EDGeS: bridging Desktop and Service Grids

Miguel Cárdenas-Montes[1], Ad Emmen[7], Attila Csaba Marosi[2], filipe Araujo[9], Gábor Gombás[2], Gabor Terstyanszky[3], Gilles Fedak[56], Ian Kelley[4], Ian Taylor[4], Oleg Lodygensky[7], Péter Kacsuk[2], Róbert Lovas[2], Tamas Kiss[3], Zoltán Balaton[2] and Zoltán Farkas[2]

[1] Extermadura Advanced Research Center (CETA-CIEMAT), Trujillo, SPAIN,
miguel.cardenas@ciemat.es
[2] MTA SZTAKI, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, Hungary
(atisu, gombasg, kacsuk, rlovas, balaton, zfarkas)@sztaki.hu
[3] Centre for Parallel Computing, University of Westminster, London, United Kingdom
(kisst, G.Z.Terstyanszky)@westminster.ac.uk
[4] School of Computer Science, Cardiff University, Cardiff, United Kingdom
(I.R.Kelley, Ian.J.Taylor)@cs.cardiff.ac.uk
[5] INRIA Saclay, Grand-Large, Orsay, France
[6] LRI, Univ Paris-Sud, CNRS, Orsay, France
fedak@lri.fr
[7] LAL Universite Paris Sud, CNRS, IN2P3, France
lodygens@lal.in2p3.fr
[8] AlmereGrid, Almere, The Netherlands
ad@almeregrid.nl
[9] Faculty of Sciences and Technology of the University of Coimbra, Coimbra, Portugal
filipius@dei.uc.pt

Abstract. Service grids and desktop grids are both promoted by their supportive communities as great solutions for solving the available compute power problem and helping to balance loads across network systems. Little work, however, has been undertaken to blend these two technologies together in an effort to create one vast and seamless pool of resources. In this paper we will introduce a new FP7 infrastructures project, entitled Enabling Desktop Grids for e-Science (EDGeS), that is building technological bridges to facilitate service and desktop grid interoperability. We provide a taxonomy for existing state of the art grid systems and background into service grids, such as EGEE and volunteer computing platforms, such as BOINC and XtremWeb. We then describe our approach within three themes for identifying translation technologies for porting applications between service grids and desktop grids and vice versa. The individual themes discuss the actual bridging technologies employed, the distributed data issues surrounding deployment and application development and user access issues.

Keywords: Desktop Computing, Service Computing, BOINC, EGEE, XtremWeb, Distributed Data.

1 Introduction

There is a growing interest among scientific communities to use Grid computing infrastructures to solve their grand-challenge problems and to further enhance their applications with extended parameter sets and greater complexity. Such enhancements were often limited or unattainable in compute systems prior to the era of Grid computing due to increased resource requirements. However, even existing grids are often smaller than many new scientific communities and their complex applications would like to use.

E-infrastructures play a distinguished role in enabling large-scale innovative scientific research. In order to establish such e-infrastructures, various grids have been created and run as a service for the scientific community. Originally, the aim of Grid systems was that anyone (donors) could offer resources for a given Grid, and anyone (users) could claim resources dynamically, according to their actual needs, in order to solve a computational or data intensive task. This twofold aim has however not fully been achieved, and we can today observe two different trends in the development of Grid systems: Service Grids and Desktop Grids.

Researchers and developers in Service Grids (SGs) 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 maintain. It is therefore natural, that individuals do not often 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. Examples of such infrastructures are EGEE, the NorduGrid, or the NGS (National Grid Service) in the UK. 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 around forty thousand processors. 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 Organisation (VO) or community authorisation services that centralise the management of trust relationships and access rights.

Desktop Grids (DGs) on the other hand are commonly known as *volunteer computing systems* or *Public-Resource Computing*, because they often rely upon the general public to donate resources. i.e. "spare cycles" or storage space. Unlike Service Grids, which are based on complex architectures, volunteer computing has a simple architecture and has demonstrated the ability to integrate dispersed, heterogeneous computing resources with ease, successfully scavenging cycles from tens of thousands of idle desktop computers. This paradigm represents a complementary trend concerning the original aims of Grid computing. In Desktop Grid systems, anyone can bring resources into the Grid, installation and maintenance of the 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 Desktop Grid resources for computation. The most well-known DG example is

the SETI@HOME [1] project, in which approximately four million PCs have been involved.

