HEP Applications and their experience with the use of DataGrid middleware

I.Augustin, F.Carminati, J.Closier, E.van.Herwijnen and A.Sciaba
CERN

D. Boutigny and J.J. Blaising
LAPP/Annecy,CNRS,France

V. Garonne and A.Tsaregorodtsev
Marseille,CNRS,France

K. Bos, D.Groep, W.van.Leeuwen and J.Templon
NIKHEF,Holland

P. Capiluppi, A.Fanfani and C.Grandi
Bologna,INFN,Italy

R. Barbera
Catania,INFN,Italy

E. Luppi
Ferrara,INFN,Italy

G.Negri, L.Perini and S.Resconi
Milan,INFN,Italy

M. Reale and A.De. Salvo
Roma1,INFN,Italy

S.Bagnasco and P. Cerello
Turin,INFN,Italy

O. Smirnova
Lund University,Sweden

O. Maroney
Bristol University,UK

F. Brochu
Cambridge University,UK

D. Colling
Imperial College,UK

F. Harris and I. Stokes-Rees
Oxford University,UK

S. Burke
RAL,UK

April 1, 2004

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. Particular reference is made to experiment 'data challenges', and a forward look is given to necessary developments in the framework of the forthcoming EGEE project.

Keywords: HEP,Applications,Grid,middleware,EDG,EGEE,LCG,LHC

1. High Energy Physics (HEP) Computing

Computing has always been important for nuclear and particle physics as many of the 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) [1] at the European Laboratory for Particle Physics CERN [2] in Geneva, will be looking for reactions which 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 [3] came at a time of the LHC project [4] 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
2) detector simulation
3) detector response simulation
4) background simulation.

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

In 2), the detector simulation step of the calculation, 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 is calculated. And at each step the probability is calculated that the particle may decay into other particles or annihilate or be absorbed.

In 3), the last step of the simulation, the detector response to the traversal of all the tracks 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 also 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.

At all steps of this program sequence data has to be fed to the running code, both through data files and experiment specific data bases accessible to the running program. 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 sample of reconstructed data throughout the collaborations.

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 a 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. So there is a need for the data needed for these analyses to be even more distributed than the raw, unprocessed data.

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

The 4 LHC experiments (ALICE, ATLAS, CMS and LHCb) have all developed detailed planning for computing leading up to the LHC

Table I. Summary of CMS Data Challenge 2004

		simulation	digitisation	reconstruction
time allocated	months	5	2	1
CPU per event	KSI2K s	160	24	12
input event size	MB	0.04	0.90	0.70
output event size	MB	0.90	0.70	0.15
events per job		250	1000	250
CPU per job	KSI2K s	40000	24000	3000
output size per job	MB	225	700	37.5
duration of job	hours	20.20	12.12	1.19
nr. of input files/job		1	4	3
nr. of output files/job		1	3	16
tot. CPU at requested eff.	KSI2K month	4115.2	617.3	308.6
tot. CPU power	KSI2K	822.80	308.55	308.00
tot. nr. of CPU's requested		1496	561	440
tot. output data size	TB	45	35	7.5
tot. nr. of files produced		200000	150000	3200000
event freq. (at 100% eff.)	Hz	5.143	12.856	25.667
job submission freq.	jobs/day	1777	1110	8870
output freq.	MB/s	4.6	9.0	3.9
file registration freq. (at 100% eff.)	Hz	0.0206	0.0386	1.6427

start-up in 2007. This includes large scale *data challenges* taking place in 2004. As an example, Table I summarises the characteristics of the data challenge being performed in spring 2004 by CMS.

The data challenge is logically divided into 3 data processing phases (*Generation & Simulation*, *Digitisation* and *Reconstruction*), followed by a data distribution phase. To reach a reconstruction event rate of 25 Hz, which corresponds to 25% of the LHC startup rate in 2007, 50 million events must be processed in one month. This work is being 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.

The number of CPUs dedicated to the different steps of the challenge has been calculated assuming 75% efficiency during the allocated time. From the characteristics of the programs, the interaction frequencies with the Grid Workload Management System and the Replica Manager have been extracted. For the reconstruction phase about 9000 jobs per day are submitted to the WMS, 1.6 files per second registered to the RM (2 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 is using components both from Grid projects and from CMS.

2. HEP computing and the Grid – the model and the use cases

For many years the HEP community has been running large distributed computing infrastructures. The Grid paradigm offers an opportunity to have a homogeneous view of a world-wide virtual computing system, and potentially access to world-wide resources.

