ISTITUTO NAZIONALE DI FISICA NUCLEARE

Progetto INFN–GRID

INFN/TC-02/16
21 Giugno 2002

INFN GRID
DATAGRID PROTOTYPE 1

F.Donno, L.Gaido, A.Ghiselli, M.Mazzucato, F.Prelz, M.Sgaravatto

INFN

Introduction

Computing and networking technology evolution leads to more powerful computers and to high speed networks as low-cost commodity components. In parallel there is a growth of large-scale computing needs with high number of users widely distributed. This scenario is changing the way to think about-and use-computing resources. One of the most interesting and challenging approaches is the "computational grid" concept. DataGrid (www.eu-datagrid.org) is a European project (EDG) aiming to build a computational grid prototype on the basis of requirements coming from users in the application fields of the High Energy Physics[1], Earth Observations (EO) [2], and Bio-Informatics [3]. The first prototype of DataGrid is in place, since December 2001, with the functionalities foreseen in the release 1 of the project. The purpose of this document is to describe the most important characteristics of the grid proposal, the services of the first prototype and some of the problems emerged from the experience of the first steps of the project.

PACS.: 89.80

Published by ***SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

DataGrid project description

DataGrid scenario

The term “Grid” refers to both the technologies and the infrastructure that enable coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations[4] such as the ones considered in the DataGrid project. Thousands of physicists from hundreds of Institutes, laboratories and universities worldwide come together to design, create, operate, and analyze the data of an experiment’s detector at CERN, the European high energy physics laboratory. During the analysis phase, they pool their computing, storage, and networking resources to create a “Data Grid” capable of analyzing petabytes of data. This is an example of the operation of what is described as a Virtual Organization (VO) in the DataGrid scenario. The EO (Earth Observation) collaborations and the international projects of the Bio-Informatics communities highlight similar characteristics.

The dynamic nature of sharing relationships means that we require mechanisms for discovering and characterizing the nature of the relationships that exist at a particular point in time[4]. For example, a new participant joining a VO must be able to determine what resources (s)he is able to access, the “quality” of these resources, and the policies that govern access. This brings the need of specific VO services to specify users identifiers, grid access policies, file catalogues etc.

One-time login based on Identity Certificates

On DataGrid, users of the Grid are not granted “accounts” in the usual sense of the word. That is they do not have a login name and password via which they can log in to Grid computers nodes (or Computer Elements CE). Rather, users own an X.509 Identity Certificate issued by a “Certification Authority”(CA) which is trusted by the EU DataGrid organization [5].

DataGrid security is based on Globus Security Infrastructure (GSI), which implements the standard (RFC 2078/2743) Generic Security Service Application Program Interface (GSS-API). This implementation is based on X.509 certificates and is developed on top of the OpenSSL (www.openssl.org) library.

The GSS-API requires the ability to pass user authentication information to a remote site so that further authenticated connections can be established (*one-time login*). The entity that is empowered to act as the user at the remote site is called “proxy”.

The fresh “proxy” certificate has a short (default value of 1 day) expiration time and is signed by the user. The user certificate is signed by a Certification Authority (CA) that is trusted at the remote end. The remote end (usually at some service provider’s site) is able to verify the proxy certificate by descending the certificate signature chain, and thus authenticate the certificate signer. The signer’s identity is established by trusting the CA, that in DataGrid belongs to the trusted CAs list (http://marianne.in2p3.fr/datagrid/ca/) .

The last remaining step is *user authorization*: the requesting user is granted access and mapped to the appropriate resource provider identifiers by checking the proxy (or user) certificate subject (X.500 Distinguished Name, or DN) and looking it up in a list (the so-called *gridmap file*) that is maintained at the remote site. This list typically links a DN to a local

resource username, so that the requesting user can inherit all the rights of the local user. Many DNs can be linked to the same local user.

The Certification Authorities, participating to the project, defined a set of rules that each CA has to adopt in order to be trusted within a Grid. In each DataGrid computing and storage resource there is the list of the trusted CAs.

VO server and automatic management of gridmap files

The user-certificate serves to "prove" a user identity (authentication); the authorization to perform the requested task must then be specifically granted (authorization). This is done on the basis of an appropriate database, listing the users of each VO. The present implementation of this database (VO server and in the future a more sophisticated Community Authorization Server) is based on LDAP (www.openldap.org). Each experiment has a VO Group administrator which manages user authentication information (user certificate subject extracted from the corresponding Authentication authority (CA)) to be used for authentication and authorization in the Grid infrastructure. This information is then used to build the users database (gridmap file) on computational resources (Computing Elements, CEs) periodically. This is done using tools developed by the project (http://cvs.infn.it/cgi-bin/cvsweb.cgi/Auth/VO/sbin). The server provides only the access policy, while the final authentication is done by the CE gatekeeper.

Replica Catalogues

In distributed systems, data replication is a well-known and accepted technique for optimizing data access and providing fault tolerance. This is achieved by storing multiple copies of data at several locations. The topology and the latency of the network have an important influence on the replication strategy to be used. In DataGrid, database files or simply plain files have to be replicated to several sites. File transfer mechanisms like GridFTP provide secure and efficient file replication. However, replica management not only deals with data transfer but also with meta-data management like replica catalogues and consistency management for file and metadata updates. In the Grid community, file replica catalogues are currently under research and the Globus replica catalog is intended as a fundamental building block in Data Grid systems. It addresses the common need to keep track of multiple physical copies of a single logical file by maintaining a mapping from logical file names to physical locations (www.globus.org/datagrid/replica-catalog.html). Presently, each VO has a Replica Catalog that keeps a list of all the available files. It provides a centralized service with the advantage of the absence of synchronization problems because all files are kept in one catalog that is easy to administer. There are some disadvantages: it is not scalable, there is a potential bottleneck due to WAN latency and high traffic, it is a single point of failure, and remote organizations depend on central organization for administration. In collaboration with the Globus team a new design & implementation for a distributed replica catalog system is under development and the results will be available for the future releases of DataGrid [6].