DGs However, cannot work as services nor be used by anyone who has not already setup their project to function in this environment. Additionally, unlike most Service Grids, which have reciprocal agreements for resource utilisation among partners, participants in DG systems, cannot use the system for their own goals. Because of this limitation, the Grid research community considers DGs only as particular and limited solutions. Until now, these two kinds of Grid systems have been completely separated and hence there has not been a mechanism to be able exploit their individual advantageous features in a unified environment. However, with the objective to support new scientific communities that need extremely large numbers of resources, the solution could be to interconnect these two kinds of Grid systems into an integrated Service GridDesktop Grid (SG-DG) infrastructure.

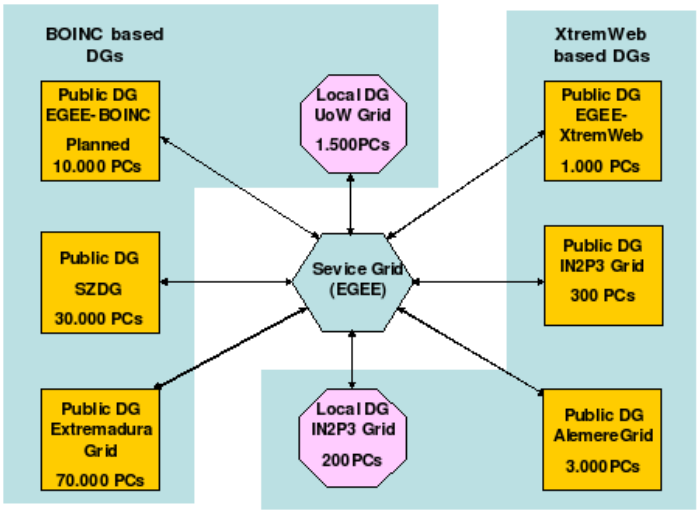

Fig. 1. Structure of the new combined e-infrastructure.

In this paper, we describe research on how such an integrated SG-DG infrastructure can be established, how applications can be adapted and developed for such an infrastructure, and how the execution of these applications can be controlled and managed. In the next section, we provide a taxonomy of existing systems. We then describe the related work and core technologies we are working with in service and desktop grids in Section 3. In Section 4, we provide an outline of the three main areas of research within the EDGeS project, in providing a SG-DG bridge, application development and user access, and the distributed data access concerns between such systems. In Section 5, we present our concluding remarks.

2 Taxonomy of Existing Desktop and Service Grids

The main distinguishing feature between SGs and DGs is the way computations are initiated at the resources of the grid. In Service Grids a job submission or a service invocation is used to initiate activity on a grid resource. Both can be considered as a specific form of the *push model* where the service requestor pushes jobs, tasks, service invocations on the passive resources. Once such a request is pushed on the resource, it becomes active and executes the requested activity. Desktop Grids work according to the *pull model*. Resources that have got spare cycles pull tasks from the application repository which is typically placed on the DG server. In this way resources play an active role in the DG system, they initiate their own activity based on the task pulled from the server.

Both SGs and DGs can be public (global) and non-public (local), figure 2. A public (or global) grid refers to a grid that connects resources from different administrative domains, which are typically interconnected by wide-area network. Non-public (or local) grids, on the other hand, connect resources within the same administrative domain by using a local-area network or a VPN. Typical public service grids are EGEE, OSG[2], TeraGrid [3], etc. Non-public service grids are typically interconnected local clusters (for example university wide local Grids like the Oxford Campus Grid[4]). Both public and local desktop grids can be further divided as volunteer and non-volunteer DGs. Resources of volunteer DGs are collected from individual desktop owners as their volunteer contribution to the Desktop Grid. Typical public, volunteer DGs are the BOINC-based DG systems like SETI@HOME, Einstein@HOME[5], SZTAKI Desktop Grid [6], etc. AlmereGrid[7] and XtremWeb[8] are also volunteer, public DG systems.

