Towards a Security Model to Bridge Internet Desktop Grids and Service Grids

Gabriel Caillat[1], Oleg Lodygensky[1], Etienne Urbah[1],
Gilles Fedak[2], and Haiwu He[2]

[1] Laboratoire de l'Accelerateur Lineaire,
Universite Paris Sud Bat 200 , 91 Orsay, France
{gcaillat,lodygens,urbah}@lal.in2p3.fr
[2] INRIA Saclay, LRI,
Universite Paris Sud Bat 490 , 91 Orsay, France
{fedak,haiwu.he}@lri.fr

Abstract. Desktop grids, such as XtremWeb and BOINC, and service grids, such as EGEE, are two different approaches for science communities to gather huge computing power from a large number of computing resources. Nevertheless, little work has been done to combine these two Grids technologies in order to establish a seamless and vast grid resource pool. In this paper we address the security issues when bridging Service Grid with Desktop Grid. We first present how to bridge EGEE resources with our XtremWeb platform using the gliding-in mechanism. Then we describe a new Desktop Grid security model to bridge this anonymous environment to the strongly securized Service Grid one. Finally we describe an implementation of this security model in the XtremWeb middleware and report on performance evaluation.

Keywords: Desktop Grid, Service Grid, BOINC, XtremWeb, Security.

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. Researchers and developers in Service Grids *(SG)* 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

E. César et al. (Eds.): Euro-Par 2008 Workshops, LNCS 5415, pp. 247–259, 2009.

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 centralizes the management of trust relationships and access rights.

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 [ACK+02] and Folding@home. They are very attractive, as "Internet Computing" platforms, for scientific projects seeking a huge amount of computational resources 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 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 [And04] or XtremWeb [FGNC01], 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 DG resources for computation. For instance the BOINC project features less than 50 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 mechanism such as sandbox or result certification to prevent attacks from other users. As a consequence, DG systems are not yet ready to be integrated in complex Grid infrastructure which requires a high level user right management, authentication, authorization and rights delegation.

Until now, these two kinds of Grid systems are completely separated and there is no way to 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 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. Our research are part of the work conducted by a new European FP7 infrastructure project : EDGeS (Enabling Desktop Grids for e-Science) [CMEM+08], which aims to build technological bridges to facilitate interoperability between DG and SG.

In this paper, we describe the research issues on how such an integrated SG–DG infrastructure can be established from a security point of view. We first review the security model of existing approaches to bridge the two classes of Grid systems. Our technical contribution is two folds. First, we propose a new users classification into Desktop Grids which allow to manage anonymous DG users and SG users in a global SG-DG infrastructure. We detail expected modifications in our XtremWeb middleware. Second, we present a bridge between the EGEE

grid and XtremWeb based on the *gliding in* solution. We show that this solution provides high security level, fault-tolerance, performance and scalability.

In the next section, we give an overview of the Service Grid and the Desktop Grid security model. In Section 3, we describe the existing solution to bridge Service Grid and Desktop Grid. In Section 4, we present our security architecture and its implementation within the XtremWeb Desktop Grid and the integration in the EGEE Grid. Performance evaluation is shown in Section5. In Section 6, we present our concluding remarks.

2 Background

2.1 Security Model of Desktop Grids

In this section we review the security model of several Desktop Grid systems.

The BOINC [And04] middleware is a popular Volunteer Computing System which permits to aggregate huge computing power from thousands of Internet resources. A key points is the asymmetry of its security model : there are few projects well identified and which belongs to established institutions (Univ. of Berkeley, Univ of Haifa...) while volunteers are numerous and anonymous. The notion of users exists in BOINC, which aims to manage volunteers contributions. However, this user definition is close to the one of avatar : it allows users to participate to forum and to receive credits according to the computing time and power given to the project.

Despite anonymity, the security model is based on trust. Volunteers trust the project they are contributing to. Security mechanism is simple and based on asymmetric cryptography.