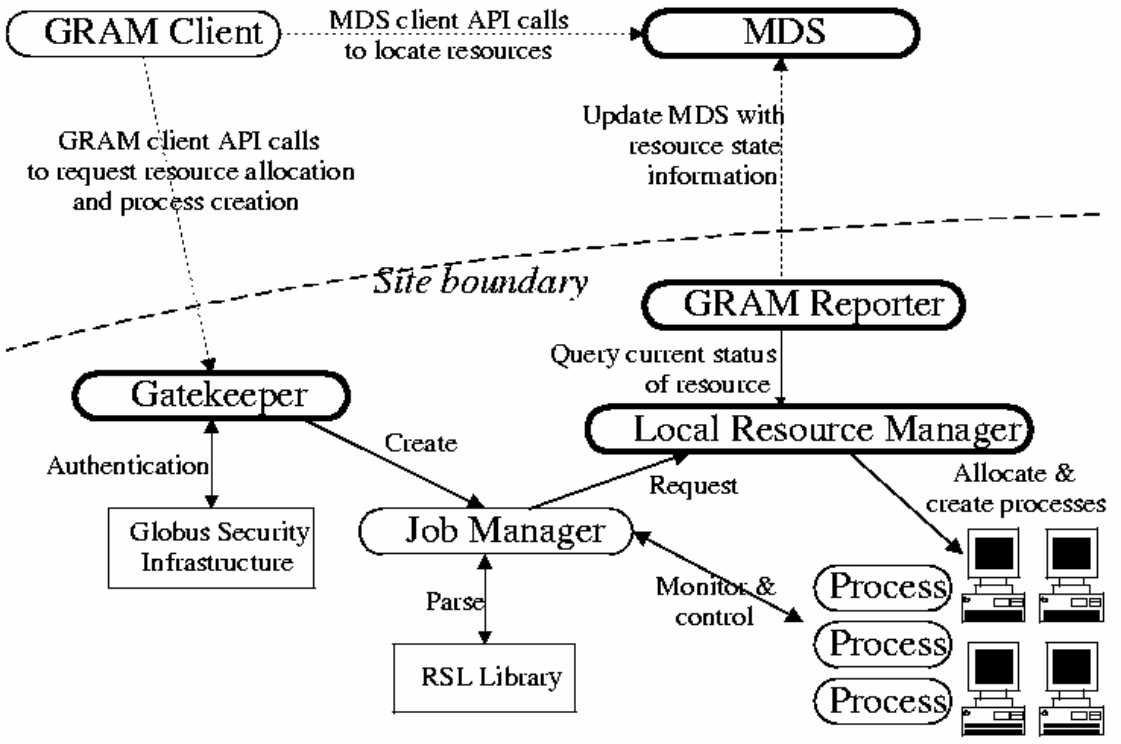
The DataGrid Resources and the Grid Information Service

The Grid resources consist mainly of large computing farms, made up of commodity PCs, and large disk storages, spread all over Europe. Their connections to the Grid is through the National Research Networks (NRNs) interconnected via GEANT.

Computing farms are referred to as Computing Elements (CE) and disk storages as Storage Elements (SE). SE identifies any disk storage with the ability to provide direct file access and transfer via FTP, NFS or other protocols using the grid security mechanisms.

The access to the grid resources is based on the Globus Resource Access Management (GRAM) service [7].

The architecture of the GRAM service is shown in this figure.



The Globus Resource Allocation Manager (GRAM) is responsible for operating a set of resources under the same site-specific allocation policy; this is often done by a local resource management system (such as LSF, PBS, Condor). Therefore a specific GRAM does not correspond to a single host, but rather represents a service; for this reason it can provide access to the nodes of a parallel computer, to a cluster of PCs, to a Condor pool. In the Globus architecture the GRAM service is the standard interface to "local" resources: grid tools and applications can express resource allocation and management requests in terms of a standard interface, while individual sites are not forced to choose a specific resource management tool.

In particular GRAM is responsible for:

- Processing RSL (Resource Specification Language) specifications representing resource requests, by either creating the process(es) that satisfy the request, or by denying that request;
- Enabling remote job monitoring and management;

- Periodically updating the GIS (Grid Information Service) information service with information about the current status and characteristics of the resources that it manages.

The GRAM reporter has been modified by DataGrid, as well as the Grid Information Service schema, in order to provide the GIS service with the information required by the grid scheduler, as described below.

Grid Information Service

The Information Service (IS) plays a fundamental role in the DataGrid environment, since resource discovery and decision making is based upon the information service infrastructure. Basically an information service is needed to collect and organize, in a coherent manner, information about grid resources and status and make them available to the consumer entities.

Hierarchical directory services are widely used for this kind of applications, due to their well defined APIs and protocols. Anyway, the main issue with using the existing directory services is that they are not designed to store dynamic information such as the status of computing (or networking) resources. For this reason other mechanism are under test within DataGrid [8].

DataGrid Release 1 is based on the Globus *Grid Information Service,* which provides an infrastructure for storing and managing static and dynamic information about the status and components of computational grids.

