J Grid Computing
DOI 10.1007/s10723-009-9137-0

EDGeS: Bridging EGEE to BOINC and XtremWeb

Etienne Urbah · Peter Kacsuk · Zoltan Farkas · Gilles Fedak · Gabor Kecskemeti · Oleg Lodygensky · Attila Marosi · Zoltan Balaton · Gabriel Caillat · Gabor Gombas · Adam Kornafeld · Jozsef Kovacs · Haiwu He · Robert Lovas

Received: 7 November 2008 / Accepted: 27 August 2009
© Springer Science + Business Media B.V. 2009

Abstract Desktop Grids, such as XtremWeb and BOINC, and Service Grids, such as EGEE, are two different approaches for science communities to gather computing power from a large number of computing resources. Nevertheless, little work has been done to combine these two Grid technologies in order to establish a seamless and vast Grid resource pool. In this paper we present the EGEE Service Grid, the BOINC and XtremWeb Desktop Grids. Then, we present the EDGeS solution to bridge the EGEE Service Grid with the BOINC and XtremWeb Desktop Grids.

Keywords BOINC · Bridge · Desktop Grid · EDGeS · EGEE · Interface · Interoperation · OGF · Service Grid · XtremWeb · XWHEP

E. Urbah (✉) · O. Lodygensky · G. Caillat
LAL, Université Paris-Sud, IN2P3/CNRS, Bld 200,
91898 Orsay, France
e-mail: urbah@lal.in2p3.fr
URL: http://www.lal.in2p3.fr/spip.php?article105,
http://www.lal.in2p3.fr/spip.php?lang=en

O. Lodygensky
e-mail: lodygens@lal.in2p3.fr

G. Caillat
e-mail: gcaillat@lal.in2p3.fr

P. Kacsuk · Z. Farkas · G. Kecskemeti ·
A. Marosi · Z. Balaton · G. Gombas · A. Kornafeld ·
J. Kovacs · R. Lovas
MTA SZTAKI, LPDS, Kende u. 13–17,
1111 Budapest, Hungary
URL: http://www.lpds.sztaki.hu

P. Kacsuk
e-mail: kacsuk@sztaki.hu

Z. Farkas
e-mail: zfarkas@sztaki.hu

G. Kecskemeti
e-mail: kecskemeti@sztaki.hu

A. Marosi
e-mail: atisu@sztaki.hu

Z. Balaton
e-mail: balaton@sztaki.hu

G. Gombas
e-mail: gombasg@sztaki.hu

A. Kornafeld
e-mail: kadam@sztaki.hu

J. Kovacs
e-mail: smith@sztaki.hu

R. Lovas
e-mail: rlovas@sztaki.hu

G. Fedak · H. He
INRIA Saclay, LRI, Bat 490,
91400 Orsay, France
URL: http://www.lri.fr/projet.associe.php?prj=15

G. Fedak
e-mail: gilles.fedak@inria.fr

H. He
e-mail: haiwu.he@inria.fr

Published online: 16 September 2009

Abbreviations

ACS	Application Contents Service (OGF recommendation)
ARC	Advanced Resource Connector (Grid middleware used by NorduGrid)
BES	Basic Execution Services (OGF recommendation)
BIFI	Institute for Biocomputation and Physics of Complex Systems (University of Zaragoza, Spain)
BOINC	Berkeley Open Infrastructure for Network Computing (Grid middleware)
CA	Certificate Authority
CE	Computing Element (EGEE)
DEISA	Distributed European Infrastructure for Supercomputing Applications
DG	Desktop Grid = loose opportunistic Grid using idle resources
DMI	Data Movement Interface (OGF recommendation)
DOM	Document Object Model (World Wide Web Consortium)
EADM	EDGeS Application Development Methodology
EDGeS	Enabling Desktop Grids for e-Science
EGEE	Enabling Grids for E-sciencE
FTP	File Transfer Protocol
GEMLCA	Grid Execution Management for Legacy Code Applications (University of Westminster)
gLite	Grid middleware used by EGEE
GRAM	Grid Resource Allocation Manager (Globus Toolkit)
INRIA	Institut National de Recherche en Informatique et en Automatique (Saclay, France)
JDL	Job Description Language (EGEE)
JSDL	Job Submission Description Language (OGF recommendation)
LB	Logging and Bookkeeping (EGEE)
LAL	Laboratoire de l'accélérateur Linéaire (Orsay, France)
LCG	LHC Computing Grid
LPDS	Laboratory of Parallel and Distributed Systems (MTA SZTAKI, Budapest, Hungary)
LRI	Laboratoire de Recherche en Informatique (Orsay, France)
OGF	Open Grid Forum
OGSA	Open Grid Services Architecture (OGF)
OGSI	Open Grid Services Infrastructure (OGF)
OMII	Open Middleware Infrastructure Institute
OSG	Open Science Grid (USA)
QoS	Quality of Service
RUS	Resource Usage Service (OGF recommendation)
SETI	Search for Extra-Terrestrial Intelligence
SG	Service Grid = Globally managed Grid of distributed, locally managed computing clusters
SRM	Storage Resource Manager
SZDG	SZTAKI Desktop Grid (Budapest, Hungary)
UI	User Interface machine (EGEE)
UR	Usage Record (OGF recommendation)
UoW	University of Westminster (London, Great Britain)
VDT	Virtual Data Toolkit (Grid middleware used by OSG)
VO	Virtual Organisation
VOMS	VO Management Service
WLCG	Worldwide LHC Computing Grid (=LCG)
WMS	Workload Management System (EGEE)
WSRF	Web Services Resource Framework (OASIS)
WU	Work Unit (BOINC)
XML	eXtensible Markup Language (World Wide Web Consortium)

1 Introduction

Originally, the aim of Grid research was to enable sharing of computing resources, facilitate collaboration, realise the vision that anyone can donate resources to the Grid, and that anyone can claim resources dynamically according to its needs. This fourfold aim has been, however, not fully achieved yet. Currently we can observe two different approaches in the development of Grid systems. In this section, we first introduce the Service Grids (SG) approach, then the Desktop Grids (DG) approach, and finally the bridging of both types of Grids.

1.1 The Service Grids Approach

There is a growing interest among scientific communities to share their distributed computing and storage infrastructures to solve their grand-challenge problems and to further enhance their applications with extended parameter sets and greater complexity. Many scientific communities call Service Grid (SG) such a shared distributed computing and storage infrastructure. Researchers and developers in Service Grids first create a Grid service that can be accessed by a large number of users. A resource can become part of the Grid by installing a predefined software set, or middleware. However, the middleware is usually so complex that it often requires extensive expert effort to install and maintain. It is therefore natural that individuals do not normally offer their resources in this manner, and SGs are generally restricted to larger institutions, where professional system administrators take care of the hardware/middleware/software environment and ensure high-availability of the Grid.

Even though the original aim of enabling anyone to join the Grid with one's resources has not been fulfilled, the largest Grid in the world (EGEE) contains more than 100,000 processors. The security model used by SGs is based on mutual authentication between users and resources which is realised by a public key infrastructure (PKI) using X.509 certificates. Anyone who obtains a valid certificate from a Certificate Authority (CA) can access those Grid resources that trust that CA. This is often simplified by Virtual Organization (VO) or community authorization services that centralizes the management of trust relationships and access rights.

Interoperation between several SGs has already been achieved by the joint work of several organisations, notably the Open Grid Forum (OGF) [1], the Open Middleware Infrastructure Institute (OMII-Europe), the Worldwide LHC Computing Grid (WLCG).

1.2 The Desktop Grids Approach

Desktop Grids (DG), literally Grids made of Desktop Computers, are very popular in the context of "Volunteer Computing" for large scale "Distributed Computing" projects like SETI@home [2] and Folding@home. They are very attractive, as "Internet Computing" platforms, for scientific projects seeking a huge amount of computational power for massive high throughput computing. DG uses computing, network and storage resources of idle desktop PCs distributed over multiple LANs or the Internet. Today, this type of computing platform aggregates one of the largest distributed computing systems, and currently provides scientists with tens of TeraFLOPS from hundreds of thousands of hosts. In DG systems, such as BOINC [3] or XtremWeb [4], anyone can bring resources into the Grid. Installation and maintenance of the client side software is intuitive, requiring no special expertise, thus enabling a large number of donors to contribute into the pool of shared resources. On the downside, only a very limited user community (i.e., target applications) can effectively use DG resources for computation. For instance, the BOINC project features a limited number of applications, and the top 5 projects share more than 50% of the total compute power. Because users are Internet volunteers, there cannot be security model based on trust between users. Because of users anonymity, security solution for DG relies on autonomous mechanisms such as sandbox or result validation to prevent attacks from other users. As a consequence, DG systems are not yet ready to be integrated in a complex Grid infrastructure which requires a high level user right management, authentication, authorization and rights delegation.

