EDGeS: a bridge between desktop grids and service grids.

Fedak, Gilles[1], Lodygensky, Oleg[2], Balaton, Zoltan[3] Farkas, Zoltan[3] Gombas, Gabor[3] Kacsuk, Peter K.[3] Lovas, Robert[3] Marosi, Attila Csaba[3], Kelley, Ian[4], Taylor, Ian[4], Terstyanszky, Gabor[5], Kiss, Tamas[5], Cardenas-Montes, Miguel[6], Emmen, Ed[7] and Araujo, Filipe[8]

[1] Haiwu HE, INRIA,LRI, University Paris-Sud XI
[2] LAL University Paris-Sud XI
[3] MTA SZTAKI, Budapest, Hungary
[4] Cardiff University
[5] School of Informatics, University of Westminster
[6] Extermadura Advanced Research,Center (CETA-CIEMAT)
[7] AlmereGrid, Almere,Netherlands
[8] University of Coimbra, Coimbra, Portugal

EDGeS:A Bridge Between Desktop Grids and Service Grids

Gilles Fedak, Haiwu HE
INRIA,LRI, University Paris-Sud XI,Orsay,France
fedak@lri.fr,haiwu.he@inria.fr

Oleg Lodygensky
LAL University Paris-Sud XI, CNRS, IN2P3, France,
lodygens@lal.in2p3.fr

Zoltan Balaton, Zoltan Farkas, Gabor Gombas,Peter Kacsuk, Robert Lovas, Attila Csaba Marosi
MTA SZTAKI, Budapest, Hungary
{balaton, zfarkas, gombasg, kacsuk, rlovas, atisu}@sztaki.hu

Ian Kelley, Ian Taylor
Cardiff University, Cardiff, UK,
{I.R.Kelley, Ian.J.Taylor}@cs.cardiff.ac.uk

Gabor Terstyanszky, Tamas Kiss
University of Westminster, London, UK,
{G.Z.Terstyanszky, T.Kiss}@westminster.ac.uk

Miguel Cardenas-Montes
Extermadura Advanced Research,Center (CETA-CIEMAT), Trujillo, Spain
miguel.cardenas@ciemat.es

Ad Emmen
AlmereGrid, Almere,Netherlands
ad@almeregrid.nl

Filipe Araujo
University of Coimbra, Coimbra, Portugal,
filipius@dei.uc.pt

Abstract

Desktop grids and service grids widely used by their different users communities as efficient solutions for making full use of computing power and achieving loads balances across Intranet or Internet. Nevertheless,little work has been done to combine these two girds technologies together to establish a seamless and vast grid resources pool. In this paper we will present a new European FP7 infrastructure project:EDGeS(Enabling Desktop Grids for e-Science), which aim to build technological bridges to facilitate interoperability between desktop grid and service grid. We give also a taxonomy of existing grid systems: desktop grids such as BONIC and XtremWeb, service grids such as EGEE. Then we describe furtherly our solution for identifying translation technologies for porting applications between desktop grids and service grids, and vice versa. There are three themes in our solution, which discuss actual popular bridging technologies, user access issues, and distributed data issues about deployment and application development.

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

978-0-7695-3306-3/08 $25.00 © 2008 IEEE
DOI 10.1109/ChinaGrid.2008.44

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 Organization (VO) or community authorization services that centralize 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 utilization 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 Grid–Desktop Grid (SG–DG) infrastructure (see figure 1).

In this paper, we described 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. The formulation of these questions and research collaboration to answer them has already been started within the CoreGrid Institute on Architectural Issues. More recently a new European project, called EDGeS (Enabling Desktop Grids for e-Science) has been accepted by the European Commission in order to build this architecture and provide it as a service for the European research community. This paper gives an overview on the research perspectives and proposed solutions within EDGeS. 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 2. In Section 3, 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.

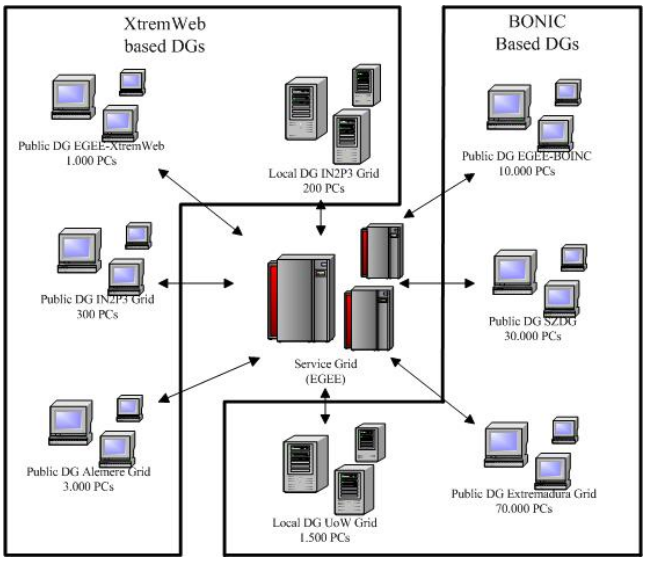

Figure 1. Structure of new e-infrasture

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 requester 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). 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.