The HEP computing user expects to see a widely 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 overall management of the necessary databases.

2.1. Users and Virtual Organisations (VO)

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 [11].

2.2. Actors and use cases

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 a LHC experiment. The same physical person may play more than one role depending on her activity at a given moment. Examples of the actors are:

- Individual Users
- Production managers

- Managers (spokesperson, computing coordinator, physics coordinator)
- Developers of experiment-specific software tools;
- Software maintainers and distributors (librarians).

2.3. The HEPCAL use cases

The HEPCAL document [11], [13] 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 project [14].

HEPCAL distinguishs two logical entities containing data: catalogues and datasets. A catalogue is a collection of data that is updatable 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 [5] or Oracle [6] databases. The catalogues typically contains two types of information, firstly Grid-specific such as information about replicas, and secondly user-defined information, so-called metadata, giving information related to the data which is application specific.

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 in given in section 4.

3. Application evaluations of Grid software components in EDG

Since late 2001 a series of evaluations were made of EDG middleware. These fell into 2 classes, firstly generic evaluations of the basic components, and secondly evaluations conduced by the HEP experiments interfacing the middleware to their production systems. Below we overview the results of these evaluations component by component. In section 6 we summarise particular important results obtained by the experiments. Throughout the EDG project HEP applications produced 3 major evaluation reports, [15], [16], [17] for the EU, and an overview for CHEP03(Computing in High Energy Physics), [18].

3.1. Storage Elements

In principle a Storage Element (SE) is a managed interface to mass storage, and the required functionality has emerged gradually during the EDG project. The initial concept only related to Mass Storage Systems (MSS) with a large tape-based robotic store, which in the HEP field have usually only been deployed at a small number of large laboratories, and generally have customised, mutually incompatible management software. However, tape robots are now being deployed locally in some Universities, and many sites have multi-TB disk servers, so a need is developing for the SE concept to be extended to provide an interface to a wider range of 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, before any interface had been developed, a minimal SE was deployed which essentially consisted of a 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.

During the course of the EDG project a number of collaborators, mainly in the HEP field, have been engaged in the definition of the Storage Resource Manager (SRM) standard, which is intended to provide a standard Grid-aware web service-based interface to any MSS. EDG has participated in this work and the EDG SE middleware has evolved in line with the standard. Other SRM implementations are being developed by other projects. However, as yet SRM is 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.

3.2. Replica Catalogues

File catalogues are at the heart of a data Grid, so performance and integrity are crucial. For the first major release of the EDG software the LDAP-based Globus Replica Catalog 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. The design of the system envisages a local catalogue at each site with hierarchical indices to aggregate the information, and evaluation awaits deployment in that mode. Even in the relatively small testbed deployed so far scalability problems can be seen, for example running 600 file registrations in parallel shows an 0.5% failure rate due to catalogue communication errors (there are no retries for such errors in the current release).

There are also performance issues in the current implentation 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. These performance issues will be addressed in the forthcoming EGEE project.

A general security mechanism for web services, allowing both authentication and authorisation using extended GSI proxies, has been developed within EDG. This awaits deployment and thorough testing. This is vital for use in a production system.

3.3. 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. Queries on these properties are used to define the input datasets to be processed. Such metadata is highly application-specific and it is unlikely that completely generic middleware solutions can be provided, but it is essential that standard interfaces are available

to allow application metadata catalogues to be integrated with Grid systems.

Metadata is 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 in the metadata catalogue. The system as deployed is essentially a prototype. 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, but this is not yet implemented.

3.4. Replica Optimisation

A significant goal of a data Grid is to optimise access to data, both in terms of running jobs at sites where they can most efficiently read their data, and in replicating files from site to site in the most efficient way. The EDG middleware has optimisation support in a variety of areas, including network monitoring to measure transfer rates between sites, a service to estimate a cost function for the replication of a file between specified Storage Elements, and some support in the job submission system to schedule jobs on the basis of the the overall cost of access to input data (although there is no support for automatic replication of files before a job is run). The optimisation features have been integrated in the EDG release and await thorough evaluation

3.5. 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. The first release of the EDG middleware used software called GDMP, 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) so far 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 the 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, and new versions of the code are predicted to be much faster.