1.3 The Bridging of Both Types of Grids

Until now, these two kinds of Grid systems are completely separated and there is no way to use their individual advantageous features in a unified environment. However, with the objective to support new scientific communities that need extremely large numbers of resources they can't find in SG, the solution could be to interconnect these two kinds of Grid systems into an integrated Service Grid–Desktop Grid (SG–DG) infrastructure.

Compared to application portals which allow job submission to different infrastructures, bridges permit seamless interoperation using low middleware layers, and prepare real interoperability. The issue of the (potentially long) time needed by a Grid to process a job is *not* a purely technical issue which could be solved inside the bridge itself, but a issue of QoS and SLA between the Grid user, the first (Service or Desktop) Grid to which the Grid user submits the Job, and any subcontractor (Service or Desktop) Grid to which a Grid sends the job for execution. The owner of a resource in a Desktop Grid does *not* care if a job comes from a Service Grid or from a Desktop Grid server, but he only cares that the Application is trusted, and has some usefulness. Therefore a SG–DG bridge must use an Application Repository for storing the validated and hence trusted applications.

Interoperation between SGs and DGs has already been explored, notably by the Lattice project [5] at University of Maryland (USA), the SZTAKI Desktop Grid [6] (Budapest, Hungary), the Condor project (BOINC backfill) at University of Wisconsin (USA), the Superlink project at Technion (Haifa, Israel), and the Clemson University (South Carolina, USA).

Our research is part of the work conducted by a new European FP7 infrastructure project: EDGeS (Enabling Desktop Grids for e-Science) [7], which aims to build technological bridges to facilitate interoperability between DG and SG.

In this paper, we describe the technical solution offered by the EDGeS project to bridge the EGEE service Grid with the BOINC and XtremWeb desktop Grids, and the relevant OGF recommendations for interoperability.

In the next section, we give a technical presentation of Service Grids and Desktop Grids. In Section 3, we give a presentation of the EDGeS project. In Section 4, we describe related research in bridging the two kinds of Grids. In Section 5, we describe the implementation of the bridges in EDGeS. In Section 6, we present the current operational status of EDGeS. In Section 7, we present the OGF standards relevant for future Grid interoperability.

2 Technical Presentation of Service Grids and Desktop Grids

Technically, service Grids and desktop Grids are very different. In this chapter we show their typical architecture and characteristics.

In order to ease the understanding of figures, we will use following colours throughout this paper: Light blue for Users, Light yellow for EGEE, Pink for BOINC, Light pink for XtremWeb, Gold for EDGeS, Light green for services or components.

2.1 Technical Presentation of Service Grids

A Service Grid (SG) is a *globally managed Grid of distributed, locally managed computing clusters*, offering a guaranteed QoS (Quality of Service). Typically, institutions with their managed clusters can join to SGs if they sign a certain SLA (Service Level Agreement) with the leadership of the SG. Since participants to a SG are most often institutions, an SG is also often called an "Institutional Computing Grid".

Inside service Grids, computing and storage resources are managed by trained staff and are authenticated by X509 certificates. Users are also authenticated by X509 certificates or proxies. Users may belong to Virtual Organisations (VOs) and get X509 proxy extensions from a VOMS server, which can allow them to access data and submit jobs. On the contrary, executables are *not* authenticated. So trust is primarily between sites and VOs.

The general principle of job processing is that users submit their jobs by delegating their X509 proxy to the SG broker (often called meta-scheduler), which *pushes* the jobs to resources that are both suitable and available. Figure 1 below shows the architecture and working mechanism of an EGEE-like SG where 7 basic Grid services are used:

1. The *Information Service* is completely distributed, provides Grid information to all other Grid services, and is not explicitly shown in the figure.
2. The *User Interface* is the machine where the SG client is deployed and from where users can submit their jobs into the SG.
3. The *VOMS Server* is the service storing user authorization information.
4. The *Meta-scheduler (WMS)* accepts the jobs submitted by the users, checks user job requirements and matches them with available Grid resources. Once it finds a matching Grid resource, it pushes the job to the selected Grid site. When the job is finished, it retrieves the Output Sandbox from the Grid site, and makes it available to the user having submitted the job.
5. The *Computing Resource*s are used to process the user jobs providing the necessary computing power.
6. The *Storage Resources* are used to provide storage capacity to user jobs.
7. *Accounting, Logging and Bookkeeping* collect and store job status information, logging and accounting information.

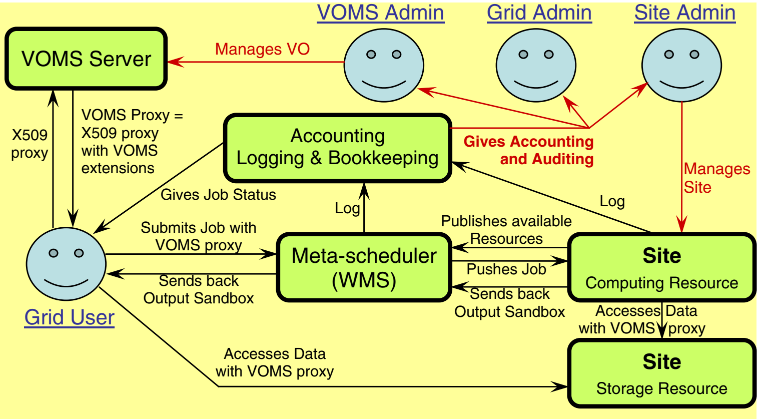
Examples of such service Grid infrastructures are EGEE, NorduGrid, OSG, DEISA, TeraGrid, Naregi.

2.2 Technical Presentation of Desktop Grids

A Desktop Grid (DG) is a *loose opportunistic Grid using idle resources*, as shown in Fig. 2 below. Inside desktop Grids, computing and storage resources are typically owned by individual volunteer owners and not by institutes (therefore it is often called “Volunteer Computing”). According to this, they are *not* managed and *not* authenticated. DG servers may be authenticated by an X509 certificate. DG users are authenticated by the DG servers but *not* by the computing and storage resources. Executables are validated, registered and deployed by managers of the DG servers, and DG users can create work units using these deployed applications by launching them with their own input data set. So, resource owners have to trust the DG servers. On the other hand, the DG servers do *not* fully trust the resource owners: For example, BOINC servers usually send each work unit redundantly to several computing resources, and then compare the results in order to assess their reliability [8].

Most DGs do not use the push model, but the *pull* model for applications execution: Since computing resources are not under control and

Fig. 1 SG = Service Grid = institutional computing Grid


VOMS Admin
Grid Admin
Site Admin
VOMS Server
Manages VO
X509 proxy
VOMS Proxy = X509 proxy with VOMS extensions
Accounting Logging & Bookkeeping
Gives Accounting and Auditing
Manages Site
Gives Job Status
Log
Log
Publishes available Resources
Submits Job with VOMS proxy
Meta-scheduler (WMS)
Site Computing Resource
Sends back Output Sandbox
Pushes Job
Sends back Output Sandbox
Accesses Data with VOMS proxy
Grid User
Accesses Data with VOMS proxy
Site Storage Resource

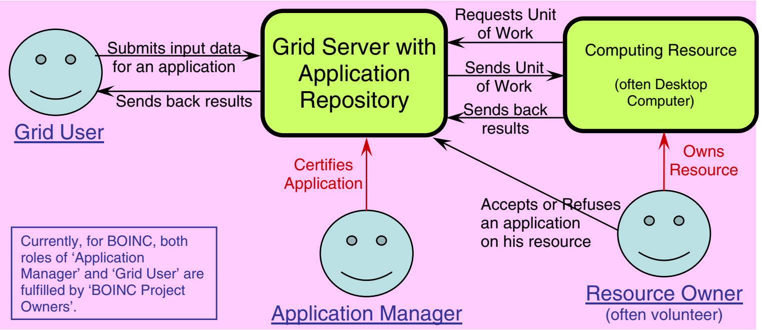

Grid User
Submits input data for an application
Sends back results
Grid Server with Application Repository
Requests Unit of Work
Sends Unit of Work
Sends back results
Computing Resource
(often Desktop Computer)
Owns Resource
Certifies Application
Accepts or Refuses an application on his resource
Currently, for BOINC, both roles of 'Application Manager' and 'Grid User' are fulfilled by 'BOINC Project Owners'.
Application Manager
Resource Owner
(often volunteer)

Fig. 2 DG = Desktop Grid = loose opportunistic Grid using idle resources

even not reachable, protected behind firewalls, starving computing resources *pull* work units from the DG server. The challenge in this concept is the ratio of correct results returned by computing resources to the number of work units they have pulled.