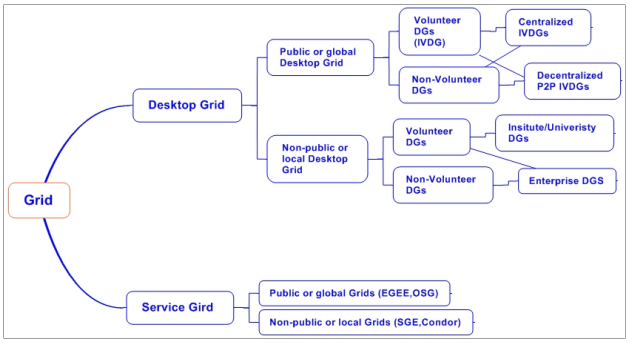

Figure 2. Taxonomy of grid systems from the Desktop Grid point of view.

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 realized as centralized DG systems having one centralized server or as decentralized DG systems where several DG servers are used and connected by a P2P network. All the previously mentioned DG systems are centralized DGs. An example for a P2P DG system is the OurGrid DG infrastructure from Brazil [9].

In the next section, we introduce the most important technologies that form the basis of the integrated SG–DG platform.

3 Core Technologies and Related Work

EGEE (***E**nabling **G**rids 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-defined 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 [10] also allows EGEE resources to temporarily join a Condor pool using the Condor Glidein [11] 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 utilize the grid resource, then it may be wasted.

Another approach followed in [12] is to configure a DG Client (in this case an XtremWeb client) as a backfill job for Condor. 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 utilization 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 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 referred 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 achived 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 SGs and DGs will enable the interoperability of EGEE and Volunteer Computing 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 in two ways. The first approach is 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. The modified client does not run the work units received from the desktop grid server itself but instead transfers the input data and executables to an EGEE VO and executes the job on an EGEE resource, using the APIs provided by the EGEE gLite middleware. This is most easily realized by launching a wrapper in place of the real application that does the conversion and job submission and acts as a proxy for the real application. The output of the job is also collected by this wrapper and then sent back to the Desktop Grid server. Figure 3 depicts a prototype version of this solution.

This approach is very similar to the Cluster Client developed by SZTAKI to utilize Condor clusters in a BOINC based DG which is described in [13]. The advantage of this solution is that submission is done via the interface provided by the SG middleware, meaning it is using a well defined interface and not relying on the internal structure of the SG. Basically the SG is a black-box for the client and the WNs do not need to have in-/outbound network connection or any direct communication with the Bridge or the Desktop Grid server.

The difficulty in this approach is to harmonize the internal scheduler of the desktop grid client with the EGEE gLite WMS. The internal scheduler of the client decides how much work the desktop grid client asks from the server.

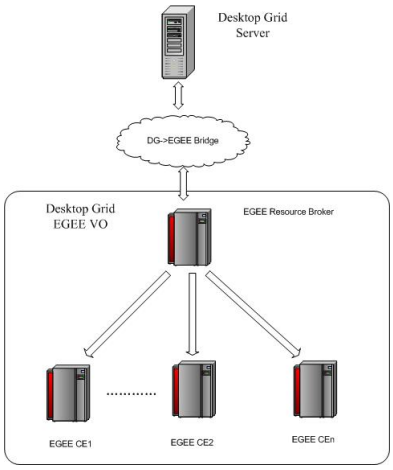

Figure 3. Structure of the DG⇒EGEE bridge.

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.

The second approach is to build an "overlay DG" on top of EGEE resources by submitting agents wrapped as grid jobs. This solution was pioneered by Condor Glide-in and is also prototyped in XtremWeb. The agents are desktop grid clients configured appropriately, so after landing at a WN and being started by the SG middleware they connect to the Desktop Grid server to obtain work and start processing it as a normal client. The agents are continue getting work from the DG and processing it as long as their time, allocated by the SG middleware, is up or there is no more work to get. The purpose of the Bridge in this solution is not to convert DG work units to SG jobs but to keep the overlay DG running by submitting and managing the agents. The advantage of this approach is that it can be easily implemented using existing components without modifying the internal scheduler of the DG client but the drawback is that the SG middleware is not utilized as intended but rather circumvented. This can be a problem e.g. for sites using internal network for WNs that do not allow network communication.