The Replica Manager is designed to support the web service security interface integrated into the catalogues and SEs, but testing so far has been limited because security is not yet enabled in the deployed systems. 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. For example, there is no single operation to replicate all files with an LFN matching a given pattern, or with a creation date between specified points. Such operations can in principle be embedded in the application-level code, but are clearly likely to be common between a wide variety of applications and hence are a good candidate for middleware support. 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.

d) A client-server model would also remove the current need for access to the wide area network from batch worker nodes, which is deprecated at many sites, and would enable the scheduling of automatic replication under given conditions.

3.6. Job handling

3.6.1. *Requirements*

Certain specifics of HEP computing had to be taken into account while developing workload management services. Typical constraints to be met by EDG could in general be characterized as follows.

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;

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

2. in absence of specific requirements, the job must 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 [11] 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.

3.6.2. *Performance*
During the runs on the EDG testbeds, the Workload Management System was put under heavy load. Typically, it performed up to expectations, allowing users to submit thousands of simultaneous jobs and to be used in production runs with high efficiencies. However various problems were encountered by users. Below is the list of major problems related to use of the EDG Workload Management System (while very few failures can be attributed to the job handling system as such, several may be avoidable with a different implementation):

– 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 retrived 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 lead to many jobs being suspended or lost altogether.

– Transfer of Input and Output Sandboxes of all the 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 the 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 download the file to an arbitrary cluster. No brokering is provided which takes this factor into account, and JDL does not

allow users to specify whether input data should be moved to the job or other way around.

At the end of the project several important enhancements were achieved, for example in the area of handling DAG (Directed Acyclic Graph) job definitions. This was successfully demonstrated in the context of CMS, and is extremely useful for defining the sub-job sequences involved in HEP job definitions.

3.7. Information systems and monitoring

An Information System is crucial for the proper functioning of the Workload Management System. A scalable, robust and reliable Information System should be able to provide a solid basis for resource and eventually job monitoring.

3.7.1. *MDS*

The MDS based Information System turned out to be the least reliable service of the first EDG testbed deployments. There were serious problems with timeout handling and the implementation of indexing services proved not to be scalable, causing servers to drop queries under heavy load and effectively emptying the cache. This led to eventual resource "invisibility".

3.7.2. *MDS amendments: BDII, GLUE schema*

Poor performance of the Information System led to EDG developers investigating a variety of solutions. One of them was to re-implement the Information Index using the Berkeley Database Information Index(BDII). This is an LDAP server backed by a Berkeley Database which provides information according to the GLUE schema, and is populated by regular queries to information providers at each site. Although problems remained with stale information there were dramatic increases in overall stability, and this model has been further refined by LCG who have configured several BDIIs on a regional basis and provided redundancy in the configurations.

It also became clear that the original EDG schema used by MDS was inadequate: many important system parameters were not published, and the entire structure of the schema was complex, largely unmaintainable and extremely difficult to interpret, especially by end-users. This triggered a cross-Grid project, developing the so-called *GLUE schema* [7], specifically oriented towards clusters and storage facilities. This schema provides more flexibility and allows sites to publish more parameters relevant not only to workload management, but also allowing extensive monitoring. It was deployed in the latest phase of EDG

and served as a basis for the information schema used by the Grid2003 sites in US.

3.7.3. *R-GMA*

With a view to replacing MDS, the *Relational Grid Monitoring Architecture* (R-GMA) [8] was designed and deployed in the later stages of EDG. It implements a producer-consumer architecture using a relational data model, without a specific built-in hierarchy. To be compatible with the RB, the information from the R-GMA tables is converted to LDIF format, complying with the GLUE schema. This new R-GMA Information System has been extensively tested during the final months of the EDG project. More development and comprehensive tests of large configurations are required before R-GMA can provide reliable services, and LCG still deploys BDII/MDS, having invested very considerable effort in developing and customising the IS configuration essential for stable operation. R-GMA does, however, offer some significant advantages over MDS, particularly in the ease with which users can add their own tables to the schema and create their own data producers. Both D0 [17] and CMS have successfully used these facilities for job monitoring.

3.7.4. *Monitoring*

Given that the information services were constantly under development, which resulted in instabilities and often insufficient information provision, 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 the LCG, the DataTAG/GridIce [9] 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.

3.8. 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 the site amounted to several hundreds, some being specific for services and some being common. EDG conveniently decided to make use of the *Local ConFiGuration system* (LCFG) [10], a centralized site installation and configuration system. By configuring Grid services to use 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.