DGs can *not* run any kind of application, but only master/worker and parameter sweep applications, where the same application code should be executed with a large set of different data. It is the task of the DG server to store the data sets and send out the needed data subset for the resources asking for new work unit.

Examples of such desktop Grid systems are BOINC, XtremWeb, OurGrid, Xgrid.

3 Presentation of the EDGeS Project

The EDGeS project is a 2 years European FP7 project started on 01/01/2008. In this section, we present the goals of the project, the activities at work, and the way EDGeS handles interoperation and interoperability issues among SGs and DGs.

3.1 Goals of the EDGeS Project

From the technical point of view the main goal of EDgeS is to develop bridges that enable the interoperation of SGs and DGs. Both the SG → DG and DG → SG direction bridges should be created by EDGeS. However, it is not enough to create the prototype of these bridges, but it is equally important to establish a production infrastructure where these bridges can be used between EGEE, BOINC and XtremWeb systems. The production infrastructure will first connect only these three Grids. But in medium term, as shown in Fig. 3 below, the developed bridge technology (in gold at the centre) should enable the integration of many Service Grids (in yellow at the top) and Desktop Grids (in pink at the bottom), as WLCG already promotes integration between EGEE, NorduGrid and OSG.

A further important goal of EDGeS is to port existing SG and DG applications to the integrated SG-DG infrastructure and to provide a seamless job execution mechanism among the interconnected SG and DG systems.

3.2 Interoperation and Interoperability

The EDGeS project currently focuses on *practical interoperation, by building and deploying today ad-hoc bridges between EGEE, BOINC and XtremWeb*. In particular, XtremWeb users must have an X509 certificate, be registered in a VO and submit their jobs with an X509 proxy. BOINC Project Owners must have an X509 certificate, be registered in the EDGeS VO of EGEE and store a medium-term X509 proxy in a MyProxy server

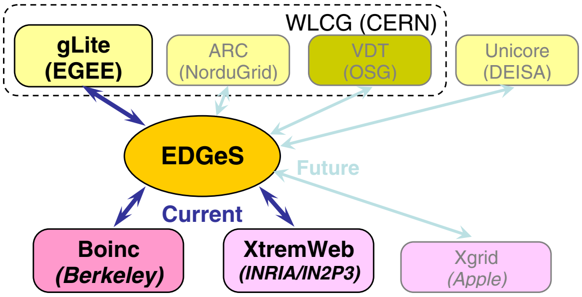


Fig. 3 Integration of service Grids and desktop Grids

of the EDGeS VO. All files must be transferred through the input and output sandboxes.

In the future, the EDGeS project will work on interoperability using OGF standards, in order to bridge more Grids (see Fig. 3 above), and provide better support of Grid file access (ByteIO, SRM, GridFTP and DMI).

4 Related Research in Bridging Desktop Grids and Service Grids

There exist two main approaches to bridge SG and DG (see Fig. 4 below).

In this section, we present the principles of these two approaches and discuss them according to architectural, functional and security perspective.

4.1 The Superworker Approach

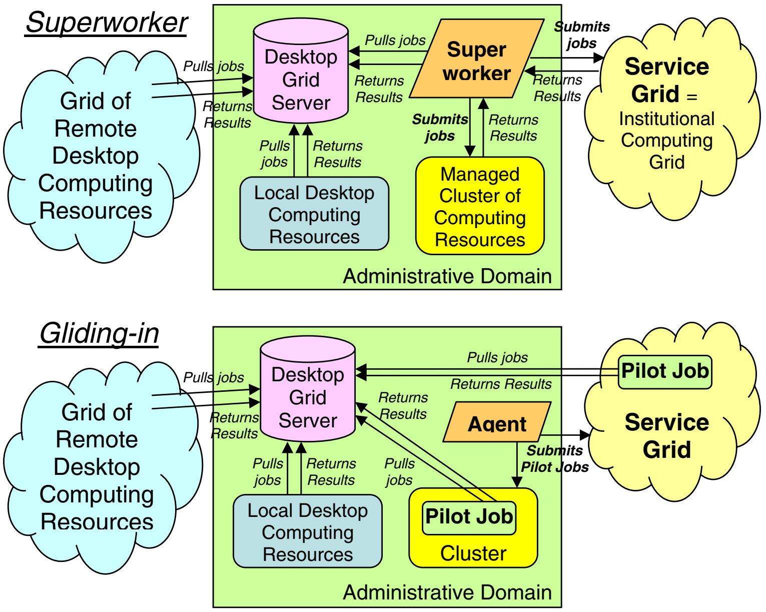
The *superworker*, proposed by the Lattice project [5] and the SZTAKI Desktop Grid [6], is the first solution. This enables the usage of several Grid or cluster resources to schedule DG tasks. The superworker is a bridge implemented as a daemon between the DG server and the SG resources. From the DG server point of view, the Grid or cluster appears as one single resource with large computing capabilities. The superworker continuously fetches tasks or work units from the DG server, wraps and submit the tasks accordingly to the local Grid or cluster resources manager. When computations are finished on the SG computing nodes, the superworker sends back the results to the DG server. Thus, the superworker by itself is a scheduler which needs to continuously scan the queues of the computing resources and watch for available resources to launch jobs.

Since the superworker is a centralized agent, this solution has several drawbacks:

1. the superworker can become a bottleneck when the number of computing resources increases,
2. it introduces a single point of failure in the system, which has low fault-tolerance.
3. the round trip for a work unit is increased because it has to be marshaled/unmarshaled by the superworker,

The superworker does not require modification of the infrastructure, but Grid security rules about traceability require that the actual job owner is the original job submitter. So, the original submitters must be authorized to submit jobs to the

Fig. 4 Bridging service Grid and desktop Grid, the superworker approach versus the gliding-in approach



Grid, and have to delegate their credentials to the superworker.

4.2 The Gliding-in Approach

The *Gliding-in* approach to cluster resources spread in different Condor pool using the Global Computing system (XtremWeb) was first introduced in [9]. The main principle consists in wrapping the XtremWeb worker as regular Condor task and in submitting this task to the Condor pool. Once the worker is executed on a Condor resource, the worker pulls jobs from the DG server, executes the XtremWeb task and return the result to the XtremWeb server. As a consequence, the Condor resources communicate directly to the XtremWeb server. This is permitted by standard firewall settings on most sites of Service Grids, which usually block all incoming connections (except those explicitly allowed), and allow all outgoing connections.

Mechanisms similar to those described just above are now commonly employed in Grid Computing [10]. For example, Dirac uses a combination of push/pull mechanism to execute jobs on several Grid clusters. The generic approach on the Grid is called a *pilot job*. Instead of submitting jobs directly to the Grid meta-scheduler, this system submits so-called pilot jobs. When executed, the pilot job fetches jobs from an external job scheduler.

The gliding-in or pilot job approach has several advantages. While simple, this mechanism efficiently balances the load between heterogeneous computing sites. It benefits from the fault tolerance provided by the DG server; if Grid nodes fail, then jobs are rescheduled to the next available resources. Finally, as the performance study of the Falkon system [11] shows, it gives better performances because series of jobs do not have to go through the meta-scheduler queue which generally has long waiting times, and communication is direct between the worker running on the Computing Element (CE) and the DG server without intermediate agent such as the superworker. Grid security rules about multi-user Pilot Jobs require that the actual job owner must *not* be the pilot job owner, but the original job submitter. EGEE will progressively enforce this job owner switching, which can now be achieved using *gLExec* [12].

5 Implementation of the SG–DG Bridges in EDGeS

In this section, we present the bridges and the application repository developed and implemented by the EDGeS project. The technologies used for the bridges are based on previous developments performed inside the SZTAKI Desktop Grid, and inside XtremWeb. The XtremWeb implementation used is XWHEP developed by IN2P3. The technology used for the application repository is based on the GEMLCA repository.

