
Input event size	kb	40	900	700
Output event size	kb	900	700	150
Number of jobs		200,000	50,000	50,000
Events per job		250	1,000	1,000
CPU time per job	sec	40,000	24,000	12,000
Output size per job	Mb	225	700	150
Duration of job	hours	20.2	12.1	4.8
No. of input files/job		1	4	3
No. of output files/job		1	3	16
Total CPU time	months	4115	617	309
Total no. of CPUs		1496	561	440
Total output data size	Tb	45	35	7.5
Total no. of files		200,000	150,000	3,200,000
Event rate (at 100% eff.)	Hz	5.1	12.9	25.7
Job submission rate	jobs/day	1777	1110	2217
Output rate	MB/sec	4.6	9.0	3.9
File registration rate	Hz	0.02	0.04	0.4

2.2. *Users and Virtual Organisations (VOs)*

A community is organised within a VO which has appropriate structures and authorisations according to the role of the user. Typically an HEP VO is organised in groups and sub-groups such as "experiment production", "physics group production", "group user analysis" and so on.

A user is any individual associated with a VO using the Grid services in the process of performing computational work. A user typically submits jobs to the Grid, i.e. requests of work to be done or actions to be taken on his behalf. Sometimes this kind of user is called an "end user" to indicate that she is not part of the service-providing infrastructure, but rather at the "end" of the service providing chain. Typically, in the scope of HEP users are physicists, engineers and computer scientists working for the ultimate goal of extracting the physics information from the collected data. Other kinds of "actors" accessing Grid resources are described in the following section. We have used the term "actor" in a set of use case analyses developed for HEP computing [1].

To identify a set of common use cases, it was necessary to start with the definition of the actors that are involved in the computing activity of an LHC experiment. The same physical person may play more than one role depending on her activity at a given moment. Examples of actors are physics users, production managers, experiment managers, software developers and software librarians.

The HEPCAL (HEP Common Application Layer) document [1, 3] was published in May 2002. It gives 43 use cases related to the expected use of Grid middleware in an HEP context. This was a natural development of the ongoing requirements work which was accomplished at the start of the EDG project [4]. These use cases were not intended to cover every aspect of HEP use of the Grid, but specify the basic functionality required.

HEPCAL distinguishes two logical entities containing data: catalogues and datasets. A catalogue is a collection of data that is updateable and transactional. A dataset is a read-only collection of data. Datasets or catalogues might be implemented as one or more files; however they might be implemented otherwise, such as in Objectivity or Oracle databases. Catalogues will often contain so-called metadata, i.e. information about other data stored in datasets. Potentially there are various kinds of catalogue; some contain

Grid-specific metadata such as information about the location of replicas, some contain application-related metadata, and others contain general application-specific data like detector conditions for a particular time period.

The use cases provide important benchmarks to measure the current state of the technology. An evaluation of EDG middleware with respect to these use cases is given in Section 3.1.

3. Qualitative Application Evaluations of Grid Software Components in EDG

Since late 2001 a series of evaluations have been made of EDG middleware as deployed on the EDG testbed. These fall into 2 classes, firstly generic evaluations of the basic components, and secondly end- to-end evaluations conducted by the HEP experiments interfacing the middleware to their production systems. In addition, towards the end of 2003 the LCG project began to deploy a Grid infrastructure including many components from EDG, which the experiments are now using to perform large-scale data challenges.

In general, the fact that the middleware has been in rapid development, and in EDG was deployed on a testbed which was not intended to have production-quality levels of support, has made it difficult to make detailed numerical evaluations of performance. Also the definition of metrics for assessment of Grid performance is not straightforward and is still under active discussion. Nevertheless it has been possible to draw many qualitative conclusions about the current state of the middleware and the desired directions for future development.

Below we describe the level of satisfaction of the HEPCAL use cases, and then overview the results of the middleware evaluations component by component. In Section 4 we summarise particular important results obtained by the experiments. Throughout the EDG project HEP applications produced 3 major evaluation reports [5–7] for the EU, and an overview for CHEP03 (the 2003 Computing in High Energy Physics conference) [8].

3.1. *The Current Status with Respect to HEPCAL Use Cases*

In the following sub-sections we summarise the satisfaction, or otherwise, by the EDG middleware of the 43 HEPCAL use cases. For this purpose we have broken down the use cases into various classes. The numbers in brackets give the number of use cases which are reasonably well-satisfied, and the total number.

3.1.1. *Basic (15/19)*

These relate to fundamental Grid operations like submitting and controlling jobs, registering and replicating files, and querying the state of the system. Of these, 15 are implemented by the EDG middleware, although in some cases there are minor areas where the implementation is not ideal, in particular concerning the detection and treatment of errors and support for file metadata.

Three of the missing cases concern the job submission system, specifying ways to control jobs and query their state. At present the EDG software allows only very limited control, and queries relate only to high-level state changes. The final unimplemented case is a technical issue involving registration of files with a known GUID, and this has subsequently been fixed by LCG.

3.1.2. *Security (3/5)*

Two use cases concern the joining and leaving of a Virtual Organisation (VO). These are implemented in EDG using an LDAP server to hold VO membership lists, but this has fairly limited functionality. EDG has developed VOMS (the Virtual Organisation Management System) in collaboration with the DataTAG project, which should allow much more flexible control of VO-based authorisation. A third case specifies single sign-on, which is satisfied by the standard Globus proxy creation and will be enhanced with the extended proxies used by VOMS.

The two final security use cases concern the advance reservation of resources and the allocation of resources between VO members, and these are not addressed in the current system.

3.1.3. *Metadata (0/2)*

Two use cases specifically involve the modification of file-related metadata, and performing queries to select files based on the metadata. The EDG Replica Metadata Catalogue offers a prototype with partial support for these use cases, but more work is needed by both application and middleware developers in this area.

3.1.4. *Virtual Data (0/2)*

The virtual data concept implies that a recipe to generate a file is stored in a catalogue, and files can therefore

be materialised on demand if a physical copy of the file does not already exist, or if that is more efficient than replicating the file from elsewhere. This was out of the scope of EDG, and is likely to require substantial further work to implement.