Evidently, not every site owner would like to resort 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 re-partitioning of disks and sp forth. Another annoying problem encountered during the tests was related to the fact that some applications software had cyclic dependencies, and in the LCFG context it prevented entire nodes from being booted at all. However, the latest LCFG version allows site owners to choose subsets of packages to be upgraded, providing more flexibility.

Deployment of the EDG middleware on a non-dedicated site is much more problematic. The site must run the supported OS version, or 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 was required to be installed in a centralized manner was convenient in the sense that users did not have to bother about propagation of the software across the testbed. This suits well the pattern of heavy production runs, lasting for monthes 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 (see Section 2.2). 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, down-

loading and unpacking the software from a cache to a dedicated area on the cluster, and publishing the corresponding tag in the Information System.

4. 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.

4.1. Basic (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 concerns the job submission system. One specifies a variety of job control operations, including suspend/resume, requeueing and changing priority. In the current EDG middleware the only supported operation is cancellation, and in this case any existing output files are lost. Two more cases involve queries. One specifies the retrieval of information about a queued or running job, such as position in a queue, cpu time consumed, I/O rates and volumes, and access to the `stdout` and `stderr` files. Essentially none of this information is currently available; the WMS only stores information about high level state changes, and the information system does not have information about individual jobs. The other missing query function concerns querying the job catalogue for jobs matching a set of criteria. The current system only supports a query specifying a submission date and time between two points.

The final unimplemented case concerns the uploading of a file which is known to be a replica of an existing file, and the registration of it under the existing file identifier (GUID). This is as opposed to using the Grid middleware itself to perform replication. Technically this use case is achievable using low-level commands to edit the Replica Catalogue directly, but there is no high-level support for it. The motivation for this use case was to maintain the possibility of data being shipped on a physical medium, but there may also be other uses, e.g. for files being replicated between Grids using different middleware. There would be a need to maintain integrity in such a situation, e.g. by using a checksum to confirm that the new file is in fact a replica.

4.2. Security (5)

Two uses 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 the VOMS system in collaboration with the DataTAG project, which should allow much more flexible control of VO-based authorisation, but so far this is not fully integrated with the other EDG middleware. 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.

4.3. Metadata (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.

4.4. Virtual Data (2)

The virtual data concept implies that instead of storing data in a file, a recipe to generate the data is stored in a catalogue, and the files are materialised on demand. This was out of the scope of EDG, and is likely to require substantial further work to implement.

4.5. Optimisation (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. 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. This was one of the goals for the EDG WMS, adapting the Condor DAGMAN software, but the functionality is not fully integrated in the deployed system.

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.

4.6. Application catalogues (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.

4.7. Application interfaces (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, e.g. to process a large number of files, and a third relates to storing user-defined metadata about jobs in the WMS job catalogue.

Three use cases concern specialised kinds of jobs: specification of input data via a metadata query, verification of the functionality of application software, and validation of the content of a dataset, either in a standalone job or as the final stage of a data production job.

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.

5. 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 most 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 affect the operation of an HEP experiment.

5.1. Authorisation

Unlike many Grid applications, 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 GSI proxy, and can be parsed by VOMS-aware middleware to enforce authorisation decisions.

VOMS appears to be a promising way to solve the authorisation needs of HEP experiments. Unfortunately integration was not finished in the lifetime of the EDG project, so practical experience has not been gained. This will be obtained soon in the LCG/EGEE projects.

5.2. 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

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 requires 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.

6. Some current experience with experiments and data challenges

Starting in July 2002 with the formation of an EDG/Atlas Task Force there has been increasing use of EDG middleware in the production environments of the experiments. The pioneering Atlas work solved several serious problems asociated with both Globus and EDG software 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 concentrated team of experts [16]. Ensuing work in 2003 [17] has seen job efficiencies of 80-90% obtained by several experiments with a reasonable amount of support effort. Also the EDG software has been used in the LCG production environment with good success.

We summarise below some of the work accomplished in early 2004 by the first two experiments, ALICE [20] [23] and CMS [21], to use the LCG service for physics production. The other LHC experiments, Atlas [24] and LHCb [19] [22] will commence production work on LCG in May 2004 building on their work in 2003. Also two non-LHC experiments, Babar [25] and D0 [26], currently running and taking data at SLAC and Fermilab respectively, will be using LCG for data processing.