The Globus Grid Information Service is based on LDAP directory services. As pointed out before, LDAP may not be the best choice for dynamic data storage, but provides a number of useful features:

- A well defined data model and a standard and consolidated way to describe data.
- A standardized API (Application Programming Interface) to access data on the directory servers.
- A distributed topological model that allows data distribution and delegation of access policies among institutions.

A good flexibility in extending the set of data returned by each GRIS (Grid Resource Information Service) in the GIS architecture is a definite requirement for a production grid system. The GRIS currently uses *trigger* programs to fetch data from the system. Such data is then cached for a configurable amount of time. The output of the trigger programs must be in LDIF format and must respect the *DataGrid schema*. There are on-going efforts to make the entire information modeling schema evolve.

Resources and services schema for a grid environment

The schema represents what makes data valuable to Grid tools and applications. DataGrid defined and implemented its own CE and SE schemas, since Globus has defined a CE schema that is not suitable for the DataGrid services and the SE was not yet included in the GIS [9]. The DataGrid schema describes the CE as a *Queue*, since it is the most suitable data structure to model a cluster of PCs locally managed by schedulers like PBS, LSF or Condor, while Globus identifies a CE as a single host.

These schemas (of Globus and DataGrid) can be easily modified, and the process of making the schema modifications consistent among various GIS's infrastructures is ongoing in the interest of various interoperability initiatives between the EU and the US. The future common schema is going to describe computing resources in an homogeneous way and will allow an easier application development between EU and US grids.

The definition and standardization of the information about computing resources is largely work in progress and the upcoming grid development efforts will have to make sure to feed back their proposals for extensions to the information schema into the standardization process, for instance within the Global Grid Forum (www-unix.mcs.anl.gov/gridforum/gis).

Grid Scheduling and Workload Management Service

The most important focus of the project is the workload management of applications dealing with large amount of data and the management of clusters with high number of machines. The ambitious aims of the project (resource management, scheduling, co-allocation, advance reservation, accounting, meta-data, high performance networking and network awareness, etc.) can be successful if an effective collaboration with the existing computer science projects and the integration of the various existing tools (Globus, Condor, etc.) will be pursued.

Allocating and scheduling computing resources in a worldwide computational grid is still an open and challenging problem, despite the effort of various computer science research teams. The purpose of the Workload Management effort in the DataGrid project is to suitably integrate existing tools, extend the research and provide the missing software, so that easy access to grid scheduling and submission services for DataGrid application users and developers is provided .

Scheduling on the Grid: which model?

The following considerations emerge from an analysis of the current state of the art[10]. The scheduler is one of the most critical components of the resource management systems, since it has the responsibility of assigning resources to jobs in such a way that the application requirements are met, and of ensuring that the resource usage limits granted to the user are not exceeded. Although scheduling is a traditional area of computer science research, the particular characteristics of the DataGrid project, and of the computational grids in general, make traditional schedulers inappropriate.

As a matter of fact, while in traditional computing systems all the resources and jobs in the system are under the direct control of the scheduler, Grid resources are geographically distributed, heterogeneous in nature, owned by different individuals or organizations with their own scheduling policies, have different access cost models with dynamically varying loads and availability conditions. The lack of centralized ownership and control, together with the presence of many competing users submitting jobs that can potentially be very different from each other, make the scheduling task much harder than for traditional computing systems.

All the existing Grid schedulers can be classified according to three factors, namely their organization (that may be centralized, hierarchical, or distributed), their scheduling policy (that may optimize either system or application performance), and the state estimation technique they use to construct predictive models of application performance.

The ***centralized organization,*** as it is in Condor, is under the control of a single entity, that receives and processes all the allocation requests coming from the Grid users. However, while this approach has the advantage of providing the scheduler with a global system-wide view of the status of submitted jobs and available resources, so that optimal scheduling decisions are possible, it is poorly scalable and tolerant to failures, a centralized scheduler represents a performance bottleneck and a single point of failure.

In a ***distributed organization***, like in AppLeS [11], Ninf [12], and NetSolve [13], the scheduling decisions are delegated to individual applications and resources. In particular, each application is free to choose (according to suitable policies) the set of resources that better fit its needs, and each resource is free to decide the schedule of the applications submitted to it. In this approach there are no bottlenecks and single points of failure but, being the scheduling decisions based on local knowledge only, resource allocation is in general sub-optimal.

Finally in the ***hierarchical*** approach, as adopted in Darwin [14] and Nimrod/G [15], the scheduling responsibilities are distributed across a hierarchy of schedulers. Schedulers belonging to the higher levels of the hierarchy make scheduling decisions concerning larger sets of resources (e.g., the resources in a given continent), while lower-level schedulers are responsible for smaller resource ensembles (e.g., the resources in a given state). Finally, at the bottom level of the hierarchy are the schedulers that schedule individual resources. The hierarchical approach tries to overcome the drawbacks of the centralized and the distributed approach, while at the same time keeping their advantages.

The other important feature of a grid scheduler is the adopted ***scheduling policy***. The schedulers of Condor and Darwin adopt a ***system-oriented*** policy, aimed at optimizing system performance metrics such as system throughput or resource utilization. However the need of dealing with resource co-allocation and advance reservation requires the development of new system-oriented scheduling policies. At the other end of the spectrum, we find systems like AppLes, NetSolve, Nimrod/G, and Ninf, that adopt ***application-oriented*** scheduling policies. Actually in DataGrid there is also a complementary need of scheduling techniques able to maximize ***user performance***. That is, the machines used to execute a given application should be chosen in such a way that its performance is maximized, possibly disregarding the overall system performance.