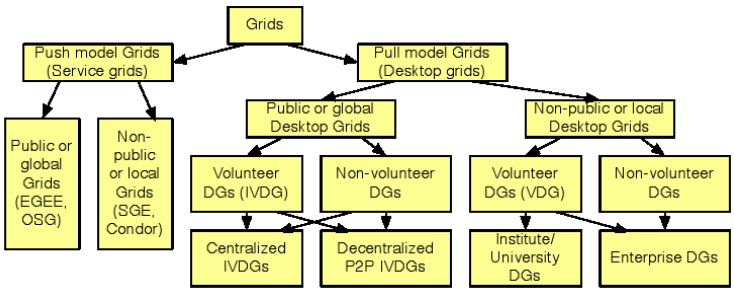

Fig. 2. Taxonomy of grid systems.

In a non-volunteer DG individual desktop owners are instructed to contribute their resources to the DG. Examples for such non-volunteer DGs are the Extremadura School DG and the Westminster DG. The Extremadura School DG is a public non-volunteer DG where the regional government instructed the schools of the region to contribute their desktops to the DG system. The Westminster DG is also a non-volunteer DG but this is a local DG working inside the University of

Westminster. Public volunteer DG systems can be realised as centralised DG systems having one centralised server or as decentralised DG systems where several DG servers are used and connected by a P2P network.

3 Core Technologies and Related Work

EGEE (*Enabling Grids for E-sciencE*) makes grids available to scientists and engineers, the second phase of the European Commission funded project (EGEE-II) has started in April 2006. The infrastructure is an ideal platform for any scientific research area, especially for high energy physics and life sciences whose computing demand is high. EGEE offers 40000 CPUs and about 5PB of storage space, with a throughput of around 100000 jobs a day.

EGEE is built on the gLite middleware, a middleware for building a grid that pulls together contributions from many other projects, including LCG and VDT. gLite supports different services, namely resource brokers, computing elements, storage elements, security services, information systems, worker nodes and user interfaces. All the services are deployed on a Scientific Linux installation. The basic building blocks of the gLite middleware are the *Worker Nodes* (WN). These machines are responsible for the actual execution of applications in gLite. Users can assume that their application is run within a well-dened environment when executing on a worker node. Worker nodes are similar to the nodes of a cluster and a group of worker nodes is attached to a *Computing Element* (CE). Computing elements provide a gateway to the worker nodes and therefore essentially CEs provide the grid resources.

Condor [9] also allows EGEE resources to temporarily join a Condor pool using the CondorG [10] mechanism. This works by submitting Condor itself to a grid resource and then this Condor instance can run jobs submitted to the original Condor pool on the EGEE resource. However this has to be configured manually and cannot be done automatically when the number of jobs would justify it. Also, if there are currently not enough jobs in the Condor pool to utilise the grid resource, then it may be wasted.

Another approach followed in [11] is to configure a DG Client (in this case an XtremWeb client) and submit this job to a Condor managed resource pool. Whenever there are unused resources available, Condor starts the backfill job (in this case a desktop grid client) on the available computers. This approach has disadvantages: first, it requires explicit support from the local job scheduler. Second, the administrator of the EGEE computing element must statically configure the desktop grid client, meaning this solution is not available for regular EGEE users. So this approach helps computing element administrators who want to increase the utilisation of their resources, but it does not help regular users who already have desktop grid applications and want to use more resources for them.

Desktop grids on the other hand are grid systems consisting of desktop machines. A common architecture followed by most of Desktop grids systems such as BOINC[12] or XtremWeb, is to generally maintain a single central service and allow users to offer their computers' CPU cycles for free. As such, they are re-

ferred to as volunteer computing and generally the more exciting a desktop grid's problem is, the more users will volunteer.

A user machines can join a desktop grid by installing a client-side application, which communicates with the DG server, sending its machine specifications (OS, CPU, performances) and a request for work. The DG server replies by sending application executables and the requested work. The client processes the downloaded data and upon completion uploads the results back to the server and then requests more work. With BOINC, users may participate to several projects and therefore the client may contact several BOINC servers.

The DG server is the key part of any desktop grid: it stores applications and tasks, ensures the scheduling of tasks to resources, provides the necessary fault-tolerance mechanisms and performs results verification.