5.1 Architecture of the 3G Bridge

The integration of the EGEE, BOINC and XtremWeb systems theoretically requires 4 different bridges:

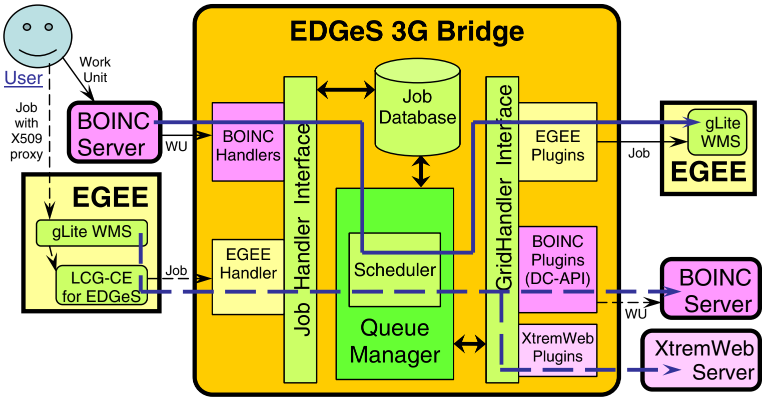
- BOINC → EGEE,
- XtremWeb → EGEE,
- EGEE → BOINC,
- EGEE → XtremWeb.

Instead of developing four different solutions for the four bridges, we rather created a Generic Grid to Grid (3G) bridge that can be easily adapted for the different cases of the required bridges. In fact, the 3G bridge is generic enough to easily adapt not only for the three Grid systems we tackle in EDGeS, but also for other Grid systems like GT2, GT4, OurGrid, Xgrid, etc. So the introduction and usage of the 3G bridge represents a significant step towards the standardized solution of integrating SGs and DGs.

The architecture of the 3G Bridge and its application in EDGeS is shown in Fig. 5 below. The role of the four core components of the 3G Bridge are as follows:

- The *Job Database* stores DG Work Units and SG jobs as generic descriptions. The Source Grid Handler Interface is used to place DG work units and SG jobs into the Job Database and to query their status.

Fig. 5 Architecture of the EDGeS 3G bridge



- The *Source Grid Handler Interface* is implemented via MySQL and as such it accepts SQL queries, inserts and updates. Besides the task of placing generic descriptions into the Job Database, its other task is to get job status information from it. Jobs/WUs coming from various source Grids are received by specific handlers that transfer the incoming jobs/WUs to the Source Grid Handler Interface. So, in order to connect a Grid as a source Grid to the 3G Bridge, a Grid-specific handler should be written for this source Grid.
- The *Queue Manager* periodically reads jobs from the Job Database and transmits them to the Target Grid Plug-in Interface.
- The *Target Grid Plug-in Interface* enables to connect various target Grids via their plug-in. The Target Grid Plug-in Interface provides a generic set of interface functions that should be implemented by the target Grid plug-ins. In order to connect a Grid as a target Grid into the 3G Bridge, a Grid-specific plug-in should be written for this target Grid. Note that the Grid plug-in is also responsible for querying the status of job/WU execution in the target Grid and retrieving the output of submitted jobs/WUs.

As Fig. 5 shows, the EDGeS 3G Bridge performs the three following bridging functionalities:

1. *BOINC to EGEE bridge*: Fig. 6 shows the BOINC handler and the EGEE plug-in.
2. *EGEE to BOINC bridge*: Fig. 8 shows the BOINC plug-in and the customized EGEE GRAM manager as EGEE handler of the 3G Bridge.
3. *EGEE to XtremWeb bridge*: Fig. 9 shows the XtremWeb plug-in and the modified EGEE GRAM manager as EGEE handler of the 3G Bridge. Note that the advantage of using 3G Bridge is that the EGEE handler can be the same for both the EGEE to BOINC bridge and for the EGEE to XtremWeb bridge.

5.2 Bridges BOINC → EGEE and XtremWeb → EGEE

The goal is to receive BOINC work units or XtremWeb jobs, and to make them execute inside the EGEE service Grid. The challenges are to securely authenticate the submitter with an X509 proxy acceptable by EGEE, and to wrap the incoming work unit or job as an EGEE job.

5.2.1 Bridge BOINC → EGEE

The main principle of the BOINC → EGEE bridge is to fetch work units from a BOINC server and to use the 3G bridge architecture presented in the previous section in order to translate these work units into EGEE jobs. Thus, in order to have a fully functional BOINC → EGEE bridge, two main additional components are needed: a *BOINC handler* and an *EGEE Target Grid Plug-In*.

The overview diagram of the complete system can be seen on Fig. 6 below.

The *BOINC handler* consists of two components shown on the left-hand side of the EDGeS 3G bridge:

- The *BOINC jobwrapper client* is a modified BOINC client which simulates a very powerful desktop PC, consisting of many CPU cores, and performing communication with the BOINC server in order to fetch work units and report their results. For each fetched work unit, the BOINC jobwrapper client creates a work unit description file describing the main properties of the WU, like the executable name, location and name of input and output files, and the command line arguments. Then, instead of starting the work unit on each simulated processor, the BOINC jobwrapper client starts one 3G jobwrapper application on behalf of each simulated processor (e.g. if the modified BOINC client simulates a PC consisting of ten processors, then it will start ten 3G jobwrapper applications).
- The *3G jobwrapper application* reads the work unit description created by the BOINC jobwrapper client in order to create a new BOINC work unit entry in the 3G Bridge database. From this point on, the 3G jobwrapper application periodically checks the status of the entry in the database. As soon as it finds that the work unit is terminated, the 3G jobwrapper application fetches available outputs from the 3G Bridge, and reports the status towards the BOINC jobwrapper client, which then considers the work unit as finished, and sends the result to the BOINC server at the next scheduling request.

The *EGEE Target Grid Plug-In*, shown on the right-hand side of the EDGeS 3G bridge, selects jobs from a queue in the 3G Bridge database and run them on EGEE. The EGEE plug-in can have different instances depending on the EGEE VO or user credential to be used for running the jobs. Thus, each EGEE plug-in is associated to a (queue name, VO name, credential) triple, where *queue name* is the name of the queue to use in the 3G Bridge database, *VO name* is the EGEE VO to be used, and *credential* is MyProxy access information for connection to the given EGEE VO. A given 3G jobwrapper application uses a configured queue name to explicitly select the VO and credential used to run the jobs on EGEE.

In order to handle EGEE jobs, the EGEE plug-in uses an advanced feature of gLite collection job submission. Using this feature, it is possible to send more jobs to the EGEE WMS in one submission request in order to lessen the load on the WMS and increase the number of jobs that can be handled. After the EGEE plug-in has sent the submission request for the collection, it retrieves one EGEE job identifier for each individual job within the collection.

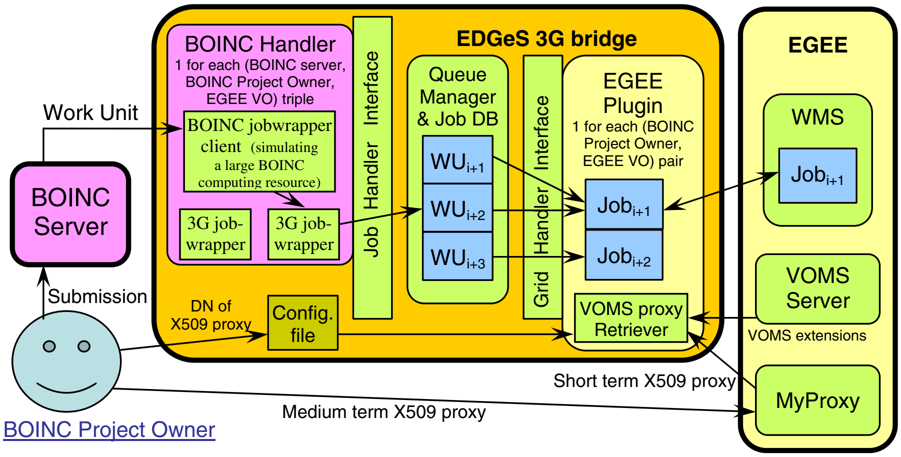


Fig. 6 Overview of the BOINC → EGEE bridge

The BOINC → EGEE bridge uses configuration files for following components:

- *BOINC jobwrapper client*: the configuration file contains the number of CPU cores to report, and the location of the 3G Bridge jobwrapper application to start instead of the work unit executables,
- *3G jobwrapper application*: the configuration file contains the target queue name and the location of the configuration file needed for database access,
- *EGEE plug-in*: the EGEE plug-in is configured within the 3G Bridge configuration file as the plug-ins are set up by the 3G Bridge on startup. For each EGEE plug-in instance, the following configuration variables have to be set: the plug-in type (EGEE for EGEE plug-ins), the VO name to use, the WMS URL, and MyProxy information for getting access to credentials (MyProxy hostname, port, DN of the X509 proxy of the BOINC project owner).