3.1.5. *Optimisation (3/4)*
One use case concerns the evaluation of cost functions for data access to allow the most efficient access method to be chosen. The EDG middleware has a substantial amount of support for this concept, but testing has been limited because the relevant network monitoring data can only be gathered in something close to a real production system.

Two other optimisation use cases refer to job submission. One concerns the specification of hints, e.g., for cpu time consumption, memory usage or disk space needed, to allow jobs to be scheduled efficiently. This is supported to the extent that jobs can apply their own constraints and ranking criteria based on information stored in the information system, but any optimisation is provided by the user rather than the WMS. More research on job distribution algorithms is desirable in the light of experience with real production Grids.

Another use case concerns the automatic splitting of jobs into subjobs. HEP jobs generally involve the sequential processing of large numbers of files, and hence are good candidates for splitting to balance the load. Functionality to address this was available right at the end of the project and was demonstrated by CMS at the final EU review.

A final use case relates to the possibility of using remote access to a small part of a file to avoid the overhead of complete replication. This is not supported.

3.1.6. *Application Catalogues (0/4)*
Four use cases concern the concept of application catalogues stored within the Grid. EDG has provided a GSI-enabled interface to underlying databases which is used, for example, for the Replica and Metadata Catalogues, but there is no explicit support for application databases. At present the model for files is that they are write-once and subsequently read-only, whereas catalogues must be updatable, which implies strong consistency requirements if catalogues are to be replicated. R-GMA provides a different model for a distributed database which may be suitable for some of the use cases, but this has not yet been investigated.

3.1.7. *Application Interfaces (0/7)*
The final set of use cases are at a higher level, and relate to interactions between middleware and application software. These can generally be achieved by implementing the functionality at the application level, but have no specific support in the middleware.

Two relate to the submission and control of large sets of jobs treated as a single production, and a third relates to storing user-defined metadata about jobs in the WMS job catalogue. Three more relate to specialised kinds of jobs.

Finally, there is the question of the installation and publication of application software. Within EDG this has been achieved by treating application software in the same way as the middleware and incorporating it in the EDG releases, but this is not suitable as a long-term solution. LCG has since developed a solution which allows software managers from each VO to install and manage software on NFS-mounted areas.

3.2. *Security Issues for HEP*

In general HEP does not have strong security requirements, in that data are not usually confidential or sensitive and the community of users is relatively trustworthy. However, as with any computer systems there is a need to protect against hackers and other forms of malicious attack, and sites require robust audit trails to allow the tracing of actions by individual users. Accounting systems are also needed to allow monitoring and enforcement of resource sharing between and within Virtual Organisations. We touch below on two of the areas which particularly affect the operation of an HEP experiment.

3.2.1. *Authorisation*
HEP Virtual Organisations are generally very large (hundreds or thousands of people) and long-lived (decades), with a well-developed internal structure. This implies the need for a relatively heavyweight authorisation management system which emphasises functionality and scalability at the cost of a higher administrative burden.

Authorisation is likely to be needed at a variety of granularities, from overall permission to use a CE down to access control on individual files, or to particular fields in the information system. There is also likely to be a need for resource reservation, e.g., to ensure sufficient resources for official production jobs which may need to take precedence over individual users.

The VOMS system securely manages users within a VO, organising them into subgroups and allowing the specification of arbitrary roles and capabilities. This information is embedded as an extension to a Globus proxy [9], and can be parsed by VOMS-aware middleware to enforce authorisation decisions.

VOMS seems to be a promising way to solve the authorisation needs of HEP experiments. It was tested to a limited extent towards the end of the project and appeared to work as expected.

3.2.2. *Service Proxies*

Many sites wish to deny direct access to the wide-area Internet from batch worker nodes. This is partly due to the fact that IP addresses are a limited resource, and partly because a large Grid could otherwise be an ideal platform from which to launch a Distributed Denial of Service (DDOS) attack. Sites also want to limit the number of holes in their firewalls as much as possible.

However, at present both middleware and application client software require outbound external access. This requirement could be removed by running service proxies on gateway nodes. This could also be useful in other ways, for example allowing retries of failing operations without blocking the client. So far there has been little development in this area, indeed between release 1 and release 2 of the EDG middleware the replica management system moved away from a client-server model to a purely client-based system.

3.3. *Storage Elements*

In principle a Storage Element (SE) is a managed interface to mass storage, but the required functionality has only emerged gradually during the EDG project. The need has developed for the SE concept to be extended to provide an interface to a wide range of tape and disk based systems, in a way which is easy to deploy and manage with limited manpower and expertise.

In addition there is a need to have some management of disk space local to jobs running on batch worker nodes, to provide general scratch space, for local copies of permanent files and to store application software needed by jobs in execution. So far the SE concept has not been extended to deal with these requirements, and management of local storage is left to applications, but this appears to be an area which is in the middleware domain and deserves further attention.

In the early stages of the project a minimal SE was deployed which essentially consisted of a Globus [10] GridFTP server running over a simple disk-based file system, or more recently with an interface to a tape-based system. This so-called "classic SE" provides the basic functionality to read and write files, and simple access control via the Grid map file, but lacks any management features.

EDG has participated in the definition of the Storage Resource Manager (SRM) standard [11], which is intended to provide a standard Grid-aware web service-based interface to any Mass Storage System (MSS). However, SRM implementations are not fully mature and a full assessment will have to wait for a stable implementation. The initial experience is encouraging, but many desired features are not yet available, e.g., reservation of disk space, pinning and automatic deletion of cached copies of files, load balancing across disk servers, fine-grained access control, and optimisation hints.

3.4. *Replica Catalogues*

For the first major release of the EDG software the Globus Replica Catalog, based on LDAP (the Lightweight Directory Access Protocol), was used, but this proved to be inadequate both in the small number of file names which could be stored (a few thousand) and poor performance under load.

The second release uses a web-service front end to a standard database; deployed systems so far use either Oracle or MySQL. This has proved to be much more robust. The system has been tested up to $\mathcal{O}(100{,}000)$ entries with no sign of degradation, and there have been no outages due to the catalogues themselves. The use of a Globally Unique Identifier (GUID) as a file identifier appears to be a good design choice to solve problems with name clashes in a distributed system.