In the case of volunteer computing project such as BOINC, it gives users information about work achieved and rewards given for their contribution in the form of credits. BOINC server administrators also operate user forums related to the project where users can ask questions and report their problems. In addition, XtremWeb allow certain classes of users to submit new tasks and application in the system. Both BOINC and XtremWeb servers rely on web server (Apache) for the project users and data transfer and use a relational database (MySQL) for for storage of applications, their related tasks, and client and user information.

4 SG and DG Bridging Technologies

EDGeS is attempting to close the gap between DG and SG computing. In particular, we would like to run SG jobs in a DG and vice versa in a seamless way. The bridge between a SG and a DG must work in either direction, but the different directions have different issues and requirements and therefore they need different solutions. A SG→DG bridge means that jobs submitted to a SG system (for example the EGEE) should be executed using DG resources while a DG→SG bridge allows resources from a SG to be used in a DG.

4.1 The SG-DG Bridge

Creating the connection between EGEE grids and Desktop Grids will enable the interoperability of EGEE and DG systems. Jobs originating from EGEE should be allowed to run on Desktop Grids, and Desktop Grids should be able to use EGEE Computing Elements as donors within a Desktop Grid project. Right now, Desktop Grids running BOINC or XtremWeb can only use traditional donors, and no other valuable computing power, like those EGEE provides. On the other hand, as DG systems are very easy to set up and maintain, using them in EGEE adds notable computing power to already existing VOs.

Bridging from DGs to EGEE The DG to EGEE bridging can be achieved by creating a modified version of the Desktop Grid client software that represents itself as a very powerful computer (with hundreds or thousands of processors)

toward the desktop grid server, figure 3. The modified client does not run the work units received from the desktop grid server itself but instead launches a wrapper application that transfers the input data and executables to the EGEE system and executes the job on an EGEE resource, using the APIs provided by the EGEE middleware. The output of the job is also collected by this wrapper application and then sent back to the Desktop Grid server.

The difficulty in this approach is to harmonise the internal scheduler of the desktop grid client software with the EGEE job schedulers. The internal scheduler decides how much work the desktop grid client asks from the server. In order to make a good decision, the internal scheduler has to know how loaded the EGEE resources are and must dynamically adapt itself to the level of resources available. This will require implementing more advanced scheduling strategies in the desktop grid client than currently available.

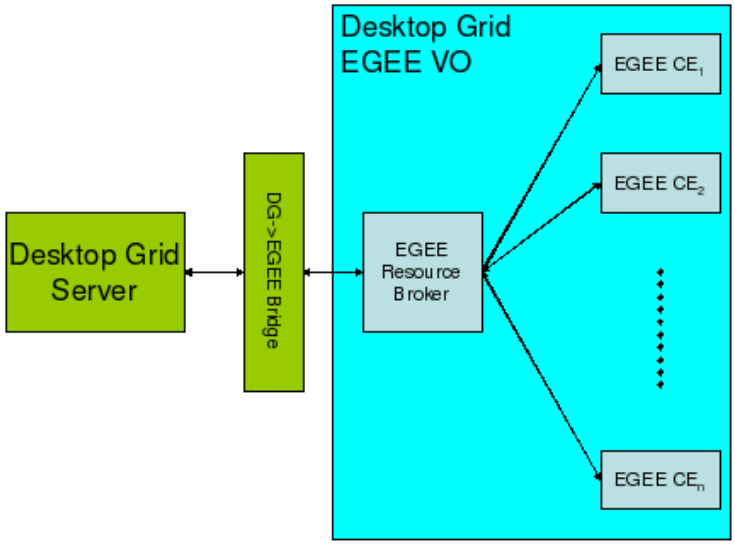

Fig. 3. Proposed structure for the DG → EGEE bridge.

The security aspects also have to be observed. Jobs arriving from the DG system do not have secure proxy certificates that the EGEE middleware expects. Therefore, the modified DG client must have its own certificate and it must use this certificate when submitting the jobs to EGEE to identify DG jobs. The lifetime of proxies used for job submission in case of long-running applications is another important question. The bridge should be able to use the proxy renewal mechanism present in EGEE.