6.1. ALICE

AliEn (ALICE Environment) is a Grid system for large scale job submission and distributed data management developed by ALICE, the CERN LHC heavy-ion experiment. With the aim of exploiting LHC Computing Grid resources to run AliEn-managed jobs for simulation and reconstruction, the problem of AliEn-LCG interoperability was addressed and an interface was designed and tested. The contact points, acting as gateways between the two systems, are the "Interface nodes": LCG User Interface machines that run also, as services, the full AliEn

site suite: Storage Element (SE), Computing Element (CE), Cluster Monitor and File Transfer Daemon (FTD). The interface CE gets jobs from the AliEn server and submits them to the LCG resource broker, taking care to handle file registration and staging across the system boundary. Therefore, a LCG Resource Broker is seen by the AliEn server as a single Computing Element, while LCG storage is seen by AliEn as a single, large Storage Element; files produced in LCG sites are registered in both the LCG Replica Catalogue and in the AliEn Data Catalogue, thus allowing access from both systems (Fig. 1).

The access to AliEn or LCG/*Grid.it* (Grid.it is a set of Italian sites configured with LCG software) is fully transparent: the AliEn frontend is used for the job configuration, while the load is automatically shared between resources under the direct control of AliEn and resources configured with LCG middleware.

All the files generated by a job running on an LCG site that must be permanently stored are doubly-registered in both (AliEn and LCG) Replica Catalogues, thus ensuring accessibility from both systems.

Whenever an output file generated by an AliEn job running on some LCG site is to be stored and registered in the catalogue, the CloseSE (in LCG terms) is determined by AliEn code (running on the WN) from the `.BrokerInfo` file generated by the RB. The file is then saved in the selected SE and registered in the LCG Replica Catalogue (uniqueness of both the Logical and LCG-side Physical File Names is enforced by AliEn). An "intermediate" file name of the form `LCG://<VO>/<LFN>` is generated using the LCG Logical File Name and the Virtual Organisation name to allow for multiple VO/RC. The file is finally registered in the AliEn Data Catalogue using this "intermediate FN" as Physical File Name; the file is thus seen by AliEn as "stored in LCG", regardless of the physical location of the LCG Storage Element on which it resides. One advantage of this approach is that if the file is replicated (using some of the LCG Replica Management tools) onto another LCG Storage Element, AliEn would be able to access any of the copies of that file without having to modify its database entry.

File accessibility is guaranteed in both directions, i.e. jobs running on either side of the interface can access data on both systems. However, since AliEn always tries to send a job where the needed data are, the basic case (jobs running on LCG nodes requiring data residing in LCG Storage Elements) is also the most common.

The interface is presently being used in production for the ALICE Data Challenge, started in Feb., 2004. Two interface sites are configured, pointing to LCG and *Grid.it* resources, respectively. After two weeks of production, LCG (*Grid.it*) processed about 43% (12%) of data, corresponding to a total of more than 7000 events at the LHC energy.

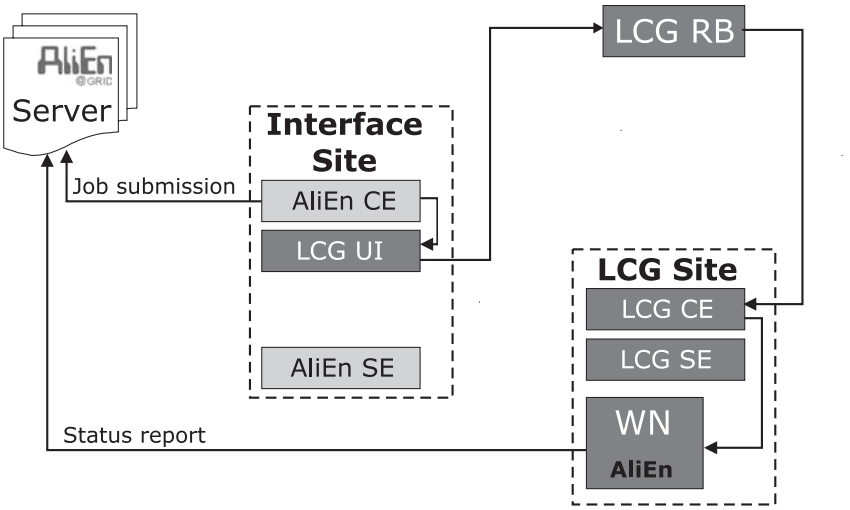

Figure 1. The interface job submission cycle. Jobs are configured on the AliEn Server and pulled by the AliEn CE on the interface site. The AliEn JDL file is then translated to a GLUE-compliant format and the job is forwarded to the LCG Resource Broker for submission to a LCG CE. When it starts on a LCG Worker Node, it begins reporting about its status directly to the AliEn Server, until completion.