However, as currently deployed we have only a single catalogue instance for the whole Grid, which clearly represents a single point of failure and a potential limit to scalability. Even in the relatively small systems tested so far some scalability problems can be seen, for example running 600 file registrations in parallel shows an 0.5% failure rate due to catalogue communication errors, which also highlights the need for retries of operations which fail for transient reasons.

There are also performance issues associated with the use of a web service interface with Java client tools. Starting the Java Virtual Machine adds an overhead of several seconds which is unsuitable for single operations with a command-line tool. In addition, the way data are transported means that queries

which return large amounts of data are extremely time-consuming, typically taking 10 minutes to return 15,000 file names. Above about 50,000 matches to the query the client exceeds the configured memory limit.

A general security mechanism for web services, allowing both authentication and authorisation, is vital for use in a production system. This is not yet available and is a strong requirement for future development.

The topic of constructing scalable replica location services is an ongoing area of research [12].

3.5. *Metadata Catalogues*

Metadata is an area which still requires research at both the application and middleware levels. Applications typically have large amounts of metadata relating, for example, to the physics properties of events stored in files. Such metadata is highly application-specific, but it is essential that standard interfaces are available to allow application metadata catalogues to be integrated with Grid systems.

Metadata catalogues are also needed at the middleware level, for example to store file sizes, timestamps and security information. The EDG middleware also stores a human-readable Logical File Name (LFN) in the metadata catalogue. The current system is essentially a prototype, from which it should be possible to determine the requirements for future development. It uses a single database instance, currently shared with the Replica Catalogue, but there is no existing model for extending this to a distributed system. Apart from the LFNs the use of metadata is rudimentary, limited to checking that the file size is correct after replication. Treating LFNs as metadata rather than binding them tightly to GUIDs may not prove to be the best design choice, but the use of file names at the application level is not yet well-defined. The current security model envisages storing security information (Access Control Lists) in proximity to the files rather than in a metadata catalogue, but the optimum strategy is still a matter of discussion.

3.6. *Replica Management*

A replica management system needs to bind together access to file and metadata catalogues, Storage Elements and optimisation services and provide a high-level managed replication and file access service [13]. The first release of the EDG middleware used software called GDMP (Grid Data Mirroring Protocol), which had been adapted from a non-Grid-enabled file mirroring system. This had a C++-based client-server architecture with the concept of subscriptions for groups of files between pairs of Storage Elements. In practice this was found to be very difficult to configure and use correctly in a Grid environment.

The second release has a completely new replica manager, which is purely client code and is written in Java. The design has (at the request of application groups) emphasised usability and reliability over functionality. Functions provided include registration, replication and deletion of single files, which can be identified by GUID or LFN. Individual replicas can also be deleted. There is an interface to an optimisation service, allowing the selection of the most efficient replica relative to a given site, and to the SE for file listing and to obtain Transfer URLs for supported access protocols. However, there is currently no high-level interface to the catalogue query and metadata operations.

In terms of the implemented functionality the only significant problem is that the operations are very slow, e.g., registering a small file can take more than 10 seconds. This is related to the switch from C++ to Java and web service interfaces for the catalogues and SEs. We hope that this can be improved with further development.

Security is not implemented in the current system, but it has been designed with a secure web service model in mind. The current security model ties the services, i.e. file catalogues and name spaces on SEs, tightly to the Virtual Organisations, in that each VO has only one catalogue and different VOs are unable to read each other's catalogues or files. This is more restrictive than the model developed by the security group, but for HEP VOs which tend to be large and static it may be adequate.

The initial priority of application groups was robustness with basic functionality. However, applications will need a replica management interface at a higher level. Some important elements to be developed are:

(a) In the present model all replicas are equivalent, there is no way to designate a master copy which should not be deleted, or temporary copies which can be deleted freely by the system.

(b) There are no bulk transfer operations, all commands operate on single files with the exception of a batch registration command. Similarly there is no middleware support for file collections to be treated as a unit.

(c) The system does not have a robust transaction model. Failures during compound operations can leave the system in a variety of inconsistent states,

and error reports often make it difficult to determine the reason for a failure. There is usually no attempt to retry a failed operation, although this would in any case be difficult without a client-server model.

3.7. *Job Handling*

3.7.1. *Requirements*

Certain specifics of HEP computing had to be taken into account while developing workload management services.

(1) Resources consist of Linux clusters which are inhomogeneous both in terms of hardware and configuration:
   - mostly commodity hardware, no mainframes or parallel systems, hence no dedicated local scheduling systems;[1]
   - a single OS (originally RedHat 6.2, later upgraded to 7.3) has been chosen for the testbed;
   - OpenPBS has been chosen as the principle supported batch system for clusters, although LSF is also supported.

(2) User tasks are predominantly non-interactive, serial jobs:
   - there is a strong interest in interactive tasks, but this was not the top priority in the first instance;
   - certain user groups need to process parallel jobs as well, but within one cluster only, hence no inter-cluster communication is required.

(3) Jobs typically process large amounts of data and involve movement of big (up to several Gigabytes) files.

The main user requirements with respect to job handling can be outlined as:

(1) the Grid must present itself to the user as a "global batch system", providing seamless scheduling over distributed resources;

(2) in the absence of specific requirements, the job should be scheduled to the best available resource, i.e., the one providing the fastest turnaround;

(3) if the job requires access to a large input data set, it is usually desirable to move the job to the data, rather then other way around.

There are more detailed requirements, e.g., concerning job monitoring (see the HEPCAL [1] document for the extended description). However, for the first test runs by the LHC experiments, the fulfillment of at least those mentioned above was essential.

[1] A couple of sites actually had dedicated batch systems, and certain attempts to make use of them "as is" were made.

3.7.2. *Performance*

During the test runs on the EDG testbeds, the Workload Management System (WMS) was put under heavy load. Typically, it performed up to expectations, allowing users to submit thousands of simultaneous jobs and to use the system in production runs with high efficiencies. However, various problems were encountered by users.