Moreover in order to obtain satisfactory performance, a scheduler must employ ***predictive models*** to evaluate the performance of the application or of the system, and use this information to determine the allocation that results in best performance. Condor, Darwin, Nimrod/G, and Ninf adopt ***non-predictive*** schedulers. However, by assuming that the current resource status will not change during the execution of applications may result in performance much worse than expected because of the possible presence of contention effects on the resources chosen for the execution of an application. AppLeS, NetSolve address this problem by adopting ***predictive*** schedulers, however, while predictive techniques have the potential of

ensuring better application performance, they usually require a higher computational cost than their non-predictive counterparts.

DataGrid Workload management model

By looking at the scheduling needs of typical users in the scientific communities that take part in the DataGrid project, we observe that a suitable grid scheduler should exhibit several properties not found in any of the currently available schedulers, and in particular:

Distributed organization. Given that several user communities (also called *virtual organizations*) have to co-exist in DataGrid, it is reasonable to assume that each of them will want to use a scheduler that better fits its particular needs (***community scheduler***). However, when a number of independent schedulers are operating simultaneously, a lack of coordination among their actions may result in conflicting and performance-hampering decisions. The need of coordination among these peer, independent schedulers naturally calls for a distributed organization.

- ***Predictive state estimation***, in order to deliver adequate performance even in face of dynamic variation of the resource status.
- ***Ability to interact with the resource information system***. At the moment, all the existing schedulers require that the user specifies the list of the machines that (s)he has permission to use. However, a fully functional grid scheduler should be able to autonomously find this information by interacting with the grid-wide information service.
- ***Ability to optimize both system and application performance***, depending on the needs of DataGrid users. As a matter of fact, DataGrid users needing high-throughput for batches of independent jobs (such as the HEP community) have to co-exist with users requiring low response times for individual applications (e.g. the bio-medical community). In this case, neither a system-oriented nor an application-oriented scheduling policy would be sufficient.
- ***Submission reliability***: Grids are characterized by an extreme resource volatility, that is the set of available resource may dynamically change during the lifetime of an application. The scheduler should be able to resubmit, without requiring the user intervention, an application whose execution cannot continue as consequence of the failure or unavailability of the machine(s) on which it is running.
- ***Allocation fairness***. In a realistic system different users will have different priorities that determine the amount of Grid resources allocated to their applications.

The workload management system (WMS) has been designed [16] having in mind the above properties for a Grid scheduler. Several interactive WMS components provide the “grid scheduling” service: the User Interface, The Resource Broker, the Job Submission Service, and the Logging and Bookkeeping service. The WMS solution is designed to support two basic designs of resource broker: a community scheduler running on a highly-available server and a personal scheduler running on a user’s personal machine.

The Resource Broker (RB) is the core component of the WMS. Its main task is to find a computing element that best matches the requirements and preferences of a submitted job, considering also the current distribution of load on the grid. Once a suitable computing element is found, the job is passed to the *job submission service* for the actual submission.

Additionally the resource broker allows cancelling a job and retrieving the output once a job has been completed.

These tasks include interacting with the Replica Catalog (RC) to resolve Logical data set names as well as to find a preliminary set of sites where the required data are stored, performing job submission and cancellation by interacting with the Job Submission Service (JSS), listing the more likely resources to execute a job at, and retrieving job outputs on behalf of the clients.

The logging and bookkeeping service is responsible to store and manage logging and bookkeeping information generated by the various components of the WMS. It collects information about the scheduling system and about active jobs.

A user can submit jobs and retrieve their output through the User Interface. The description of a job is expressed in the Job Description Language (JDL), which is based on the *classified advertisement* scheme developed by the Condor project. This choice has been made because:

It is a semi-structured data model: no specific schema is required

- Symmetry: all entities in the grid, in particular applications and computing resources can be expressible in the same language.
- Simplicity for both syntax and semantics.

Data Management in DataGrid

The data management architecture [17] is focused on file replication services and the main objectives include optimized data access, caching, file replication and file migration. The most important tasks are:

- Management of a universal namespace for files (using replica catalogues)
- Secure and efficient data transfer between sites
- Synchronization of remote copies
- (Optimized) wide-area data access/caching
- Management of meta-data like indices and file meta-data
- Interface to mass storage systems

Data Management in DataGrid has to deal with heterogeneity of storage systems and thus Storage Elements (SE). The interface to SE has to be unique regardless of the underlying storage technology. SE is a basic storage resource in DataGrid and also define the smallest granularity for a compound storage system. A Storage Element can either be a large disk pool or a Mass Storage System (MSS) having its own internal disk pool. The current storage system implementations include MSS like HPSS, Castor as well as distributed file systems like AFS [18] or NFS (www.nfsv4.org). All these implementations can potentially co-exist and the Storage Element subsystem has to hide the data access mechanisms (and their complexity) specific to each storage system from middleware upper layers.

The building blocks of the DM architecture are: the Replica Manager, the Replica Catalog, the File Copier and the Consistency Service.

The *Replica Manager* manages file transfers (replication) by using the File Copier service and the replica catalogue information through the replica catalog service. The first

implementation is based on GDMP (Grid Data Mirroring Package) [19] which is a file replication tool that implements most of the replica manager functionalities. The main tasks of the replica manager are to securely and efficiently copy files between two Storage Elements and update the replica catalogue when the copy process has successfully terminated.

The access to the replica files shall be optimized by using some performance information such as the current network throughput and latency and the load on the Storage Elements. The *Replica Optimizer's* duty is to select the "best" replicas.

The *File Copier* (also called Data Mover) is an efficient and secure file transfer service that has to be available on each Storage Element. Initially this service will be based on GridFTP protocol [20].