The security concepts of the two bridged Grids are different: on one hand, BOINC does not require X509 certificates, but permits adding of new work units only to the BOINC project owner, and on the other hand, EGEE provides access to every user with a registered X509 certificate. In order to gain access to EGEE resources and run the project's work units on EGEE, the BOINC project owner has to send (just only once) the DN of his X509 proxy to the 3G bridge administrator. Then, following standard EGEE rules, he stores his X509 proxy inside a MyProxy server which unconditionally trusts the EDGeS 3G bridge. As a consequence, the EDGeS 3G bridge must be operated with at least the same security level as a MyProxy server.. The bridge components generate detailed logging, so that the bridge administrator can quickly identify the BOINC project and work unit for an incidentally maliciously working EGEE job started by an EGEE plug-in.

5.2.2 Bridge XtremWeb → EGEE

The XtremWeb project had developed a prototype of a bridge to EGEE before the beginning of the EDGeS project. So, instead of creating a brand new XtremWeb handler inside the 3G bridge, we decided to upgrade the existing prototype into a production feature.

The general principle is an XtremWeb bridge performing job gliding [9, 10] using an XtremWeb worker job as a mono-user pilot job: When an XtremWeb worker job runs on an EGEE resource, it contacts the XtremWeb server, retrieves 1 user job, executes it, and sends back the result to the XtremWeb server.

Care must be taken that jobs sent to EGEE really belong to users having the right to access EGEE resources, and really permit traceability of the original submitter. This is achieved using an X509 proxy of the original submitter.

XtremWeb features a security model allowing both anonymous users (for instance volunteers giving their computing resources) and authenticated users (for instance EGEE users submitting jobs) to coexist within the same infrastructure. This security model (private, group, public), which follows the standard POSIX model (user, group, other), is enforced by the XtremWeb servers which manage access rights between users, applications, data and jobs.

The X509 proxy permitting to certify the XtremWeb user job for EGEE is a critical data which must be readable only by its owner. So, the user job must be an XtremWeb *private* user job, and the corresponding mono-user pilot job submitted by the XtremWeb bridge must be an XtremWeb *private* worker job.

The processing sequence, shown in Fig. 7 below, contains the following nine steps:

1. An XtremWeb user creates an X509 proxy, and contacts an EGEE VOMS server to obtain VOMS extensions allowing him to access EGEE resources,
2. The XtremWeb user submits to the XtremWeb server his user job with the X509 proxy and VOMS extensions, which permits to certify this user job,
3. The XtremWeb bridge periodically requests the XtremWeb server for user jobs certified by an X509 proxy. If such one is pending, the XtremWeb server sends him the user job certified with the X509 proxy,
4. The XtremWeb bridge submits to an EGEE WMS an XtremWeb worker job, which is a

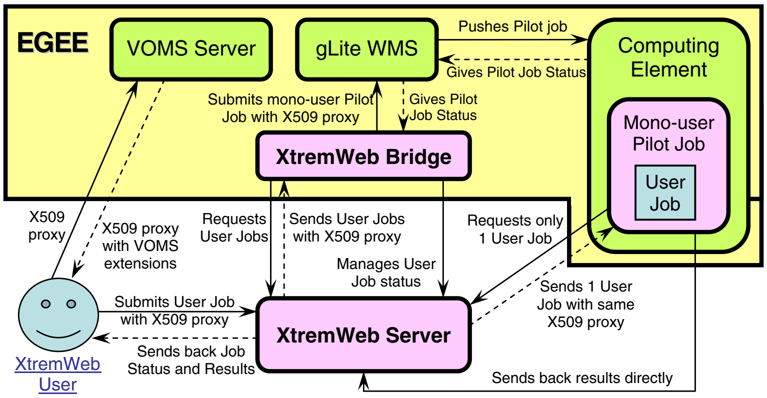


Fig. 7 Bridge XtremWeb → EGEE

mono-user pilot job with this X509 proxy (job description in *JDL*),

5. The EGEE WMS pushes the pilot job to an EGEE CE, which executes it,
6. The mono-user pilot job, which is an XtremWeb worker job, requests only the original user job from the XtremWeb server, and stops if it receives none,
7. The XtremWeb server verifies that the requested user job is certified with an X509 proxy, and sends the user job and the X509 proxy to the pilot job,
8. The pilot job verifies that the received X509 proxy is the same as its own X509 proxy, and executes the user job inside a sandbox [13–15],
9. At the end of the user job, the pilot job sends the job results directly to the XtremWeb server, then stops.

This satisfies EGEE security requirements [12, 16].

5.3 Bridges EGEE → BOINC and EGEE → XtremWeb

The goal is to receive EGEE jobs, and to make them execute as BOINC work units or XtremWeb jobs. The challenges are to present BOINC and XtremWeb resources as an *LCG-CE* (current version of the EGEE Computing Element) to EGEE, and to allow only jobs executing an application which already has been validated for execution inside desktop Grids.

5.3.1 GEMLCA Repository of Validated Applications

As already stated, owners of resources inside Desktop Grids are willing to donate their resources only to trusted applications. That requires an application repository for storing the validated and hence trusted applications.

In the case when an EGEE WMS submits (on behalf of an EGEE user) a job containing an application *not* present inside the application repository, the 3G bridge will *not* submit this job to BOINC, but will immediately notify a failure to the EGEE WMS. Depending on the number of retries allowed by the job description, the EGEE WMS can resubmit the job to another EGEE Computing Element.

The repository for validated applications must fulfill following five requirements:

1. Restrict content modification to only the Repository administrators, so that the EDGeS bridge can trust the contents of the repository.
2. Allow desktop Grid site administrators to publish when their site supports a given application, in order to permit the automation of the application registration process on the desktop Grid site.
3. Allow EGEE users and desktop Grid site administrators to query the available applications, so that the EGEE users can determine which applications can run on the desktop Grid infrastructure, and desktop Grid administrators can determine which applications they can register on their sites.

4. Allow EGEE users and desktop Grid site administrators to download repository contents: executables, shared libraries, input files etc. EGEE users need to download repository content when they want to submit to the WMS a job which can be forwarded to a desktop Grid, which requires a validated executable. Site administrators use the download facility when they get from the repository the application to be registered on their desktop Grid.
5. Allow the EGEE → Desktop Grid bridges to check the authenticity of an application received within a job, in order to determine whether the application can be forwarded to the desktop Grid infrastructure.

Prior EDGeS project, GEMLCA [17, 18] (Grid Execution Management for Legacy Code Applications), developed by the University of Westminster (UoW), already provided the six following repository functionalities, however its concept was focusing on the execution of the applications in the repository:

1. Management of repository entries, which describe applications with their inputs and outputs, parameters and necessary execution environments.
2. Retrieve list of validated applications with human readable description and id.
3. Search (using WSRF) the id, status (public, private, system) and the list of desktop Grid sites supported by each validated application.
4. Retrieve files with GridFTP from a non-hierarchical virtual filesystem. (Files related to a single validated application should reside in the same directory).
5. Retrieve application data from XML files to GEMLCA's proprietary format
6. Determine application ownership based on the certificate subject of the user.

In order to comply with the requirements, we have extended GEMLCA with following two additional repository functionalities:

1. Better searching facility now provides interfaces for searching applications with their version information too. This helps the desktop Grid site administrators to determine whether an installed application on their site is different than the one stored in the repository.
2. An interface to collect file hashes of the repository contents. This helps the EDGeS bridge to determine whether an application submitted is the same what can be found in the repository.

Finally, University of Westminster offers a portlet for administering GEMLCA services. This portlet has been enhanced to handle also the EDGeS enhanced GEMLCA versions and offers help for the desktop Grid administrators for registering their sites. It also provides a simple interface for the EDGeS validated application repository administrator for managing the repository contents.

5.3.2 Bridge EGEE → BOINC

The EGEE → BOINC bridge is based on the 3G Bridge. The rough difference with the BOINC → EGEE bridge described in a preceding chapter is that the EGEE → BOINC bridge needs an EGEE handler for receiving EGEE jobs and a BOINC plug-in for sending jobs to the BOINC target Grid.

The EGEE → BOINC bridge must be a transparent solution for EGEE users, i.e. a traditional EGEE user with an EGEE User Interface access should be able to submit jobs and query job progress information with traditional EGEE command-line tools. Moreover, a BOINC resource should be visible in the EGEE Information System like a traditional EGEE Computing Element. Thus, from EGEE's point of view, the solution must fulfil the two following requirements:

1. proper information reporting towards the BDII server,
2. proper interaction with the WMS and the L&B servers.