- A centralized Resource Broker (RB) represents a single point of failure. Even though another RB instance can be deployed, jobs submitted via one RB can not be retrieved via another. This resulted in many jobs being inaccessible due to the unavailability of the original RB.
- A centralized Logging/Bookkeeping (LB) service is another single point of failure. Jobs cannot be managed while the LB is down, and lack of a distributed design for such a service led to many jobs being suspended or lost altogether.
- The RB allows the transfer of input and output files with the job, in addition to the use of Grid-registered files stored on SEs. However, the transfer of these files for all users through a single RB is not scalable, especially in the absence of disk management tools. Many RB failures can be attributed simply to full disks.
- The fact that data management is largely decoupled from job handling forced users to prepare wrapper scripts performing data movement prior to and after the main task execution. While moving jobs to data is often profitable, it is not unusual that jobs will wait in the queue on the cluster to which data are local longer than it takes to replicate the files to an arbitrary cluster. No brokering is provided which takes this factor into account, and the Job Description Language (JDL) does not allow users to specify whether input data should be moved to the job or other way around. Also, although there was some development of network monitoring and other tools aimed at optimising data transfer and job placement decisions, this functionality is not yet available.

Several important enhancements to job management were delivered right at the end of the EDG project, notably in the area of handling DAG (Directed Acyclic Graph) composite job definitions. This was successfully demonstrated in the context of CMS at the final EU review of the EDG project, and is extremely useful for defining the sub-job sequences involved in HEP workflows.

3.8. *Information Systems and Monitoring*

An Information System is crucial for the proper functioning of the Workload Management System, since it is used for resource discovery and the choice of an execution site, and should also be able to provide a solid basis for resource and job monitoring.

EDG initially adopted the Globus MDS (Monitoring and Discovery Service) system, based on LDAP, with information cached in a hierarchy of index servers. However, the use of LDAP imposes some constraints as it uses a fairly simple data model, and also additions to the schema, e.g., to allow an application to publish monitoring data, are fairly difficult. EDG also used an alternative cache with a Berkeley Database LDAP back-end (BDII), which is currently successfully used by LCG.

3.8.1. *R-GMA*

A different system called R-GMA (Relational Grid Monitoring Architecture) [14] was developed within EDG. This uses a relational data model and a producer/consumer architecture, and allows applications to define their own tables in a straightforward way. An adapter was also written to allow R-GMA information to be republished in the LDAP format to allow interoperability.

A central Registry holds pointers to information producers but does not cache the information itself (at present the Registry is a single point of failure). However, caches can be set up as necessary, and for the EDG deployment a cache was defined locally to each RB in a similar way to the MDS-based Information Indices used in the first EDG release, and the BDIIs (Berkeley Database Information Indices) used in LCG.

R-GMA was available for the second major release of the EDG software from the summer of 2003. The initial version had very poor stability, with a mean time between failures of a few hours. Intensive debugging extended this to at least a week by the end of the project, but it was not considered sufficiently robust to be used by LCG, although they will shortly be deploying it in parallel with the LDAP-based system to allow further experience to be gained.

As a result of this it has not yet been possible to systematically evaluate the performance of R-GMA or compare it directly with alternative systems. However, it has been possible to get an idea of the capabilities of the system. Apart from the instability, R-GMA appeared to be fast enough that the limiting factors for job submission related to the RB rather than the information system, and no scalability problems were seen up to the 15 or so sites in the EDG testbed.

Both D0 [7] and CMS have successfully tested the use of R-GMA for job monitoring, with jobs publishing start and end records into application-specific tables. This was found to be relatively easy to implement and worked well within the restricted testing environment available. A significant advantage over direct publication to an external database is that jobs running on worker nodes do not need access to the WAN, only to the local machine running the R-GMA servlets.

An interesting study of the relative merits of MDS and R-GMA was made by Zhang et al. in 2003 [15].

3.8.2. *Schema Issues*

An important issue, particularly in the context of interoperability between different Grid projects, is that the interpreration of information items needs to be consistent across all providers and users. EDG initially used a custom-designed schema, but in 2002 the GLUE [16] project was set up to produce a uniform schema to be used by both US and European Grids. This has worked reasonably well, but some problems have been encountered in the precise interpretation of some of the values, for example how the total number of CPUs should be represented when hyperthreading is enabled or when multiple jobs can be run on a single CPU. For a large distributed Grid with no central control the definition of schema items needs to be as unambiguous as possible.

In addition, monitoring tools need to be developed to verify that the information being published is correct. In a number of cases sites have become “black holes”, attracting jobs which they are unable to run, due to incorrect information being published.

Also, the initial version of the GLUE schema has proved to be insufficiently flexible to reflect the wide variety of configurations at different sites. In particular it does not allow for the complex scheduling policies used in many batch systems, notably that different VOs may have different quotas and priorities. It would also be desirable to include information about jobs in the information system.

Possible changes to the schema have been in discussion for some time, but so far it has not proved possible to implement them. Schema changes are inevitably difficult because they affect the entire system, but Grid projects need to provide both technical and institutional mechanisms to allow schema evolution

in a way which is reasonably timely and no more disruptive than necessary.

3.8.3. *Monitoring*

Rather little was accomplished in the area of stable Grid monitoring tools on the applications testbed in EDG. The only reliable way to monitor job status was via the Logging and Bookkeeping database, and very little site monitoring was available. However for LCG, the DataTAG/GridIce [17] monitoring service has been developed, based on the EDG architecture and services and local site monitoring tools. It provides a convenient access via a Web browser to the information on Grid resources, accumulated activity reports, status of the Information System, etc.

3.9. *Site Deployment*

The EDG approach to site deployment with Grid middleware and applications software was driven by the requirement to have a simple procedure available quickly. Since the testbed was chosen to run a uniform OS based on RedHat Linux, the straightforward choice for the packaging tool was RPM. While this did not necessarily coincide with tools preferred by the HEP experiments, all the user groups provided their application software distribution kits in the required RPM format.

The total set of packages to be deployed on a site amounted to several hundred, some being specific to services and some being common. EDG decided to make use of the *Local ConFiGuration system* (LCFG) [18], a centralized site installation and configuration system. By configuring Grid services using LCFG, system administrators were largely spared the troubles of upgrading separate packages and going through complicated configuration process. Every time such a configured box boots, it retrieves the latest approved configuration from the central server and deploys it. The packages served by LCFG range from operating system upgrades to application-specific software. Therefore, by checking the latest software upgrades into LCFG one guarantees that the changes will be propagated to all the sites that use the service.