On the resource provider side, we expect that not all resources want to run jobs arriving from DGs. Therefore, solutions such as setting up new virtual organisations (VO) will be investigated as a means of providing a way to differentiate

the jobs and allow the service providers control over what is run and where. Jobs arriving from desktop grids will then be sent only to resources that are part of this desktop grid VO.

Bridging from EGEE grids to Desktop Grids EGEE users require transparent access to DG resources. They want to get information about the DG, submit jobs to the DG, and get job status information and results back from the DG using EGEE tools. Users should also be able to run jobs that make full use of the EGEE infrastructure, for example accessing fies located on EGEE Storage Elements (SE). In order to achieve this, the DG must behave like an EGEE Computing Element (CE) belonging to the VO the user wants to use, figure 4.Let us overview the most important aspects of achieving this goal.

In order to make the bridge capable of transferring jobs to the DG, the bridge must provide a GRAM interface. Using this interface, the EGEE VOs Resource Broker (RB) can talk to the DG.

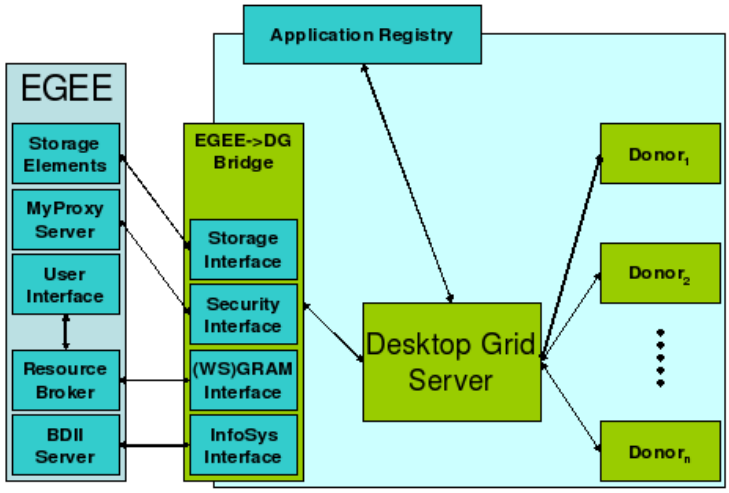

Fig. 4. Planned structure for the EGEE → DG bridge.

Every job submitted from EGEE to DG will generate a single work unit. This ensures the same behaviour for the DG resource that is expected from an EGEE CE. Direct mapping between the EGEE job and the Desktop Grid work unit allows verifying that the submitted job has all parameters set for the execution. Input fies need to be retrieved, if they are not stored locally. These fies are then mapped to the input fies of the Desktop Grid application and a ready-to-submit work unit is created.

Since the DG clients are outside EGEE they have no access to Storage Elements. Therefore, remote fies required for work unit creation must be retrieved before execution. Work units are created on the DG server. Before the executable is sent to clients, the bridge must ensure that all input fies are present on the DG server.

Security is a key challenge in this case as well. DGs are typically single user systems and do not apply user certificates for authentication and authorisation. This means, that some kind of mapping from EGEE user certificates to DG projects needs to be implemented.

4.2 Application Development and User Access

Once the bridge between SG and DG systems is created we have to assure that applications can be efficiently developed and run on this hybrid platform. Running an application on a Service Grid, like the EGEE, usually does not require specific modification of the executable. Of course, the more efficient execution of the code may require more sophisticated alterations like creating an MPI parallel version of the application, or using standard APIs to access lower level Grid services. Existing efforts for providing a uniform API for SG systems include the Grid Application Toolkit (GAT) developed by the European GridLab project, and the DRMAA and SAGA APIs defied by the OGF. However, all of these efforts are modelled after the traditional batch system paradigm of SG systems which is not adequate for DG platforms and for the combined SG-DG infrastructure.

Concerning the target DG solutions, XtremWeb offers the capability to run any application directly on the DG infrastructure. Unfortunately, life is more complicated in case BOINC-based Desktop Grid systems. The application has to be significantly modified in order to be run on this infrastructure. An existing solution to overcome this shortcoming is the Distributed Computing API (DC-API) [6] that hides the idiosyncrasies of BOINC and other DG systems, allowing developers to focus on their own applications and/or research tasks. The EDGeS project extends and further develops these APIs in order to support the porting of applications to the SG-DG platform.