The Consistency Service has to guarantee the synchronization of the file replicas when an update occurs. This service is provided on top of the replica manager.

DataGrid Release1 description

The first prototype contains the basic functionalities that provide the users with an environment allowing to define and submit jobs to the grid system. It also allows to find and use the best grid resources, according to a set of requirements about the job specified by the user and to the characteristics and status of the available resources (CPU power, memory size, CPU load etc.). Bookkeeping and logging information are stored in an appropriate server and are available to the user. The Grid services actually used in the DataGrid Release1 are:

UI User Interface	Lightweight component for accessing to the workload management system.
RB Resource Broker	The core component of the workload management system, able to find a resource matching the user's requirements (including data location).
LB Logging and Bookkeeping	Repository for events occurring in the lifespan of a job.
JSS Job Submission Services	It is the result of the integration of Condor-G and Globus services to achieve reliable submission of jobs via the Globus GRAM protocol.
II Information Index	Caching information index (based on Globus MDS-2) directly connected to the RB, to achieve control of the cache times and to prevent blocking failures when accessing the information space.
Replica Manager /GDMP	To consistently copy (replicate) files from one Storage Element to another and register replicas. A manual replication tool (GDMP) has been released for prototype1
Replica Catalog	It stores information about the physical files on all the Grid Storage Elements. A centralized replica catalog has been chosen for prototype1
BrokerInfo	It allows access to information obtained and used in the brokering process by a running job.
MDS (GRIS and Info.Providers)	Grid Information System used by the Resource Broker. The first implementation is based on the Globus MDS system, with resource schema and information providers defined and implemented by DataGrid.
Authen. and Author. services	VO directory configuration and tools to periodically generate authorization lists
Automatic Installation and Configuration management	Large Scale Linux Configuration (LCFG) tool for very large computing fabrics (CE)

The basic building blocks of the DataGrid resources are the Computing Elements and the Storage Elements.

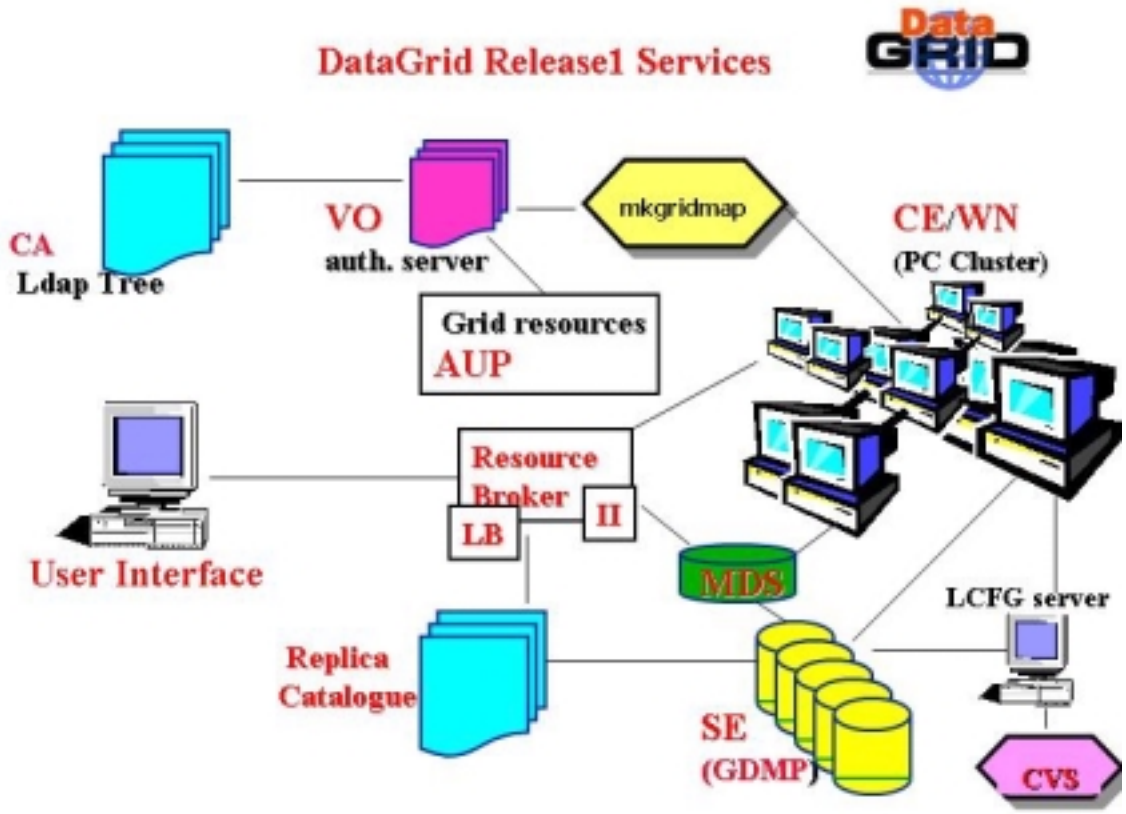
The first UI version allows the user interaction with the system by means of a classad-based Job Description Language (JDL) and a command-driven user interface providing commands to perform a certain set of basic operations. The main operations made possible by the UI are:

- Submit of a job for execution on a remote Computing Element, including:
  - automatic resource discovery and selection
  - staging of the application and data (input sandbox)
- Selection of a list of suitable resources for a specific job
- Cancellation of one or more submitted jobs
- Retrieval of the output file(s) produced by a completed job (output sandbox)
- Retrieval and display of bookkeeping information about submitted jobs
- Retrieval and display of logging information about jobs.