It should be noted that one such ALICE event comprises maybe 80 thousand particles, and typically takes 24 hours to process on a 1GHz CPU.

6.2. CMS

During Data Challenge 2004 (DC04) CMS uses the EDG/RLS both as a POOL catalogue (POOL is an LHC project to provide cataloguing services to the experiments) and as a file catalogue. In general interaction with the RLS, and in particular file registration, is done through POOL commands. Adding or removing physical file names is done through the Local Replica Catalogue (LRC) C++ API. File replication and deletion is done via the tool chosen by each Tier-1. Supported tools are: the LCG Replica Manager (RM), the Storage Resource Manager (SRM) and the Storage Resource Broker (SRB). The registration of files was done via the Replica Manager commands. The DC04 setup includes an RLS catalogue at CERN and a replica of the catalogue using the ORACLE multi-master mirroring technique at CNAF in Bologna.

The sequence of the operations on the RLS catalogue is given below.

1. The reconstruction job runs at CERN in a local environment, using a local XML catalogue. A dedicated agent publishes to the RLS the new files produced by all the jobs starting from the XML file.

2. The reconstructed events are assigned for transfer to the Tier-1 regional centres by the Data Transfer Configuration agent depending on their properties, which are recorded in the RLS. All the information on the data transfer processes is kept in a CMS dedicated database (Transfer Management Database, TMDB) located at CERN.

3. Dedicated agents copy files to the supported (RM, SRM and SRB) data servers (Export Buffers, EB) and remove them once they have been transferred.

4. At different Tier-1s there are agents that replicate files locally using the preferred transfer tool. Interactions with the RLS are done by the Replica Manager if using the RM, the GMCAT synchronisation process if using SRB or directly by the Tier-1 transfer agent if using SRM. In some cases a local MySQL POOL catalogue is maintained. In this case the Tier-1 transfer agent also reads the full file record from the RLS and publishes it to the local catalogue

5. The analysis jobs run either on local Tier-1 farms or on Grid resources. When submitted via a Resource Broker a query to the RLS is done by the RB to send the job close to the data.

6. The analysis jobs, either submitted to Grid resources or to local farms, may use the central RLS catalogue at CERN, the RLS mirror at CNAF or a local MySQL catalogue if available. In all cases this process may imply the creation of local XML catalogues as done for the CERN reconstruction.

RLS has worked well as a file catalogue, and now evaluations are proceeding for its use together with POOL metadata stored on the Replica Metadata catalogue(RMC). Currently some scaling issues are being investigated.

7. Summary of the main lessons learned and a look forward to EGEE and the development of experiment analysis systems

Throughout the 3 years of the DataGrid project the middleware has evolved and as of 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 90% 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.
- The information system is the nerve centre of the Grid. We look to RGM-A to provide the long term solution to the scaling problems encountered with MDS.
- 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 space management on SEs and WNs remain 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 (Grid File Access Library) allowing application 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.

The project has collectively taken note of these lessons, and this input is being carried forward into the EGEE project where there will

be a 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 [12], 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.

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 application have had with the Project Office, the middleware and testbed groups and our other application colleagues in Biomedicine and Earth Sciences.

Throughout our work our working colleagues within LCG and the experiments have been fully cooperative in the accomplishment of common goals.

References

1. Large Hadron Collider
   http://user.web.cern.ch/user/Index/LHC.html
2. CERN
   http://user.web.cern.ch/user/cern.html
3. EU DataGrid project
   http://eu-datagrid.web.cern.ch/eu-datagrid/
4. LHC Computing Grid project
   http://lcg.web.cern.ch/LCG/default.htm
5. Objectivity database
   http://www.objectivity.com/
6. Oracle database
   http://www.oracle.com/
7. GLUE schema – conceptual model and implementation, CHEP03
   http://datatag.web.cern.ch/datatag/presentations/chep2003a.ppt
8. Relational Grid Monitoring Architecture
   http://www.r-gma.org/
9. GridIce – DataTAG system for grid monitoring
   http://server11.infn.it/gridice/
10. Local ConFiGuration system
    http://www.lcfg.org