Another aspect of application development is the creation of computational workows. There are several tools supporting the creation of computational Grid workows for SG systems, like Triana, Taverna, or the P-GRADE workow engine. However, there is no workow support provided for DG systems at the moment. To overcome this, the P-GRADE portal workow engine and user interface are being extended towards Desktop Grids. The solution enables creating computational workows, even parameter sweep workows, where workow components are either mapped to SG systems, to DG systems, or to any combination of these. The extended P-GRADE portal will provide a user-friendly, graphical environment where users can not only easily develop, but can also execute applications on both SG and DG systems. This environment hides the difference between SGs and DGs, leaving it to the underlying grid broker/scheduler to fid the most suitable service and/or desktop grid resources for execution. The P-GRADE portal, which is already deployed in several EGEE VOs, will also be interfaced with BOINC

and XtremWeb based DG systems. EGEE users can still access the infrastructure through the portal, as they can access at the moment. However, in the background, the portal will also utilise the DG infrastructure for suitable tasks. DG users will not only be provided with a currently missing user interface to utilise the DG infrastructure, but their applications can also use EGEE resources. finally, applications, which were neither specifically written for the EGEE nor for the DG infrastructure, will be orchestrated by the portal to utilise the combined resources in the most suitable manner.

4.3 SG-DG Data Access

One key component to EDGeS is the ability to satisfactorily handle the data requirements that arise when transferring jobs between service and desktop grids. The easiest solution to this problem would be to directly expose the service grid data layer to the desktop grid environment. This would closely mimic the functionality that is currently employed by most BOINC projects, where data is centrally distributed to all Desktop Grid participants through a set of known, trusted, and centralised servers. This simple solution, however, has many potentially limiting drawbacks that make it an unattractive solution for EDGeS, for example: Service Grids might not be able to cope with the increased bandwidth requirements imposed by this solution; there are significant security implications in exposing these data systems to direct outside semi-anonymous access; and, unlike traditional BOINC projects, which are relatively static in their data inputs and code requirements, the jobs being migrated by EDGeS would be dynamic need-based transfers that would rely on an underlying system that can dynamically build, expose, and propagate data to network participants.

With these ideas in mind, the EDGeS project will be working to build Peer-to-Peer data sharing mechanisms for data propagation. When considering applying P2P data access technologies to the scientific application domain, two broad challenge areas must be addressed: *social acceptability and technological challenges*. Socially, Peer-to-Peer technologies, especially when used for sharing data, are often viewed with a sceptical eye, having been long associated with widespread file sharing of copyrighted material. Additionally, there is also substantial concern that mixing Peer-to-Peer with volunteer computing could, in the event of malicious attacks on the network, cause irreparable damage to the volunteers' trust in the network and thereby adversely effect their willingness to continue donating resources. During the EDGeS project, these social concerns are ongoing and take on a very important role during the current design process, in which we are seeking to identify solutions that not only move forward Desktop Grid utilisation, but also to introduce Peer-to-Peer networks and P2P file sharing as both valid and legitimate options for scientific computing.

Within the technical area, security and scalability are the main issues that are being considered. Scalability for large P2P networks has evolved into two general categories: Distributed Hash Tables (DHTs) and super-peer topologies. Both of these approaches are valid and have their unique advantages and disadvantages depending on the problem one is trying to solve, generally with a trade-off between

speed, accuracy, and exibility - finding the correct balance for each individual situation is the important factor. With this in mind, scalability research in EDGeS is focusing on designing an adaptable network that can automatically change its topology to optimally balance network load, an especially useful trait in the case of super-peer technologies, where effective algorithms can help promote an efficient and scalable design.

Security is a much larger issue. Due to the sensitive and vulnerable nature of Desktop Grids, it is critical that not only are peer nodes secure from malicious attacks, but also that data integrity and reliability is ensured. The easiest solution, and perhaps the most susceptible to attacks is a pure P2P network, in which any node is allowed to receive and share information with any other node on the network. This allows for the most network exibility and client resource usability, however, since in this scenario any node has the capability to promote information, it also has the ability to ood the network with false information. Even though safeguards and hashing can be put in place to mitigate these effects, there is still the potential for malicious network utilisation. In a more restricted network, where only "trusted" peers are allowed to act as data cachers and rendezvous nodes the probability that this will happen is diminished, however usability and exibility are reduced as a result.