Not every site can move to a complete LCFG-driven installation. For example, if a site runs a non-standard OS, complete LCFG installation would imply installation of the official OS, possibly including repartitioning of disks and so forth. The latest LCFG version allows site owners to choose subsets of packages to be upgraded, providing more flexibility. If a site does not run the supported OS version it must find manpower to provide the porting. Packages have to be installed manually, which is quite time-consuming. A very sensitive issue is the fact that even worker nodes require some EDG-specific installation and configuration, which is often unacceptable for sites which share their resources with other user communities.

The fact that application-specific software can be installed in a centralized manner is convenient for production runs, lasting for months without changing the software. However, this does not scale for the case of user analysis and even non-standard production. When parts of application software are expected to be changed at times on an hourly basis, it makes sense to delegate responsibility for the software installation from system administrators to actual users - perhaps privileged ones only. This all but excludes the usage of RPM for packaging, and prompts for more user-friendly and localizeable installation tools. While such a possibility was not provided by EDG, it is under development by LCG. Such user-triggered software installation will be achieved by means of dedicated Grid jobs, downloading and unpacking the software from a cache to a dedicated area on the cluster, and publishing the corresponding tag in the Information System.

4. Quantitative Experience with Experiments' Data Challenges

Reasonable scale quantitative evaluations of the middleware have been conducted in three main phases. Firstly in the period July 2002 to April 2003 when, commencing with ATLAS and CMS, all experiments interfaced their Monte Carlo production systems to middleware on the EDG Application Testbed, and provided associated middleware evaluations. The second main phase ran from April 2003 to January 2004 when EDG were developing middleware, having learnt from the experiences of the first phase running, and LCG were setting up their operation based on EDG middleware, as well as waiting for stability in the EDG middleware. In this period the experiments used the LCG testbeds to prepare for their data challenges by learning how to distribute their software world-wide, and tested out the developing LCG infrastructure. During this second phase ATLAS and CMS performed some production running with early versions of the LCG system. In the third phase, from January 2004 to the autumn of 2004, all experiments ran large scale

data challenges on the LCG production service. This phase of running was much larger in scale, both in the number of sites used, and in the scale of the data processing.

In addition to experiment specific evaluations it has been shown to be important to have generic application tests running regularly to evaluate the overall performance of the system. This is a key element in monitoring the Grid, where the provision of ongoing site certification has been shown to be vital for running a production service.

4.1. *Experiment Running July 2002–April 2003*

Starting in July 2002 with the formation of an EDG/ATLAS Task Force there was increasing use of EDG middleware in the production environments of the experiments. The pioneering ATLAS work solved several major problems and paved the way for the first serious production use by an experiment, CMS, in December 2002. This work was very labour intensive, requiring a dedicated team of experts [6]. All of this work was done with configurations of the order of 10 sites on which it was possible to exert a reasonable amount of direct control.

The ATLAS jobs had the classical profile for experiment Monte Carlo running:

- A user submitted a job description to the Resource Broker.
- The Resource Broker performed job matching, using the information provided by both the Replica Catalog (to locate a Computing Element close to the input data) and the Information Service (to satisfy other job requirements).
- Specified input files were transferred via the Resource Broker to the chosen CE, and the job started.
- Selected output files were stored in the CERN Storage Element (with a possible move to the CERN CASTOR mass storage system). The EDG job management tools did not allow automated output file upload and registration, and so these operations were included in the production script, being a part of the batch job.

The early running by ATLAS was of very low efficiency, since they encountered many fundamental problems in the EDG 1.2 release. The EDG Workload Management System (WMS) relied on the standard Globus submission process, which uses the Globus GASS mechanisms. They provide staging of the executable and standard input files, and access to the standard output and error files from the client machine. During the tests, problems in the GASS cache system were discovered. One of the consequences was that a simultaneous submission of more than five jobs to the same resource caused job execution failures. The WMS was also seriously affected by problems with the faulty publication of service information, and poor response from MDS with increasing number of sites. In the period August to November 2002 several such serious bugs and configuration problems were resolved or worked around, preparing the way for larger scale running.

ATLAS conducted a second set of tests with EDG release 1.4.3 in February 2003. This was done in parallel with ALICE and LHCb stress tests. The aim of the test was not to stress the system, but to check the reliability of the testbed for a period of time of about 10 days, working within the system limitations identified by CMS. The test consisted of a systematic submission by two testers of about 10 jobs per day. During the two weeks of tests two users submitted 130 jobs. In the first week about 80% were successful, but in the second there were more failures due to local site problems.

Table 2 summarises the failures and successes according to the sources of failure. An important message from these tests was the importance of controlling quotas on services at sites, and the lack of appropriate tools to achieve this.

CMS conducted their EDG stress test in the period November to December 2002 with some follow-up running in January 2003. They measured the failure and success rate of Grid submitted CMS jobs. These measurements were possible thanks to redundant job tracking and monitoring. The redundancy was provided by the CMS job tracking system BOSS in addition to EDG WMS logging. Rates were measured for the two kinds of CMS simulation jobs submitted to the Grid, CMKIN and CMSIM. CMKIN jobs were called short jobs, because of their low CPU time requirement of about 10 seconds and light access to Grid services. In particular the Replica Catalog was only accessed to write and register (via the Replica Manager) the final produced files. CMSIM jobs were called long jobs, because of their large CPU time requirement of about 10 hours and heavy access to Grid Services. CMSIM jobs needed to find the input data file by querying the Replica Catalog, and then matched the required resources via the Resource Broker. Finally the produced output had to be written to storage and registered in the Catalog.

Tables 3 and 4 show the total number of jobs submitted to the EDG testbed for both CMS simulation

Table 2. Success/failure breakdown of the ATLAS controlled test with 130 jobs.

	EDG site	No. jobs
PBS failure	CERN	1
Can't read job wrapper output	IN2P3	4
Total failures due to EDG middleware		5
SE corrupted and a WN down	RAL	16
SE disk full at CERN	RAL	18
SE disk full at CERN	CERN	11
SE disk full at CERN	NIKHEF	20
NFS failure at CERN	NIKHEF	3
Total failures due to site problems		68
Success	CERN	40
Success	CNAF	15
Success	RAL	2
Total successful jobs		57

Table 3. EDG stress test classification of submitted jobs for the CMKIN-like processes.