As already mentioned, the user has to own a valid X.509 certificate issued by a DataGrid trusted CA. Every time a command is executed, the system checks for the existence and the expiration date of a user proxy certificate: if the proxy certificate does not exist or it is expired a new one is automatically created by the UI using the GSI services.

In order to get access to the Testbed1 a user has first to sign the EDG Usage Guidelines (https://marianne.in2p3.fr/datagrid/documentation/EDG-Usage-Guidelines.html).

The following picture represents the relationships between the Release1 services.



Middleware integration and release1

The EDG services of the Release1 are provided by several software components implemented by different EDG Work Packages developed on top of Grid basic services provided by Globus and Condor.

All these software components are integrated in an adequate way to constitute specific machine types called *Grid Elements*, the basic building blocks of the testbed.

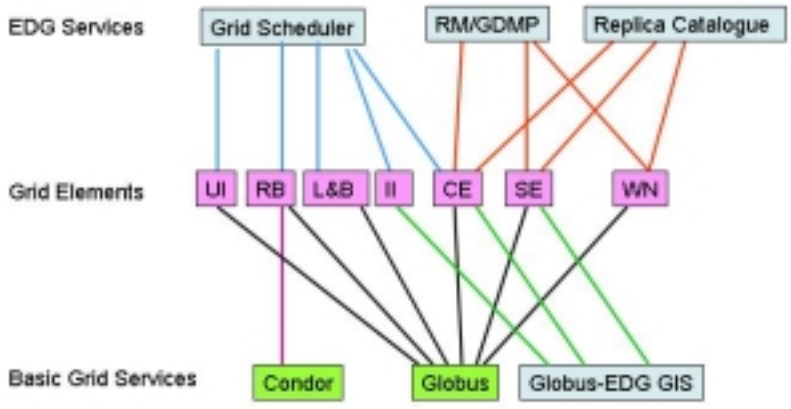
These are:

- User Interface, the gateway for job submission to the grid testbed.
- Computing Element, which are the gateway to the grid farm nodes. A job dispatched to a Computing Element is then passed to a local scheduler for its execution on the controlled farm nodes.
- Worker Node, belonging to the computing farm attached to the various Computing Elements.
- Storage Elements, providing services to store and locate data, to replicate them over the grid and to publish information about data availability.
- Resource Broker, responsible to match job requirements and grid resources and to schedule the job submission to the "best matching" computing element.
- Logging & Bookkeeping (LB) Element is responsible for keeping track of the history of the job execution and provides monitoring and job management services.

Other structural services that are essential in the testbed are:

- Information Indices are sitting by the Resource Brokers to provide a cached and decoupled view of the GIS information about the resource locations and the status of the testbed.
- Replica Catalogues store the location of data in the testbed.

The figure below shows how the software coming from Work Packages and Globus, Condor subcomponents merge together in the configuration of the grid elements described above.



The Testbed1 Software Release

In order to guarantee consistency and correctness of operations of the testbed, it was necessary to define an EDG release of the software and specific installation instructions for the deployment on each sites. The strategy of enforcing a well defined software environment to be deployed by all sites has been adopted. In particular a specific version of the Operating System and related updates, together with external package dependencies, the Globus release to be deployed, configuration and user environment procedures have been identified. The nature of the various "Grid elements" allow for a more flexible definition of the testbed topology and calls for well-defined distribution and installation procedures to guarantee uniformity and consistency of the testbed over wide area.

This way of operating allows to easily extend the testbed including new sites, to push new configurations and solutions to all sites, to reproduce and track down problems.

The definition of an EDG Release

The hardest task of the Integration Team [5], a working group in charge of integrating the different pieces of software, has been to collect all the EDG software packages and Globus, to study the functionality and interdependencies, the requirements for a correct operation of the testbed and come up with a topology and precise installation and configuration instructions for deployment. In order to achieve these goals it has been necessary to construct a node-centric view of the testbed deployment specifying the profile of each node type (*grid elements*).

Because of the requirements coming with the installation and distribution tools provided for the farms, the EDG release has been packaged via Linux RPMs (Red Hat Package Manager). In particular the Globus main subcomponents were packaged in separate RPMs that could be installed and configured independently.

After identifying the main grid node types (*grid elements*), the EDG Integration Team has defined, via an iterative process, the set of RPMs that provide the functionality of each of the above elements. A list of software RPMs and configuration RPMs defines a *grid element* in an EDG release. During this work, it was necessary to specify detailed configuration instructions and requirements that allow these elements to interoperate.

Once the definition of the profile for a grid element has been optimized, a small test suite has been used to verify that all the functionalities required were present and working.

For each of the grid elements an LCFG template provides [21]:

1. the RPM lists of the packages required for a specific element
2. a typical LCFG configuration that needs to be customized for a specific testbed site.
3. a specific set of instructions about configuring the grid element.

All the above constitutes the content of the edg-release package available in the CVS repository of the DataGrid Release [22].

Before tagging a release, the installation and distribution procedures via LCFG are tested on a small development testbed.

The Testbed1: deployment of Datagrid Release1

The first prototype of the EDG (European DataGrid) grid infrastructure (Testbed1) has been deployed in December 2001 using the official - tagged- EDG software Release 1, based on Globus 2 beta 21. The Testbed1 has been used for the validation of EDG software, performed by the Validation Team.

The Testbed1 was initially made up by a limited amount of grid elements in 5 European countries (CERN, FR, UK, IT, NL), but it is currently being extended to about 30 sites all over Europe including also some other countries such as the Czech Republic, ES, PT, DE and the Nordic Countries.