Before the implementation, we have examined different CE solutions offered by EGEE: LCG-CE, GLITE-CE and CREAM CE. LCG-CE is mature and widely used. GLITE-CE is a proof-of-concept solution, not recommended for production by EGEE. CREAM-CE is a new technology based on Web Services, working correctly when accessed directly, but *not* accessible through the

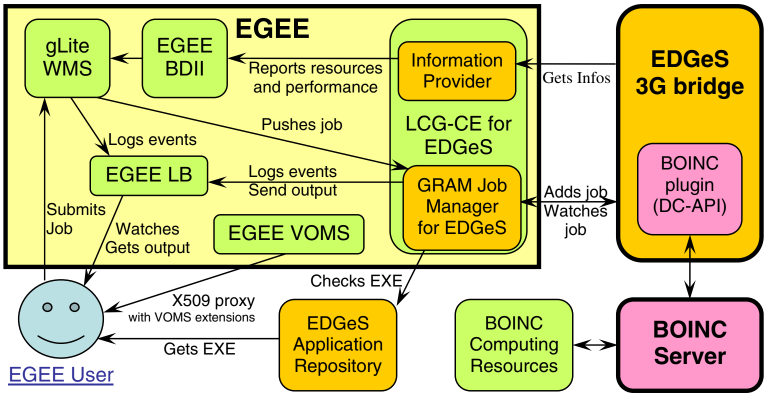


Fig. 8 The EGEE → BOINC bridge

EGEE WMS yet. As a consequence, we have decided to create a solution based on LCG-CE, as shown in Fig. 8. Later, when the CREAM interface becomes stable, we will offer both the LCG-CE and the CREAM-CE interfaces for the 3G bridge.

For the processing of a job, a standard LCG-CE performs the three following steps:

1. It submits a wrapper script to a selected GRAM jobmanager on the CE,
2. Before running the wrapper script on a worker, it starts a helper script which periodically updates the proxy on the worker node in a background process and starts the wrapper script,
3. The wrapper script contains every information about the job defined in variables, like the name of the executable, command line arguments, name and URL of input files, name and expected location of output files, etc. The wrapper script first fetches input files from their location (from the WMS), then runs the executable, and finally copies the output files to their specified location. During the process, the wrapper script sends status change events to the L&B server, and produces output in some WMS-specific files.

The EGEE → BOINC bridge has to perform almost the same steps as above, using a somewhat modified LCG-CE with a 3G bridge-specific job-manager. However, there are some differences:

- the bridge must check if the received executable is a validated application on the connected BOINC server. Using the EDGeS Application Repository described in the previous subsection, the bridge checks if the executable is registered as an EGEE application, if this application has a BOINC variant and it is registered on the connected BOINC project.

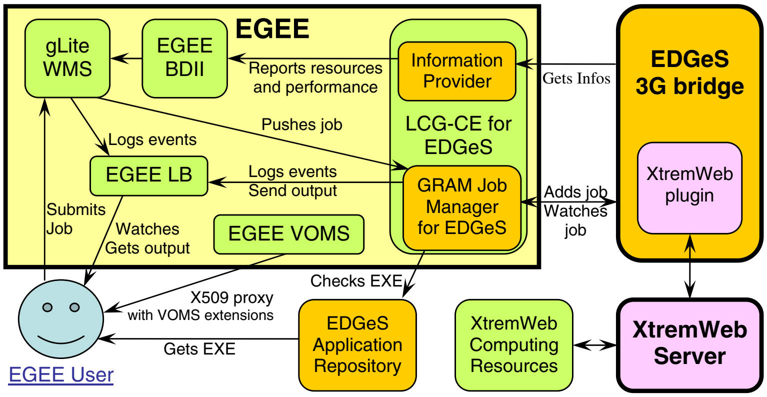


Fig. 9 Bridge EGEE → XtremWeb

The job is accepted if and only if all these statements are true,

- the helper script doesn't have to be started, as the proxy doesn't leave the CE, so there is no need to update proxies on Worker Nodes,
- there is no need to run the wrapper script, it only has to be parsed, so our GRAM jobmanager can send the job to the 3G Bridge and can interact with the L&B server instead of the wrapper script,
- moreover, the GRAM jobmanager periodically has to check the status of the job in the 3G Bridge database, and update the status of the job for EGEE.

In order to implement the EGEE → BOINC bridge, we have extended the 3G bridge with a Web Service interface. So, there is no need to place the 3G bridge and the BOINC server on the same machine as the EDGeS CE: the 3G bridge and the BOINC server are completely separated from the EDGeS CE, which uses the WS interface to communicate with the 3G bridge.

On BOINC side, the DC-API [6] plug-in is used to create BOINC work units out of entries in the 3G Bridge Job Database, query their status and get the results of processed work units. Once work units are created in the BOINC database, they are processed sooner or later by attached BOINC clients. If desired, the BOINC server can perform any job redundancy and checking as usual.

The EGEE → BOINC bridge publishes information to the BDII according to *GLUE 1.3*, contains an EGEE producer and a BOINC GIP plug-in. The BOINC plug-in is responsible for reporting performance information about the BOINC project. For this, BOINC statistics provided by the BOINC project are used.

5.3.3 Bridge EGEE → XtremWeb

The general principle of creating the EGEE → XtremWeb bridge is the same as in the case of the EGEE → BOINC bridge since both solutions use the 3G bridge as the heart of the EGEE → DG bridges. The architecture of EGEE → XtremWeb bridge, depicted in Fig. 9, clearly shows that the EGEE → 3G bridge part is the same as the one shown in Fig. 8 above for the EGEE → BOINC case. The only difference is the replacement, inside the 3G bridge, of the BOINC plug-in by the XtremWeb plug-in.

6 Current Operational Status of EDGeS

The EDGeS 3G bridge is not an pure research prototype, but implementations are in *real operation* between EGEE and Desktop Grids, as shown in Fig. 10 below:

6.1 Operational DG → EGEE Infrastructure

The DG → EGEE bridges of the EDGeS system have been prototyped in June 2008 and put into operation in September 2008.

The BOINC → EGEE bridge is currently in operation at SZTAKI in Budapest, Hungary.

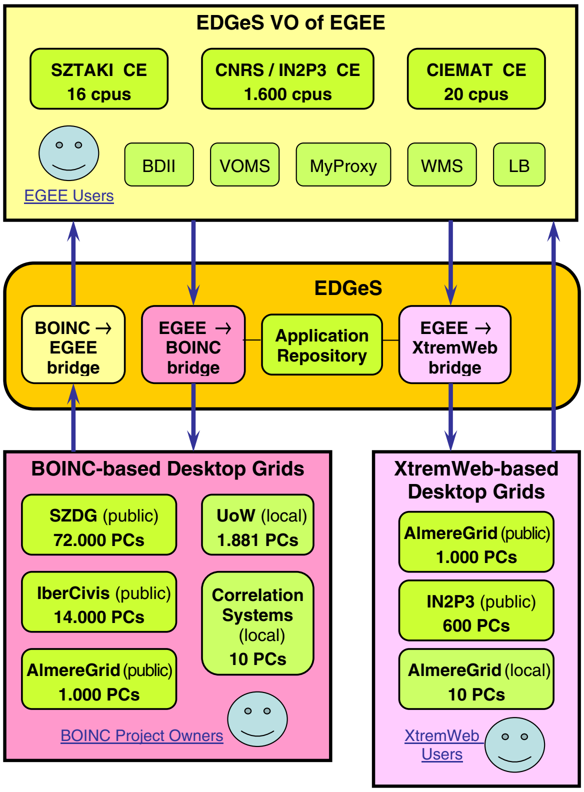


Fig. 10 EDGeS operational EGEE ←→ DG infrastructure

It connects the public BOINC-based SZTAKI desktop Grid (SZDG) in Budapest, Hungary, the public BOINC-based Ibercivis desktop Grid in Spain, the local UoW desktop Grid at London, Great Britain, and the local DG of the Correlation Systems Ltd., Israel to a dedicated external VO of EGEE, named desktopgrid.vo.edges-grid.eu.

The XtremWeb → EGEE bridge is now included in the standard distribution of the XtremWeb middleware. It is in operation at the local IN2P3 desktop Grid at Orsay, France, at the public INRIA desktop Grid in Saclay, France, and at the AlmereGrid desktop Grid in Almere, The Netherlands.

In the operational DG → EGEE infrastructure of EDGeS, the desktopgrid.vo.edges-grid.eu VO currently has access to three computing elements as EGEE resources:

- *CNRS/IN2P3* CE (approx. 1,600 CPUs, non dedicated)
- *SZTAKI* CE (16 CPUs, dedicated)
- *CIEMAT* CE (20 CPUs, dedicated)

It also contains the basic EGEE core services like BDII, VOMS, MyProxy, WMS and LB.