	EDG evaluation	CMS evaluation	EDG 1.4.3
Finished correctly	5518	4742	1014
Crashed or bad status	818	958	57
Total number of jobs	6336	5700	1071
Efficiency	.87	.83	.95

steps (10500 jobs in about three weeks), with a breakdown of successfully finished and failed jobs. The first column shows the results obtained in 2002 as seen from the Grid monitoring, and the second column shows how many jobs actually produced valid physics results (the application monitoring). Note the difference between the Grid and application monitoring: for short jobs, the application monitoring shows fewer jobs successful because the registration of output data sometimes failed which is not recognized by the Grid monitoring. On the contrary, long jobs risked being marked failed by the Grid monitoring because of communication problems with the job or problems retrieving the job output, even though it may still have produced valid results. The third column shows the results obtained in January 2003 after the middleware had been upgraded.

The tables also provide information about the measured efficiencies to run jobs. A total of 6336 CMKIN jobs were launched into the Grid, and a total of 5518 were successful. The overall efficiency during the whole test for CMKIN jobs for the EDG evaluation of testbed and middleware performance turned out to be 87%. A total of 4340 CMSIM jobs were launched into the Grid, and a total of 1678 were successful, giving an efficiency of 39%.

The CMS work produced over 260K physics events, and marked a watershed in the use of EDG middleware, prior to its deployment in the first LCG production service.

The main residual problems following the CMS work, which were worked on in 2003, were

- MDS response and Information Index instability.
- Replica Catalog response for file registration.
- Job submission chain (Resource Broker, Job Submission Service and local scheduler interfaces).

The work by the four LHC experiments in Phase 1 was summarised at the CHEP03 conference in March 2003: ALICE [19, 20], ATLAS [21], CMS [22] and LHCb [23, 24].

Table 4. EDG stress test classification of submitted jobs for the CMSIM-like processes.

	EDG evaluation	CMS evaluation	EDG 1.4.3
Finished correctly	1678	2147	653
Crashed or bad status	2662	935	264
Total number of jobs	4340	3082	917
Efficiency	.39	.70	.71

BaBar [25] and D0 [26], whose work was also reported at CHEP03, also accomplished some interesting work in this period. BaBar were later to accomplish some production running on an LCG based configuration on the Italian LCG-based Grid and D0 were to perform the first evaluations of R-GMA for job monitoring.

4.2. *Experiment Running April 2003–January 2004*

This period was mainly dedicated to preparing the experiments' software for the large scale data challenges in 2004. Nevertheless, CMS continued to use Grid technologies regularly on a separate testbed running the first version of the LCG middleware. It consisted of about 170 CPUs and 4 TB of disk. The system produced 0.5 million generated events, 1.5 million digitised events and 0.6 million reconstructed events. This corresponded to about 10,000 jobs, 100 CPU-months and 2 TB of data. The overall failure rate varied from 5–10% to 25–30%. The main sources of failure were RLS (Replica Location Service) unavailability for several periods, site misconfiguration, network problems and hardware failures.

4.3. *Experiment Running January 2004–September 2004*

In January 2004 the large scale data challenges by ALICE, ATLAS and LHCb started. CMS also executed a data challenge, but made relatively little use of LCG WMS services. They concentrated on proving their computing model from the aspect of data flows between sites. This also involved an evaluation of the EDG metadata cataloguing software.

All three experiments implemented their production system on the LCG software stack in a similar way. They were all operating in a multi-Grid environment and each saw the LCG service as one, and 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.

Jobs were defined and submitted to the experiment-specific production database. A translator facility retrieved the jobs from the central database, translated the jobs to LCG JDL, and submitted them via the regular LCG job-submission tools. For ATLAS and ALICE, the job being sent to the 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.

4.3.1. *ALICE*

Phase I of the ALICE data challenge ran from February to May 2004. 56K jobs were successfully completed, each running for 7.5 hours. Out of these 11K used LCG job submission services. The jobs generated 26 TB of data in the CERN mass storage system. As a pure job execution service LCG provided an efficiency of 76%. In this first phase the ALICE data catalogue was used for file registration.

In July ALICE commenced phase 2 of the data challenge, and made more use of LCG data management services. A total of about 14k jobs were submitted, and the total efficiency of the LCG services (WMS + data management + storage) was seen as 62%, with data management errors being responsible for two thirds of the failures. Ensuing investigations have shown that this class of failures were mostly due to site configuration problems for software and hardware resources. Thus by late September the LCG

efficiency for ALICE had risen to 80%. However, the tracking down of site-related problems proved very painful.

4.3.2. *LHCb*

The LHCb data challenge started in May 2004 running mostly on LHCb-owned, non-LCG sites. Progressively, more and more LCG sites were commissioned and added to the LHCb pool of sites. The LCG share in the total volume of produced data grew from 11% in May to 73% in August, with a total of 43 LCG sites executing LHCb jobs. A total of 211K jobs was submitted to LCG, but 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 WMS giving an LCG efficiency of 61%.

The jobs were all complex, lasting 24 hours or more, and with multiple stages involving event generation, data digitisation and event reconstruction.

The breakdown of job statistics is given below. At least the active cancellation due to proxy expiration could have been circumvented by using longer-lasting proxies or proxy renewal techniques.

- 211K jobs were submitted.
- 26K jobs were cancelled by the production manager anticipating proxy expiry.
- 185K jobs remained to be submitted to LCG.
- 37K jobs were aborted before getting into running status.
- 148K jobs made it to running.
- 37K jobs were aborted during running.
- 113k jobs completed and were retrieved by LHCb.

4.3.3. *ATLAS*

ATLAS, who started their data challenge in the summer after ALICE and LHCb, executed their data challenge using the US and Nordic Grids as well as LCG. Phase 1 of the ATLAS challenge started in the first week of July. As of September 7th the number of successfully completed jobs was 36k, with a typical job duration of 1 day. For the period from August 1st the overall LCG job efficiency was 81%. The following shows the breakdown of successes and failures in the LCG service from August 1st to September 7th:

- 29303 jobs completed successfully.
- 750 jobs failed due to misconfigured sites.
- 1600 jobs failed due to failures in the WMS or related LCG services.
- 4350 jobs failed due to failures in data management, Storage Elements or related LCG services.