In the initial Testbed1 layout the "common" grid elements (User Interfaces, Computing Elements, Worker Nodes and Storage Elements) have been installed and configured at each site while the grid elements devoted to the central grid services (Resource Brokers, Information Indexes and Logging and Bookkeeping servers) have been set up at CERN and INFN-CNAF (IT). Some other dedicated servers (the Virtual Organization LDAP servers and the VO Replica Catalogues) have been hosted at NIKHEF (NL) and INFN-CNAF.

The central repository hosts the EDG software (in rpm format) including the external software it depends on. The local site manager gets the software to be installed from the central repository. The automatic installation and configuration tool (LCFG) has been used to install the grid element at each site. LCFG is very useful also when the grid elements have to be upgraded; the site manager has only to modify the node's LCFG profile and to issue the LCFG upgrade command.

On Testbed1 the grid elements are shared by the different Virtual Organizations (the High Energy Physics, Earth Observation and Biomedical communities). This setup allows a great flexibility for Testbed1 usage. By means of the Virtual Organization dedicated servers (X509 Certificate LDAP servers and Replica Catalogues), authorization and data management issues are addressed, while, at the run level, it is possible to find out on which CE the VO specific software has been installed through the usage of simple environment variables in the JDL script.

In addition, the DataGrid Testbed1 flexibility allows for the definition of different grid topologies (e.g. different grid domains with dedicated services for the different VOs) according to the needs.

Validation of Release1

It is essential for the success of the project that the developed middleware satisfies the user's initial requirements. For this reason, a software validation phase performed by the user communities using real applications in a large scale environment is crucial within the project.

After a successful validation phase the users will have to develop their applications trying to profit by the EDG software.

The validation activity will be progressively done during the whole project lifetime, in order to test each middleware release.

Short term use cases have been used at the beginning and real applications are going to be used later on [23].

Conclusions

The first DataGrid prototype is in place in Europe as result of the collaboration of all the actors of a distributed computing environment: resource owner, middleware developers, scientific application programmers and scientific application users. The testing phase demonstrated the power of the grid whose basic functionalities allow to select the most appropriate CE-SE pair just specifying job characteristic and related input/output data in a high level language. The preliminary Release1 validation phase, completed at the end of January 2002, provided a first evaluation of the services that will be taken in account for the next releases, foreseen every 3 months. The EU DataGrid project successfully passed the first year review (On March, 1st 2002), performed by external experts appointed by the European Union. During the review, a demo over the testbed1 has been successfully performed using the experiment applications.

The major project release, foreseen for September 2002, will introduce new important services like support for dependent, parallel, and partitionable jobs, resource co-allocation, advance reservation and accounting, as well as more efficient information and monitoring services.

Acknowledgement

The author gratefully acknowledges all the members of the EDG collaboration. Special thanks are due to F.Donno, L.Gaido, F.Prelz, M.Sgaravatto and H.Stockinger for their contributions to the writing of the paper.

References

(all the document referenced with 'DataGrid-…..' can be found at:
http://eu-datagrid.web.cern.ch/eu-datagrid/deliverables/default.htm)

1. DataGrid User Requirements and Specifications for the DataGrid Project (WP8), DataGrid-09-D9.1-0101-1_2
2. Requirements Specification. EO Requirements for Grid, DataGrid-09-D9.1-0101-1-2
3. Requirements for a Grid-aware Biology Applications DataGrid-10-D10.1-0102-3-8
4. I.Foster, C.Kesselman, S.Tuecke The Anatomy of the Grid: Enabling Scalable Virtual Organization. International Jurnal of High Performance Computing Applications.2001
5. Evaluation of Testbed Operation, DataGrid-06-D6.4-0109-1-0
6. Data Management (WP2) Architecture Report DataGrid-02-D2.2-0103-1-2
7. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A resource management architecture for metacomputing systems. In *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.
8. Information and Monitoring (WP3) Architecture Report DataGrid-03-D3.2-0101-1-0
9. M.Sgaravatto ,WP1 Inputs to the DataGrid Grid Information Service Schema Specification, Sept. 2001, http://grid.infn.it/workload-grid/documents.htm
10. WP1 report on current technology, DataGrid-01-TED-0102-1-0
11. F. Berman and R. Wolski. The AppLeS Project: A Status Report. In *Proc. of the 8th NEC Research Symposium*, Berlin, Germany, May 1997.
12. H. Nakada, M. Sato, and S. Sekiguchi. Design and Implementation of Ninf: towards a Global Computing Infrastructure. *Future Generation Computing Systems*, October 1999. Special Issue on Metacomputing.
13. H. Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. *Intl. Journal of Supercomputing Applications and High Performance Computing*, 11(3), 1997.
14. P. Chandra, A. Fisher, and C. Kosak et~al. Darwin: Customizable Resource Management for Value-Added Network Services. In *Proc. of the 6th Int. Conf. on Network Protocols*}. IEEE,1988
15. R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. In *Proc. of Int. Conf. on High Performance Computing in Asia-Pacific Region*, Beijing, China, 2000. IEEE-CS Press.
16. Definition of Architecture, technical Plan and Evaluation Criteria for Scheduling, Resource Management, Security and Job description DataGrid-01-D1.2-0112-0-3
17. Data Management (WP2) Architecture Report DataGrid-02-D2.2-0103-1-2
18. ASF home page: http://oss.software.ibm.com/developerworks/opensource/afs/
19. A. Samar, H. Stockinger. Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication, *IASTED International Conference on Applied Informatics (AI2001)*, Innsbruck, Austria, February 19-22, 2001. http://cmsdoc.cern.ch/cms/grid/
20. www.globus.org/datagrid/gridftp.html
21. www.lnl.infn.it/datagrid/wp4-install/testbed-report_2_v3/index.html
22. EDG software repository: http://marianne.in2p3.fr/datagrid/testbed1/repositories/index.html
23. Testbed1 Assessment by HEP Applications, DataGrid-08-D8.2-0111-0-3