The security aspects also have to be observed for both approaches. Jobs arriving from the DG system do not have secure proxy certificates that the EGEE middleware expects. Therefore, the Bridge 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 organizations (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 files 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. 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 VO's Resource Broker (RB) can talk to the DG.

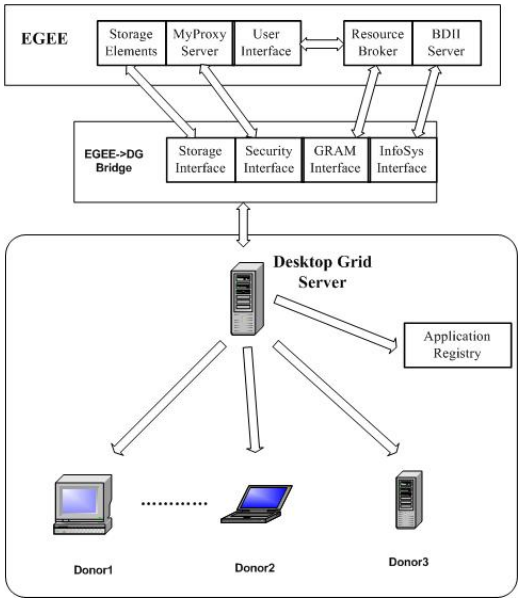

Figure 4. Structure of the EGEE⇒DG bridge.

Every job submitted from EGEE to DG will generate a single work unit. This ensures the same behavior 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 files need to be retrieved, if not stored locally. These files are then mapped to the input

files 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 files 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 files 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 authorization. This means, that some kind of mapping from EGEE user certificates to DG projects needs to be implemented.

4.2 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 centralized 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, through its JRA3 Data Access activity, 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 skeptical 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 utilization, 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 flexibility — 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 flexibility and client resource usability, however, since in this scenario any node has the capability to promote information, it also has the ability to flood 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 utilization. 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 flexibility 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 centers. 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 fine-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.

EDGeS's 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: *(i)* data migration from Service Grids to Desktop Grids; *(ii)* data distribution in Desktop Grids; and, *(iii)* data access inside Desktop Grids.

5 Conclusion

The EDGeS project started in January 2008 but collaboration between consortium partners started earlier within the Scalability for Desktop Grids Research Group of the CoreGrid Institute on Architectural Issues. The work presented here describes the main research themes for the project for enabling bridging technologies between service and desktop Grids that were identified during collaboration within CoreGrid. 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. The bridging solutions discussed in this paper were already prototyped to evaluate their advantages and disadvantages of possible approaches in order to select the ones that will be elaborated by the project. These prototypes and preliminary results of the evaluation were also presented. The duration of the EDGeS project is two years and it will end in December 2009. However, by not starting from scratch but basing our work on results achieved earlier and existing collaboration induced by CoreGrid we are confident that we can meet the ambitious goals in this short timeframe.

Acknowledgments

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] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, 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] Zoltan Balaton, Gabor Gombas, Peter Kacsuk, Adam Kornafeld, Jozsef Kovacs, Attila Csaba Marosi, Gabor Vida, Norbert Podhorszki, and Tamas 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] Ad Emmen. Almeregrid, the world's first 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. In *Proceedings of CCGRID'2001 Special Session Global Computing on Personal Devices*, Brisbane, Australia, May 2001. IEEE/ACM, IEEE Press.

[9] N. Andrade, L. Costa, G. Germoglio, and W. Cirne. Peer-to-peer grid computing with the OurGrid Community. In *Proc. of the 23rd Brazilian Symposium on Computer Networks*, 2005.

[10] 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.

[11] D. Thain, T. Tannenbaum, and M. Livny. Distributed Computing in Practice: The Condor Experience. *Concurrency and Computation: Practice and Experience*, 17, 2005.

[12] Oleg Lodygensky, Gilles Fedak, Franck Cappello, Vincent Neri, Miron Livny, and Douglas Thain. Xtremweb & condor : Sharing resources between internet connected condor pools. In *Proceedings of CCGRID'2003 Special Session Global Computing on Personal Devices*, number 2003, Tokyo, Japan. IEEE/ACM.

[13] Attila Csaba Marosi, Zoltan Balaton, Peter Kacsuk, and Daniel Drotos. SZTAKI Desktop Grid: Adapt Clusters for Desktop Grids. Technical report, MTA SZTAKI, 2008. Submitted to the 3rd International Workshop on Distributed Cooperative Laboratories: Instrumenting the Grid (INGRID 2008), Lacco Ameno, Island of Ischia, Italy.