Security model aims at enforcing the trust between volunteers and the project itself. At installation time, the project owners produce a pair of public/private keys and store them in a safe place, typically in a machine isolated from the network, as recommended on the BOINC web site. When volunteers contribute for the first time to the project, they obtain the public key of the project. Project owners have to digitally sign the project application files, so that volunteers can verify that the binary codes downloaded by the BOINC client really belongs to the project. This mechanism ensures that, if a pirate get access to one of the BOINC server, he would not be able to upload malicious code to hundreds of thousands resources. If volunteers trust the projects, the reverse is not true. To protect against malicious users, BOINC implements a result certification mechanism [Sar02], based on redundant computation. BOINC gives the ability to project administrator to write their own custom results certifying code according to their application.

XtremWeb is an Internet Desktop Grid middleware which also permits public resources computing. It differs from BOINC by the ability given to every participants to submit new application and tasks in the system. XtremWeb is a P2P system in the sense that every participants can provide computing resources but also utilize others participants' computing resources. XtremWeb is organized as a three-tiers architecture where clients consumes resources, worker provides

resources and coordinator is a central agent which manages the system by performing the scheduling and fault-tolerance of tasks. XtremWeb implements the notion of user, used to facilitate the platform management and to separate between users' tasks, results and applications. In contrast with BOINC, because everyone can submit application, there cannot be any form of trust between users, application, results and even the coordinator itself. Thus XtremWeb security model is based on autonomous mechanisms which aims at protecting each components of the platform from the others elements. For instance, to protect volunteers' computer from malicious code, a sandbox mechanism is used to isolate and monitor the running application, and prevent it to damage volunteers system. Public/private keys mechanism are also used to authenticate the coordinator to prevent results to be uploaded to an other coordinator.

The Xgrid system, proposed by Apple is a Desktop Grid designed to run on a local network environment. Xgrid features ease of use and ease of deployment. To work, the Xgrid system needs a Xgrid server, which can be configured with or without password. If the server run without password, then every user in the local environment can submit jobs and application; otherwise a password is needed to do so. Computing nodes, in the Xgrid system can accept jobs or no, this property is set on the computing nodes itself. Thus there is no real distinction between users and there's no possibility for a user or a machine to accept or refuse other users' application or work. While this solution is acceptable when used within a single organization (lab or small company), this solution won't scale to a Service Grid setup which typically aims at several institutions to cooperate.

2.2 Security Model of Service Grids

The Grid Security Infrastructure (GSI) in EGEE enables secure authentication and communication over an open network. GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol, with extensions for single sign-on and delegation. In order to authenticate himself to Grid resources, a user needs to have a digital X.509 certificate issued by a Certification Authority (CA) trusted by EGEE; Grid resources are generally also issued with certificates to allow them to authenticate themselves to users and other services. The user certificate, whose private key is protected by a password, is used to generate and sign a temporary certificate, called a proxy certificate (or simply a proxy), which is used for the actual authentication to Grid services and does not need a password. As possession of a proxy certificate is a proof of identity, the file containing it must be readable only by the user, and a proxy has, by default, a short lifetime (typically 12 hours) to reduce security risks if it is stolen. A user needs a valid proxy to submit jobs; those jobs carry their own copies of the proxy to be able to authenticate with Grid services as they run (so that the job can access user data, for example). For long-running jobs, the job proxy may expire before the job has finished, causing the job to fail. To avoid this, there is a proxy renewal mechanism to keep the job proxy valid for as long as needed. In terms of security a proxy is a compromise. Since the private key is sent with it anyone who steals it can impersonate the owner, so proxies need

to be treated carefully. Also there is no mechanism for revoking proxies, so in general even if someone knows that one has been stolen there is little they can do to stop it being used. On the other hand, proxies usually have a lifetime of only a few hours so the potential damage is fairly limited.

Grid security is based on the concept of public key encryption. Each entity (user, server...) has a private key which must be kept totally secure.

Each private key has its associated public key; they are referred as asymmetric keys. Any encryption using one key can be decrypted using the associated one. This mechanism is used to prove identity, to encrypt data and to check their integrity.

Certificates are issued by a Certification Authority (CA) which has its own certificate.

To check the validity of a certificate, the public key of the CA is then needed. Potentially this could create an infinite regression, but this is prevented by the fact that CA certificates, known as root certificates, are self-signed, i.e. the CA signs its own certificate.

A system called VOMS (VO Management Service) is used in EGEE to manage information about the roles and privileges of users within a VO. This information is presented to services via an extension to the proxy. At the time the proxy is created one or more VOMS servers are contacted, and they return a mini certificate known as an Attribute Certificate (AC) which is signed by the VO and contains information about group membership and any associated roles within the VO.