Glossary

AFS	Andrew File System
API	Application Programming Interface
ATF	DataGrid Architecture Task Force
CA	Certification Authority
CAS	Community Authorisation Service
CE	Computing Element; a Grid-enabled computing resource.
CERN	Centre Europeen pour la Recherche Nucléaire
CERT	X.509 Certificate
CNRS	Centre National de la Recherche Scientifique
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CVS	Concurrent Versioning System
DAQ	Data Acquisition
DBMS	Data Base Management System
DHCP	Dynamic Host Configuration Protocol
DMA	Direct Memory Access
EC	European Commission
EDG	European Data Grid; (official project name is “DataGrid”).
EO	Earth Observation
ES	Earth Science
ESD	Event Summary Data; used in HEP: information required for detailed analysis and high-level reconstruction.
ESRD	Earth Science Requirement document
EU DataGrid	European Data Grid Project
FSM	Finite State Machine
FTP	File Transfer Protocol
GDMP	Grid Data Mirroring Package; a WP2 application.
GGF	Global Grid Forum
GIF	Graphics Interchange Format
GIS	Grid Information Service; (e.g. IMS or Globus MDS).
Globus	Project that aims at developing fundamental technologies needed to build computational grids.
GMA	Grid Monitoring Architecture; monitoring architecture defined by GGF.
GNU	GNU's Not Unix
GRAM	Grid Resource Allocation Management
GriFIS	Grid Fabric Information Service
GS	Grid Scheduler; service responsible for selecting which Grid resources to use for a given job.
GSI	Grid Security Infrastructure (Globus Security mechanism).
GUI	Graphical User Interface
HEP	High Energy Physics
HSM	Hierarchical Storage Manager. HSM software allows infrequently accessed data to be migrated to less expensive offline storage automatically.
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HW	Hardware

I/O	Input/Output
IDL	Interactive Data Language
IMS	Information and Monitoring System; catalogue and distribute static and dynamic data about the Grid.
IST	Information Society Technologies
JDL	Job Description Language; to describe Grid jobs.
JSS	Job Submission Service
LB	Logging and Bookkeeping
LCAS	Local Centre Authorisation Service
LCG	LHC Computing Grid
LCMAPS	Local Credential MAPping Service
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LFN	Logical File Name; A globally unique name to identify a specific file which is mapped by the RC onto one or more PFNs.
LHC	Large Hadron Collider
LRMS	Local Resource Management System; controls resources within a CE e.g. PBS or LSF.
LSF	Load Sharing Facility
MDS	Globus Meta-computing Directory Service
MPI	Message Passing Interface
MPI	Standard API for exchanging messages between networked computers. Several implementations of MPI are available on different platforms. MPICH is a freely available, portable implementation of the MPI standard. A Globus-aware MPICH implementation is proposed on Globus home page
MR	Monitoring Repository
MSA	Monitoring Sensor Agent
MSS	Mass Storage System
MUI	Monitoring User Interface
MySQL	Widely distributed SQL open source implementation.
NFS	Network File System
NIS	Network Information System
NMA	Node Management Agent
OS	Operating System
PB	PetaByte
PDS	Payload Data Segment
PFN	Physical File Name; URL of actual physical instance of an LFN.
PKI	Public Key Infrastructure
PM	Project Manager
PMB	Project Management Board
PTB	DataGrid Project Technical Board
QoS	Quality of Service
RB	Resource Broker
RC	Replica Catalog; associates an LFN to one or more PFNs.
RCS	Revision Control System
RD	Research and Development
RDBMS	Relational Database Management System
Replica	A copy of a file that is managed by the Grid middleware.
Replica	The combination of Replica Catalogue, Replica Manager, and all associated

Service	components.
RM	Replica Manager; provides several services related to replicas.
RMS	Resource Management Subsystem
ROI	Region of interest
RPC	Remote Procedure Call
RPM	Red Hat Package Manager
RSL	Globus Resource Specification Language
RUP	Rational Unified Process
SAN	Storage Area Network.
SE	Storage Element; a Grid-enabled storage system.
SI2000	SpecInt 2000 Benchmark
SP	Software Package
SQL	Standard Query Language for relational databases.
SR	Software Repository
SSL	Secure Sockets Layer
SW	Software
TAG	Event Selection Tag; used in HEP: used for fast event selection.
TeV	Tera electron Volt
Testbed 0	DataGrid initial Glogus toolkit deployment
Testbed 1	First release of DataGrid Software - Year 1
Testbed 2	Second release of DataGrid Software - Year 2
Testbed 3	Third release of DataGrid Software - Year 3
TFN	Transport File Name; URL to access a given file on a SE.
TFTP	Trivial File Transfer Protocol
UI	User Interface
URL	Uniform Resource Locator
VO	Virtual Organization; A set of individuals defined by certain sharing rules - e.g. members of a collaboration.
VPN	Virtual Private Network
WM	Workload Management
WMS	Workload Management System
WP	Work Package
WP1	Grid Work Scheduling
WP2	Grid Data Management
WP3	Grid Monitoring Services
WP4	Fabric Management
WP5	Mass Storage Management
WP6	Integration Testbed and Demonstrators
WP7	Network Services
WP8	HEP Applications
WP9	Earth Observation Applications
WP10	Biology Applications
WP11	Dissemination
WP12	Project Management
WWW	World Wide Web
XML	eXtensible Markup Language