4.3.4. *Generic Testing*

The flavour of data challenge work in 2004, as described above, changed dramatically, with LCG having to offer services to multiple HEP VOs each with their own variety of technical problems associated with using the service. Hence the service support was extremely complex and time consuming, as was the debugging of problems. As a consequence, it was useful to have some simple generic test jobs which regularly exercised the system.

We developed a generic application-flavoured monitoring job which simply used the LCG sites 'blind' in the sense that it did not discriminate between sites in the LCG configuration. By contrast each experiment using the LCG service benefitted from a high level MDS/BDII configuration listing sites regarded as suitable for their experiment running, since in the LCG start-up phase one could not rely simply on the status information published by the site.

The generic testing consists of a test job submitted to LCG once an hour; the job is targeted at an input file replicated to each SE, with a ranking expression allowing it to go randomly to matching CEs. It checks that it can read the input file using RFIO (a standard HEP remote-access protocol), registers a file to the close SE, replicates it to a randomly-chosen remote SE, and finally copies it back to the batch worker node (WN). The job is submitted with a cron job running on an LCG UI node. The job only runs for a few minutes, so it does not test proxy renewal or stability over long periods. 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.

To illustrate the current level of performance we summarise the outcome for a ten-week period in summer 2004 which reflects in a compact way the variety of problems that a job can encounter on a large scale Grid service. The results are summarised in Table 5, which shows the number of jobs lost successively to each reason for failure, and as a fraction of the number of jobs remaining in the previous line (e.g., 7.9% of the 1652 jobs submitted by cron failed in submission).

Of the possible 1680 jobs, 28 were not submitted because the UI was rebooted, and for some reason the cron job did not restart automatically. The broker rejected a further 131 jobs. 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, one was probably a side effect of bugs in a batch system interface

Table 5. Breakdown of failures for generic test jobs.

	No. of jobs	Fraction	Remaining jobs
Potential number of jobs	1680		
Cron failure	28	1.7%	1652
Submission failure	131	7.9%	1521
BDII failure	169	11.1%	
Proxy expiry	18	1.2%	
Other	7	0.4%	
Aborted	196	12.8%	1325
Execution failed	65	4.9%	1260
Input file read failed	293	23.3%	
Remote SE failed	144	11.4%	
Job ran multiple times	77	6.1%	
Jobs with any errors	434	32.8%	826
Overall efficiency	826	49.2%	

and one may have been related to the upgrade to a new middleware version 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, 196 were aborted, 169 of those due to BDII failures. These again happened in bursts when the BDII was unstable. Another 18 had proxies expire due to a proxy re-use bug, which has since been fixed, while the other 7 were miscellaneous failures.

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 at individual sites.

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. 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, but some loss may be inevitable.

There were also 144 cases where the replication to a remote SE failed. That may happen because of an information system problem 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. In addition, some jobs run successfully in terms of registering files but are seen as failures by the broker, usually because the job output is 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 already counted above.

The final count of jobs that ran only once and showed no (detected) errors is 826. This would suggest an overall efficiency of around 50% based on the potential 1680 jobs in the period. However, this number depends very much on the details of the test job and the particular problems in the system during the test period, so it should be taken as a general indication of the level of performance rather than a precise measurement. However, a recurring theme in all evaluations, either experiment-specific or generic, has been the large influence of system and site configuration problems, which substantially dominate the effect of bugs or of faults in central services.

5. Summary of the Main Lessons Learned and a Look Forward to EGEE and the Development of Experiment Analysis Systems

Throughout the three years of the European DataGrid project the middleware has evolved and by 2003 reached a level where its components for job and data management have been successfully deployed onto a production service for LHC computing as provided by LCG. All 6 experiments participating in EDG have performed substantial data processing using EDG software, with job efficiencies reaching 85% in well managed production exercises.

The following have been the main lessons drawn from the project:

- It would have been better to start with simpler prototype middleware, and to have had more frequent incremental releases to the users. The integration of middleware to form a robust, working system has proved to be much harder than expected.
- The application groups should have worked closely with the middleware groups from the beginning, both in defining the architecture and planning the testing of the prototypes. The formation of Application/Middleware task forces from July 2002 demonstrated the crucial nature of such organisation.
- Having an application oriented team of experts working across applications was vital to the success of the project. This promoted synergy within the project, and formed a vital element of the intellectual backbone of the project.
- Site configuration and certification must be automated. Improperly configured sites are a major factor in job losses in the Grid. Middleware needs to be fault-tolerant and self-correcting in the face of configuration problems and other errors.
- Similarly, disk space management on SEs and WNs remains a serious souce of residual problems affecting overall efficiency.
- The applications await a stable uniform mass storage interface. It is expected that this will be provided by SRM implementations, supplemented by GFAL (the LCG-developed Grid File Access Library) allowing applications transparent access to Grid files.
- The applications need flexible schemes for individual user software installation on the Grid. The RPM scheme is too inflexible for individual use.
- Last, but definitely not least, security considerations need to be taken into account from the beginning, as it is difficult to make software secure retrospectively.

The project has collectively taken note of these lessons, and this input is being carried forward into the EGEE project where there is an HEP applications group acting as part of an overall applications activity. In addition to the ongoing data challenges there will be developments of the LHC experiments' analysis models which must cope with the use of the Grid by thousands of individual users. This poses special problems of random demands for data and CPU resources, all to be managed within the context of the VOs and their policies. The HEPCAL use case work has been extended [27], taking account of new requirements arising from individual user analyses, for example the management of interactive sessions, tracking the provenance of data though many transformations, and the provision of query facilities for data selection from vast datasets.

2004 has seen a major increase in the scale of the use of Grid technology by HEP experiments using the LCG service. Experiments have passed hundreds of thousands of jobs through LCG, being executed in a world-wide configuration of sites, totalling over 70 in September. The data produced is being used for physics analyses. The LCG job efficiencies for this running on a global Grid have ranged from 60% to 80%, with a major source of errors being site configuration, this key area being actively addressed now by the LCG operations team.