6.2 Operational EGEE → DG Infrastructure

The EGEE → DG bridges of the EDGeS system have been prototyped in December 2008 and put into operation for the EGEE users in March 2009.

In the operational EGEE → DG infrastructure of EDGeS, where the desktopgrid.vo.edges-grid.eu VO currently has access to six desktop Grids:

- *AlmereGrid*: Public BOINC DG in Almere, The Netherlands (1,000 PCs)
- *SZDG*: Public BOINC DG in Budapest, Hungary (72,000 PCs)
- *University of Westminster*: Local BOINC DG in London, UK (1,881 PCs)
- *IN2P3*: Public XtremWeb DG in Orsay, France (600 PCs)
- *AlmereGrid*: Public XtremWeb DG in Almere, The Netherlands (1,000 PCs)
- *AlmereGrid*: Local XtremWeb DG in Almere, The Netherlands (ten PCs)

As soon as the BOINC-based Extremadura desktop Grid will be available, it will also be connected to the EDGeS operational infrastructure.

Table 1 Applications already ported to EDGeS

Applications already ported to EDGeS	Organisation	Runs on desktop Grid	Runs on EGEE (EDGeS VO)
Video Stream Analysis in a Grid Environment (VISAGE)	Correlation Systems Ltd Israel	√	√
Digital Alias-free Signal Processing	University of Westminster	√	√
Protein Molecule Simulation using Autodock	University of Westminster	√	√
E-Marketplace Model Integrated with Logistics (EMMIL)	SZTAKI	√	√
Anti-cancer Drug Design (CancerGrid)	SZTAKI	√	√
Cellular Automata based Laser Dynamics (CALD)	University of Seville and University of Westminster	√	√
Signal and Image Processing using GT4 Tray	Forschungszentrum Karlsruhe	√	
Analysis of Genotype Data (Plink)	Atos Origin	√	
Distributed Audio Retrieval using TRIANA (DART)	Cardiff University	√	√
Fusion Plasma Application (ISDEP)	BIFI	√	
3-D Video Rendering using Blender	University of Westminster	√	
Profiling Hospitals in the UK based on Patient Readmission Statistics	University of Westminster	√	√

6.3 Applications Already Ported to EDGeS

There are already several applications that were already ported to the DG → EGEE infrastructure and they have been using the DG → EGEE infrastructure since September 2008. These applications are shown in Table 1 below.

6.4 Load Limit and Performances

The EDGeS 3G bridge currently supports a load of several thousands jobs waiting to be completed, with a sustained transfer throughput of 7 jobs/s.

The latency introduced by the bridge components themselves has been measured to be 2 s, but the current conservative sleep time of 300 s of the jobwrapper introduces an average latency of $300/2 = 150$ s.

7 OGF Standards Relevant for Future Grid Interoperability

In order to bridge more Grids, and provide better support of Grid file access, the EDGeS project will work on interoperability using following OGF recommendations, which are published using documents named GDF.

Figure 11 below presents the relationships between OGF standards related to computation:

Following recommendations are currently in the process of implementation by the gLite middleware of EGEE:

- For *information retrieval and publication*: *GLUE* is an information model for Grid entities described using natural language and enriched with a graphical representation using UML Class Diagrams. Therefore, GLUE is the fundamental basis permitting to publish and retrieve information about Grid entities. When gLite will migrate from GLUE 1.3 to GLUE 2.0, defined in document GFD.147 [19], the EDGeS bridge will also migrate, in order to publish useful information to the BDII, and to retrieve necessary information from it.
- For *job management*: *BES* (Basic Execution Service), defined in document GFD.108 [20], is a service to which clients can send requests to initiate, monitor, and manage

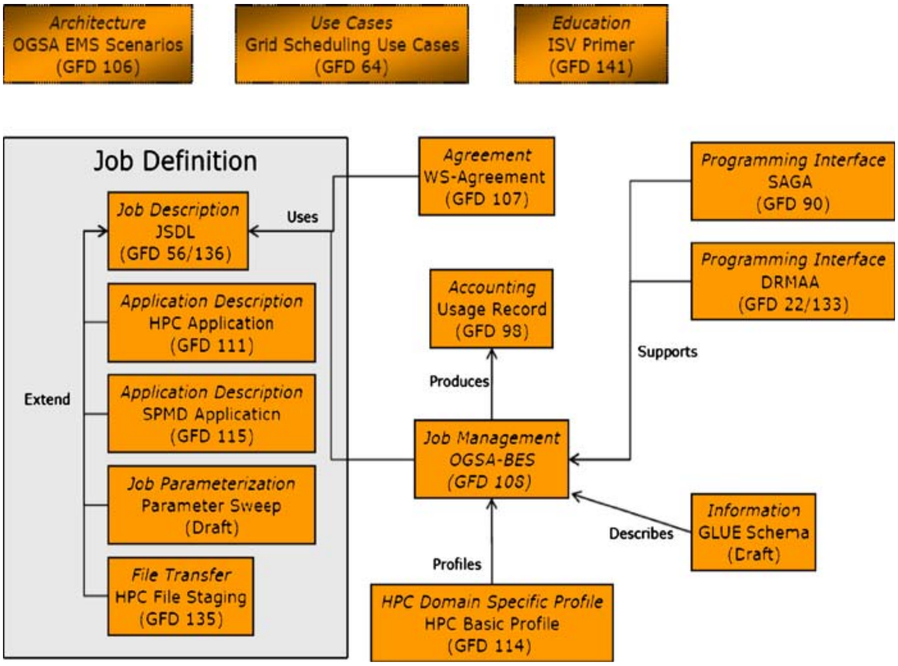


Fig. 11 OGF standards related to computation

computational activities. It defines an extensible state model for activities, and an extensible information model for a BES implementation and for the activities that it creates. The EDGeS bridge will use BES to receive job submissions, for example from the GridSphere Portal. The EDGeS bridge will also use BES to submit jobs to the BES-compliant CREAM CE when this latter will really be in operation.

- For *job description*: *JSDL* (Job Submission Description Language), defined in document GFD.136 [21], is used to describe the requirements of computational jobs for submission to resources, particularly in Grid environments. When gLite will migrate from JDL to JSDL, the EDGeS bridge will also migrate, in order to describe submitted jobs.
- For *data management*: *GridFTP*, defined in documents GFD.20 [22] and GFD.21 [23], adds Grid extensions to the File Transfer Protocol (FTP) specified in RFC 959. *SRM*, defined in document GFD.129 [24], is an open specification for Storage Resource Managers, which are Grid storage services providing interfaces to storage resources, as well as advanced functionality such as dynamic space allocation and file management on shared storage systems. EGEE is a main contributor to these two recommendations, and already implements them. So the EDGeS project will have to take these recommendations into account inside its "data management" activity.

Following OGF recommendations could also potentially be used:

- For *accounting records*: *UR* (Usage Record), defined in document GFD.98 [25], describes a common format, encompassing both job level accounting and aggregate accounting, with which to exchange basic accounting and usage data over a Grid instantiation. *RUS* (Resource Usage Service) [26] is intended to accommodate requirements on Grid resource usage auditing and accounting as well Grid economic model.
- For *data transfers*: *ByteIO*, defined in documents GFD.87 [27] and GFD.88 [28], describes efficient manipulation of, access to, and management of bulk data sources and sinks in the Grid.
- For *file management*: *DMI* (Data Movement Interface), defined in document GFD.134 [29], specifies the support of the instantiation and management of data transfers within and across Grid deployments.
- For the GEMLCA *application repository*: *ACS* (Application Contents Service), defined in document GFD.73 [30], is an OGSA service, which maintains Application Contents as an Application Archive. The ACS repository provides functions to retrieve the contents and their change histories. ACS also defines a standard format of an Application Archive for its management and exchange.

OGF recommendations for *authentication and authorization* are highly desirable, because they are transverse to all other Grid services, and their standardisation is a prerequisite for any interoperability. But the OGF AUTHZ working group, who is officially in charge of these subjects, has currently *no* published recommendation. The OGF PGI working group, where EDGeS actively participates, has achieved a survey of the existing more or less compatible implementations, and is working to publish proposals.

8 Conclusions

EDGeS so far achieved to put into operation the integrated DG-EGEE infrastructure into the DG → EGEE and EGEE → DG directions in the dedicated EDGeS desktop Grid VO.