11. Common Use Cases for A HEP Common Application Layer (HEPCAL)
http://lcg.web.cern.ch/LCG/sc2/RTAG4/finalreport.doc
12. Common Use Cases for a HEP Common Application Layer for analysis - HEPCAL II
http://lcg.web.cern.ch/LCG/sc2/GAG/HEPCAL-II.doc
13. LHC requirements for GRID middleware,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/pres/285.PPT
14. DataGrid User Requirements and Specifications for the DataGrid Project,Sep 2001
https://https://edms.cern.ch/document/332409
15. Testbed1 assessment by HEP applications,Feb 2002
https://https://edms.cern.ch/document/334920
16. Testbed2 assessment by HEP applications,April 2003
https://edms.cern.ch/document/375586
17. Final HEP Application Testbed evaluation,Dec 2003
https://edms.cern.ch/document/428171
18. HEP Applications evaluation of the EDG Testbed and Middleware,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/THCT003.PDF
19. DIRAC Distributed Implementation with Remote Agent Control,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/TUAT006.PDF
20. ALICE experience with EDG,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOCT002.PDF
21. Running CMS software on Grid Testbeds,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOCT010A.PDF
22. Ganga: The ATLAS and LHCb User Interface to the Grid,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/TUCT002.PDF
23. AliEn - EDG interoperability in ALICE,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/TUCP005.PDF
24. ATLAS Data Challenge 1,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOCT005.PDF
25. Use of the European Data Grid software in the framework of the Babar distributed computing model,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOCT004.PDF
26. DZero breaks new ground in global computing efforts,Fermi News, Volume 27, February 2004 http://www-d0.fnal.gov/computing/reprocessing/
27. The GENIUS Grid portal,CHEP03
http://www.slac.stanford.edu/econf/C0303241/proc/papers/TUCT001.PDF

Appendix

A. Glossary of terms and acronyms

Alice: An LHC experiment.

Analysis: Processing reconstructed data to test physics theories. This may be an organised production or may be chaotic processing by individuals.

Atlas: An LHC experiment.

BaBar: An experiment at SLAC.

BDII: Berkeley Database Information Index. An LDAP server backed by a Berkeley Database which provides information according to the GLUE schema, and is populated by regular queries to information providers at each site.

Calibration: The response of detectors must be calibrated, for example to allow a signal level to be converted into a particle energy. Calibration constants are usually stored in databases.

Catalogue: A read/write collection of data (typically one or more databases).

CE: Computing Element, a Grid-enabled computing resource. In EDG a CE is mapped to a queue in a batch system.

CERN: The European particle physics laboratory in Geneva.

Classic SE: An SE with no management middleware. Supports GridFTP for data transfer.

CMS: An LHC experiment.

CNAF: An HEP laboratory in Bologna which hosts a large computing facility.

Condor: A US project which provides tools for job matching and submission.

D0: An experiment at Fermilab.

DAG: A Directed Acyclic Graph, allowing dependencies between jobs to be specified.

DAGMAN: Condor Middleware which manages sets of jobs defined with DAGs.

Data Challenges: In advance of the startup of the LHC in 2007 experiments are performing a series of increasingly large scale exercises to ensure that computing models and systems are able to cope with the expected data volumes and rates.

DataTAG : An EU project closely related to EDG.

Digitisation: Part of the Simulation process, which takes simulated particles traversing the experiments, simulates the response of detector elements and produces digitised output in the same format as from the real detectors.

DMS: Data Management System, the middleware which provides high-level data management functions.

DS: Dataset, a collection of read-only data (typically a set of files).

EDG: The European DataGrid project, an EU project which ran for three years from 2001-04.

EGEE: Enabling Grids for E-science in Europe, the successor project to EDG.

Event: The set of particles emerging from a single collision of two incoming particles in a detector. Also the digitised data resulting from such a collision.

File Catalogue: A Catalogue which incorporates one or more Replica Catalogues together with middleware-level Metadata to allow management of files.

Fermilab: An HEP laboratory near Chicago.

GGF: The Global Grid Forum.

GDMP: The Grid Data Mirroring Protocol. Middleware which provides automatic file replication via subscription between pairs of Storage Elements.

GFAL: Grid File Access Library. Middleware developed by the LCG project to allow transparent access to files stored on the Grid.

Globus: A US Grid project which provides the foundation middleware for EDG.

GLUE: Grid Laboratory for a Uniform Environment, a joint project between EDG and US HEP Grid projects to define standards to enable interoperability, in particular a common information schema.

GMA: The Grid Monitoring Architecture, defined by the GGF, which allows data consumers to locate data producers using a registry.

GMCAT: Part of the SRB DMS.