All the experiments are using more than one Grid, and with substantial success. We have moved now to a stage where Grid technology is being used routinely in physics data production, and where we are learning to refine the operational aspects of running world-wide Grids. In addition we await re-engineered middleware, benefiting from our start-up experiences, which will enable the community to use Grid techology for physics analyses conducted by thousands of users distributed world-wide.

Acknowledgements

We wish to acknowledge the support of the EU and our national funding agencies. We would also like to acknowledge the excellent relations HEP applications have had with the Project Office, the middleware and testbed groups and our other application colleagues in biomedicine and earth sciences.

Also throughout the project colleagues within LCG and the experiments have been fully cooperative in the accomplishment of common goals.

References

1. F. Carminati, P. Cerello, C. Grandi, E. Van Herwijnen, O. Smirnova and J. Templon, "Common Use Cases for a HEP Common Application Layer – HEPCAL", Technical Report, LHC Computing Grid Project, 2002, http://lcg.web.cern.ch/LCG/sc2/RTAG4/finalreport.doc.
2. I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid", *The International Journal of High Performance Computing Applications*, Vol. 15, No. 3, pp. 200–222, Fall 2001.
3. F. Carminati, P. Cerello, C. Grandi, O. Smirnova, J. Templon and E. Van Herwijnen, "LHC Requirements for GRID Middleware", *Presentation Given at 2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, 2003, http://www.slac.stanford.edu/econf/C0303241/proc/pres/285.PPT.
4. "EDG WP8–HEP Applications", DataGrid User Requirements and Specifications for the DataGrid Project, EU DataGrid Project, Deliverable D8.1, September 2001, http://edms.cern.ch/document/332409.
5. "EDG WP8–HEP Applications", TestbedI Assessment by HEP Applications, EU DataGrid Project, Deliverable D8.2, Feb. 2002, http://edms.cern.ch/document/334920.
6. "EDG WP8–HEP Applications", Testbed2 Assessment by HEP Applications, EU DataGrid Project, Deliverable D8.3, April 2003, http://edms.cern.ch/document/375586.
7. "EDG WP8–HEP Applications", Final HEP Application Testbed Evaluation, EU DataGrid Project, Deliverable D8.3, Dec. 2003, http://edms.cern.ch/document/428171.
8. I. Augustin, F. Carminati, J. Closier et al., "HEP Applications Evaluation of the EDG Testbed and Middleware", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
9. I. Foster, C. Kesselman, G. Tsudik and S. Tuecke, "A Security Architecture for Computational Grids", in *Proceedings of the 5th ACM Conference on Computer and Communications Security Conference*, November 1998, pp. 82–92.
10. I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *Intl. J. Supercomputer Applications*, Vol. 11, No. 2, pp. 115–128, 1997.
11. P. Kunszt, H. Stockinger, K. Stockinger, E. Laure, J.-P. Baud, S. Occhetti, J. Jensen, E. Knezo, O. Synge, O. Barring, B. Hess, A. Kowalski, C. Watson, D. Petravick, T. Perelmutov, R. Wellner, J. Gu, A. Shoshani and A. Sim, *The Storage Resource Manager Interface Specification Version 2.1*, Dec. 2003, http://sdm.lbl.gov/srm-wg/doc/SRM.spec.v2.l.final.pdf.
12. A. Chervenak et al., "Giggle: A Framework for Constructing Scalable Replica Location Services", in *SC'2002*, Baltimore, USA, November 2002.
13. B. Allcock et al., "Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing", in *18th IEEE Symposium on Mass Storage Systems and 9th NASA Goddard Conference on Mass Storage Systems and Technologies*, San Diego, April 17–20, 2001.
14. A. Cooke, A.J.G. Gray, L. Ma, W. Nutt, J. Magowan, M. Oevers, P. Taylor, R. Byrom, L. Field, S. Hicks, J. Leake, M. Soni, A. Wilson, R. Cordenonsi, L. Cornwall, A. Djaoui, S. Fisher, N. Podhorszki, B.A. Coghlan, S. Kenny and D. O'Callaghan, "R-GMA: An Information Integration System for Grid Monitoring", in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, Catania, Sicily, Italy, Nov. 2003, Lecture Notes in Comput. Sci., Vol. 2888, Springer-Verlag, pp. 462–481.
15. X. Zhang, J. Freschl and J. Schopf, "A Performance Study of Monitoring and Information Services for Distributed Systems", in *Proceedings of HPDC*, August 2003.
16. S. Andreozzi, C. Vistoli and M. Sgaravatto, "Sharing a Conceptual Model of Grid Resources and Services", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
17. S. Andreozzi, A. Ghiselli, C. Vistoli, S. Fantinel, G. Tortone and N. De Bortoli, "GridICE: A Monitoring Service for the Grid", in *Cracow 03 Grid Workshop*, Cracow, Poland, October 2003.
18. P. Anderson and A. Scobie, "LCFG: The Next Generation", in *UKUUG Winter Conference*, UKUUG, 2002.
19. F. Carminati et al., "ALICE Experience with EDG", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
20. S. Bagnasco, R. Barbera, P. Buncic, F. Carminati, P. Cerello and P. Saiz, "AliEn – EDG Interoperability in ALICE", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
21. G. Poulard et al., "ATLAS Data Challenge 1", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
22. D. Bonacorsi, P. Capiluppi, A. Fanfani et al., "Running CMS Software on GRID Testbeds", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
23. N. Brook, A. Bogdanchikov, A. Buckley et al., "DIRAC – Distributed Infrastructure with Remote Agent Control", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
24. K. Harrison, W.T.L.P. Lavrijsen, P. Mato et al., "GANGA: A User-Grid Interface for Atlas and LHCb", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
25. D. Boutigny, D.H. Smith, E. Antonioli et al., "Use of the European Data Grid Software in the Framework of the BaBar Distributed Computing Model", in *2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, La Jolla, California, 2003.
26. K. Riesselmann, "DZero Breaks New Ground in Global Computing Efforts", *Fermi News*, Vol. 27, No. 2, pp. 2–5, Feb. 2004.
27. F. Carminati and J. Templon (eds.), *Common Use Cases for a HEP Common Application Layer for Analysis – HEPCAL II*, http://lcg.web.cern.ch/LCG/SC2/GAG/HEPCAL-II.doc.