For the DG → EGEE direction, it means that work units of the connected desktop Grids can be seamlessly executed in the EDGeS VO of EGEE. This helps to increase the available processing capacity for the connected local and public DGs. It is especially useful for small local DGs like the DG of Correlation Systems Ltd.

For the EGEE → DG direction, it means that EGEE jobs using validated applications can be seamlessly executed in the DGs connected to EDGeS. Intensive work is currently performed to port the existing EGEE applications to the

EDGeS infrastructure. Methodology of porting EGEE applications to EDGeS has already been developed as well as the validation procedure of such applications.

The developed EGEE-DG infrastructure is created in a way to be easily adapted for other SG and DG systems. The 3G Bridge that is the heart of the EDGeS infrastructure is easily adaptable for other Grids: only the necessary source handlers and target plug-ins should be written for the Grids to be connected into EDGeS. The solution is not specific to the EDGeS desktop Grid VO, but can also be used for different VOs of EGEE. Available standards are also considered for future adaptation of the EDGeS technology in order to solve the long-term interoperability issues of service Grids and desktop Grids.

Acknowledgements The EDGeS (Enabling Desktop Grids for e-Science) project receives Community funding from the European Commission within Research Infrastructures initiative of FP7 (grant agreement Number 211727).

References

1. Riedel, M., et al.: Interoperation of world-wide production e-Science infrastructures. In: Concurrency and Computation: Practice and Experience (2008)
2. Anderson, D., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: Seti@home: an experiment in public-resource computing. Commun. ACM **45**(11), 56–61 (2002)
3. Anderson, D.: BOINC: a system for public-resource computing and storage. In: Proceedings of the 5th IEEE/ACM International GRID Workshop, Pittsburgh, USA (2004)
4. Fedak, G., Germain, C., Nri, V., Cappello, F.: XtremWeb: a generic global computing platform. In: Proceedings of 1st IEEE International Symposium on Cluster Computing and the Grid CCGRID'2001, Special Session Global Computing on Personal Devices, pp. 582–587. IEEE/ACM, IEEE Press, Brisbane, Australia (2001)
5. Myers, D.S., Bazinet, A.L., Cummings, M.P.: Expanding the Reach of Grid Computing: Combining Globus- and BOINC-based Systems, Chapter Grids for Bioinformatics and Computational Biology. Wiley Book Series on Parallel and Distributed Computing (2008)
6. Balaton, Z., Gombas, G., Kacsuk, P., Kornafeld, A., Kovacs, J., Marosi, A.C., Vida, G., Podhorszki, N., Kiss, T.: SZTAKI Desktop Grid: a modular and scalable way of building large computing Grids. In: Proc. of the 21st International Parallel and Distributed Processing Symposium, Long Beach, California, USA, 26–30 March 2007
7. Cardenas-Montes, M., Emmen, A., Marosi, A.C., Araujo, F., Gombas, G., Terstyanszky, G., Fedak, G., Kelley, I., Taylor, I., Lodygensky, O., Kacsuk, P., Lovas, R., Kiss, T., Balaton, Z., Farkas, Z.: Edges: bridging desktop and service Grids. In: Proceedings of the 2nd Iberian Grid Infrastructure Conference, University of Porto, Portugal, 12–14 May 2008
8. Sarmenta, L.F.G.: Sabotage-tolerance mechanisms for volunteer computing systems. Future Gener. Comput. Syst. **18**(4), 561–572 (2002)
9. Lodygensky, O., Fedak, G., Cappello, F., Neri, V., Livny, M., Thain, D.: XtremWeb & condor: sharing resources between internet connected condor pools. In: Proceedings of CCGRID'2003, Third International Workshop on Global and Peer-to-Peer Computing (GP2PC'03), pp. 382–389. IEEE/ACM, Tokyo, Japan (2003)
10. Thain, D., Livny, M.: Building reliable clients and services. In: The GRID2, pp. 285–318. Morgan Kaufman (2004)
11. Raicu, I., Zhao, Y., Dumitrescu, C., Foster, I., Wilde. M.: Falkon: a fast and light-weight task execution framework. In: IEEE/ACM SuperComputing (2007)
12. Sfiligoi1, Koeroo, O., Venekamp, G., Yocum1, D., Groep, D., Petravick, D.: Addressing the Pilot security problem with gLExec. Technical Report FERMILAB-PUB-07-483-CD, Fermi National Laboratory (2007)
13. Alexandrov, A., Kmiec, P., Schauser, K.: Consh: a confined execution environment for internet computations. In: Proceedings of the Usenix Annual Technical Conference. http://www.usenix.org/events/usenix99/ (1999)
14. Acharya, A., Raje, M.: Mapbox: using parameterized behavior classes to confine applications. In: Technical report TRCS99-25, University of California, Santa Barbara (1999)
15. Goldberg, I., Wagner, D., Thomas, R., Brewer, E.: A secure environment for untrusted help application—confining the wily hacker. In: Proceedings of the 6th Usenix Security Symposium (1996)
16. Caillat, G., Lodygensky, O., Urbah, E., Fedak, G., He, H.: Towards a security model to bridge internet desktop Grids and service Grids. In: Lecture Notes in Computer Science (2008)
17. Terstyanszky, G., Kiss, T., Kacsuk, P., Delaitre, T., Kecskemeti, G., Winter, S.: Legacy code support for commercial production Grids. In: Conf. Proc. of the UK E-Science All Hands Meeting, Nottingham, UK, ISBN 0-9553988-0-0, 18–21 September 2006
18. Delaitre, T., Kiss, T., Goyeneche, A., Terstyanszky, G., Winter, S., Kacsuk, P.: GEMLCA: running legacy code applications as Grid services. In: Journal of Grid Computing, vol. 3. No. 1–2, pp. 75–90. Springer Science + Business Media: 1570–7873 (2005)
19. Andreozzi, S., Burke, S., Ehm, F., Field, L., Galang, G., Konya, B., Litmaath, M., Millar, P., Navarro, J.P.: [GFD.147] GLUE Specification v. 2.0. http://www.ogf.org/documents/GFD.147.pdf (2009)

20. Foster, I., Grimshaw, A., Lane, P., Lee, W., Morgan, M., Newhouse, S., Pickles, S., Pulsipher, D., Smith, C., Theimer, M.: [GFD.108] OGSA® Basic Execution Service Version 1.0. http://www.ogf.org/documents/GFD.108.pdf (2007)
21. Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A.: [GFD.136] Job Submission Description Language (JSDL) specification, version 1.0. http://www.ogf.org/documents/GFD.136.pdf (2008)
22. Allcock, W., Bester, J., Bresnahan, J., Meder, S., Plaszczak, P., Tuecke, S.: [GFD.20] GridFTP: protocol extensions to FTP for the Grid. http://www.ogf.org/documents/GFD.20.pdf (2003)
23. Mandrichenko, I.: GridFTP Protocol Improvements. [GFD.21] http://www.ogf.org/documents/GFD.21.pdf (2003)
24. Sim, A., Shoshani, A., Badino, P., Barring, O., Baud, J.-P., Corso, E., De Witt, S., Donno, F., Gu, J., Haddox-Schatz, M., Hess, B., Jensen, J., Kowalski, A., Litmaath, M., Magnoni, L., Perelmutov, T., Petravick, D., Watson, C.: [GFD.129] The storage resource manager interface specification version 2.2. http://www.ogf.org/documents/GFD.129.pdf (2008)
25. Mach, R., Lepro-Metz, R., Jackson, S.: [GFD.98] Usage record—format recommendation. http://www.ogf.org/documents/GFD.98.pdf (2007)
26. Chen, X., Khan, A., Ainsworth, J., Newhouse, S., MacLaren, J.: WSI Resource Usage Service (RUS) Core Specification (draft 19). http://forge.gridforum.org/sf/docman/do/downloadDocument/projects.rus-wg/docman.root.documents.version_1_0.draft_wsi_rus_19/doc14304/1 (2007)
27. Morgan, M.: [GFD.87] ByteIO specification 1.0. http://www.ogf.org/documents/GFD.87.pdf (2006)
28. Morgan, M.: [GFD.88] ByteIO OGSA® WSRF Basic Profile Rendering 1.0. http://www.ogf.org/documents/GFD.88.pdf (2006)
29. Antonioletti, M., Drescher, M., Luniewski, A., Newhouse, S., Madduri, R.: [GFD.134] OGSA-DMI Functional Specification 1.0. http://www.ogf.org/documents/GFD.134.pdf (2008)
30. Fukui, K.: [GFD.73] Application contents service specification 1.0. http://www.ogf.org/documents/GFD.73.pdf (2006)