The EDGeS project is currently working to pursue a balance between a free forming and a restricted network. Current security infrastructure is being based upon the idea of secure super-peer data centres. In this type of system, every network peer is allowed to receive data, however, only those that meet certain security restricts are allowed to propagate the data to other network participants. Although these security constraints could be based upon any number of configurable factors, in its initial iteration, we envision it to be something as simple as a dynamic set of trusted peers that are identified through being signed by a common X509 root certificate. In future iterations of the security infrastructure, the feasibility of more interesting and file-tuned scenarios will be investigated, such as making use of a users' "BOINC credit" standing to certify them as a "trusted party" that can safely relay messages and store data.

EDGeSs data access research is broken down into three distinct tasks that, when completed, should provide a complete data access solution for the EGEE-DG Bridge as well as a useful data access layer for generic Desktop Grids. The tasks involved are as follows:

- data migration from Service Grids to Desktop Grids;
- data distribution in Desktop Grids; and,
- data access inside Desktop Grids.

5 Conclusion

The EDGeS project starts in January 2008 and the work presented here describes the main research themes for the project for enabling bridging technologies between service and desktop Grids. The main issues discussed in this paper include

security and bridging techniques for translating SG primitives into their DG counterparts and vice versa, as well as proposed distributed data access and scalability solutions. We have also discussed application and user access across the heterogeneous infrastructures that will be elaborated by the project. The duration of the EDGeS project is two years and it will end in December 2009. It is funded through the Research Infrastructures initiative of the 7th Framework Programme of the European Commission, grant agreement Number 211727.

References

1. D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. Seti@home: an experiment in public-resource computing. Commun. ACM, 45(11):5661, 2002.
2. OSG Executive Board. The open science grid. Submitted to SciDAC PI Meeting, 2007.
3. The Teragrid Project. http://www.teragrid.org.
4. Oxgrid, a campus grid for the university of oxford. http://www.ict.ox.ac.uk/strategy/events/wallom/.
5. Einstein@Home Project. http://www.physics2005.org/events/einsteinathome/.
6. Z. Balaton, G. Gombas, P. Kacsuk, A. Kornafeld, J. Kovacs, C. A. Marosi, G. Vida, N. Podhorszki, and T. Kiss. Sztaki desktop grid: a modular and scalable way of building large computing grids. In Proc. of the 21th International Parallel and Distributed Processing Symposium, 26-30 March 2007, Long Beach, California, USA. IEEE, 2007.
7. A. Emmen. Almeregrid, the worlds fist city supercomputer, is taking shape, October 2004. http://www.hoise.com/primeur/04/articles/monthly/AE-PR-10-04-69.html.
8. Gilles Fedak, Cécile Germain, Vincent Néri and Franck Cappello. XtremWeb: A Generic Global Computing Platform. Proceedings of CCGRID'2001 Special Session Global Computing on Personal Devices, 2001, Brisbane, Australia, May, IEEE/ACM, IEEE Press.
9. M. Litzkow, M. Livny, and M. Mutka. Condor: A Hunter of Idle Workstations. In 104-111, editor, Proceedings of 8th International Conference of Distributed Computing Systems, June 1988.
10. D. Thain, T. Tannenbaum, and M. Livny. Distributed Computing in Practice: The Condor Experience. Concurrency and Computation: Practice and Experience, 17, 2005.
11. Oleg Lodygensky, Gilles Fedak, Franck Cappello, Vincent Neri, Miron Livny and Douglas Thain. XtremWeb & Condor: Sharing resources between Internet connected Condor pools, Proceedings of CCGRID'2003, Special Session Global Computing on Personal Devices, 2003, Tokyo, Japan, IEEE/ACM
12. D. P. Anderson. Boinc: A system for public-resource computing and storage. In R. Buyya, editor, GRID, pages 410. IEEE Computer Society, 2004.