GridFTP: Grid File Transfer Protocol. The Globus Grid extensions to the standard FTP protocol.

Grid map file: A file used to map GSI credentials to local Unix accounts.

GSI: Grid Security Infrastructure. The Globus middleware which provides the basic security framework.

GUID: Globally Unique IDentifier, a system-generated identifier which is guaranteed (or highly likely) to be unique.

HEP: High Energy Physics (particle physics).

HEPCAL: An LCG document specifying a number of Grid Use Cases for HEP applications.

IS: Information System, Grid-enabled middleware for the distribution of information.

JDL: Job Description Language, uses the Condor Classads library to describe job characteristics and requirements.

Job Catalogue: A Catalogue containing information for each job managed by the WMS.

LB: The EDG Logging and Bookkeeping system which manages information about the lifcycle of a job.

LCFG(ng): Local ConFiGuration (next generation), an installation and configuration tool developed by EDG from a system written at Edinburgh Univ.

LCG: The LHC Computing Grid project, which will provide the computing infrastructure for the LHC.

LDAP: Lightweight Directory Access Protocol. A protocol for retrieval of hierarchically-structured data.

LDIF: LDAP Data Interchange Format. A standard textual representation of data used in LDAP.

LDN: Logical Dataset Name, a unique name referring to a Dataset.

LFN: Logical File Name, a user-defined name for a single file.

LHC: The Large Hadron Collider, the next generation of particle accelerators, located at CERN and due to start operation in 2007. There will be four large particle detectors, called Alice, Atlas, CMS and LHCb.

LHCb: An LHC experiment.

LSF: Load Sharing Facility, a batch scheduler.

MDS: Monitoring and Discovery Service. The Globus middleware to provide a Grid Information System, based on LDAP.

Metadata: Data which describe or relate to the content of a file or Dataset.

Metadata Catalogue: A Catalogue which stores application-defined Metadata and allows the selection of Datasets via queries.

Monte Carlo: Simulation of particle collisions and the response of particle detectors using random numbers.

Production: An organised processing to produce a large number of simulated or reconstructed data files.

MSS: Mass Storage System, a large-scale managed storage facility, typically incorporating a tape storage robot.

OpenBSD: An open-source version of the BSD Unix OS.

OS: Operating System.

PBS: Portable Batch System, a batch scheduler.

POOL: An LCG project to develop high-level Dataset Catalogues and management tools for the LHC experiments, which has been interfaced with the EDG RM.

Raw Data: The digitised output from particle detectors, or a simulation thereof.

RB: Resource Broker, the EDG middleware which matches jobs with resources.

Reconstruction: Processing of raw data from either real or simulated events to reconstruct the underlying trajectories and energies of particles.

R-GMA: Relational Grid Monitoring Architecture, an implementation of the GMA developed within EDG which uses a relational data model.

Replica: A physical instance of a file.

Replica Catalogue: A Catalogue which stores the location of Replicas.

RLS: Replica Location System, a model for distributed Replica Catalogues implemented by both EDG and Globus.

RM: Replica Manager, the EDG DMS.

RPM: RedHat Package Manager, a file packaging tool developed by RedHat. Also the format of packages produced by the tool.

Sandbox: A set of files transferred to a WN with a job (input sandbox) or retrieved from the WN when the job finishes (output sandbox).

SE: Storage Element, a Grid-enabled storage system.

SLAC: An HEP laboratory in California.

SRB: Storage Resource Broker, a non-EDG DMS.

SRM: Storage Resource Manager, a Grid-enabled protocol for the management of an SE.

SURL: Site URL, a URL which refers to a Replica stored in an SE.

Tier 1: The major computer centres providing resources to LCG.

Tier 2: Local LCG computer centres, generally providing fewer resources and lower levels of support than Tier 1 centres.

Trigger: A hardware or software algorithm to select interesting events for permanent storage.

TURL: Transfer URL, a URL which enables a Replica to be read or written using a specific transfer protocol.

UI: User Interface, a machine with client software installed from which a user can submit jobs, manage data and query the Information System.

Virtual Data: Data generated on demand using a known algorithm.

VO: Virtual Organisation. In HEP these are expected to correspond to the experimental collaborations, typically with hundreds or thousands of members.

VOMS: Virtual Organisation Management System, a set of middleware to manage VO membership and authorisation information.

WMS: Workload Management System, the middleware which provides high-level management of computing jobs.

WN: Worker Node, a machine on which a job can run, usually via a batch system.