To create a VOMS proxy the ACs are embedded in a standard proxy, which is signed with the private key of the parent certificate. Services can then decode the VOMS information and use it as required, e.g. a user may only be allowed to do something if he has a particular role from a specific VO. One consequence of this method is that VOMS attributes can only be used with a proxy, they cannot be attached to a CA-issued certificate.

3 Bridging Service Grids and Desktop Grids

There exists two main approaches to bridge SG and DG (see Fig. 1). In this section we present the principles of these two approaches and discuss them according to security perspective.

The superworker approach. The *superworker*, proposed by the Lattice [MBC08] project and the SZTAKI Desktop Grid [BGK+07], is a 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

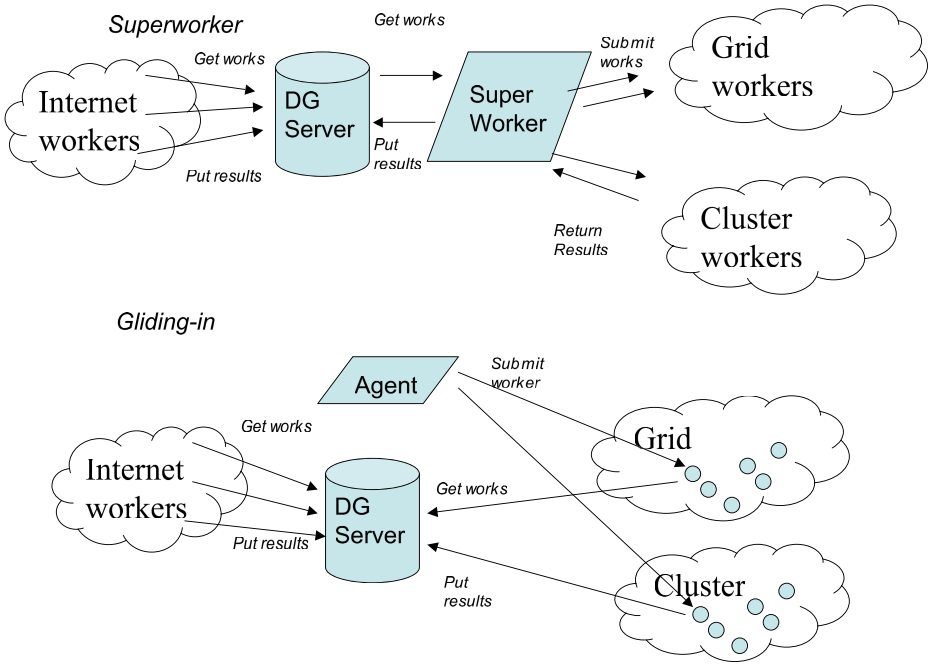

Fig. 1. Bridging Service Grid and Desktop Grid, the superworker approach versus the Gliding-in approach

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 : *i*) the superworker can become a bottleneck when the number of computing resources increases, *ii*) the round trip for a work unit is increased because it has to be marshalled/unmarshalled by the superworker, *iii*) it introduces a single point of failure in the system, which has low fault-tolerance. On the other hand, this centralized solution provides better security properties, concerning the integration with the Grid. First the superworker does not require modification of the infrastructure, it can be run under any user identity as long as the user has the right to submit jobs on Grid. Next, as works are wrapped by the superworker, they are run under the user identity, which conforms with the regular security usage, in contrast with the approach described in the following paragraph.

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 [LFC+03]. 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 communicates directly to the XtremWeb server. Similar mechanisms are now commonly employed in Grid Computing [TL04]. For example Dirac [TGSR04] 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 gatekeeper, 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 balance 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 [RZD+07] system shows, it gives better performances because series of jobs do not have to go through the gatekeeper queues which is generally characterized by long waiting time, and communication is direct between the worker running on the computing element (*CE*) and the DG server without intermediate agent such as the superworker. From the security point of view, this approach breaks the Grid security rule about Pilot Jobs. This rule does not allow actual jobs owner to be different than pilot job owner. This is a well known issue of pilot jobs and new solution such as *gLExec* [SKV+07] are proposed to circumvent this security hole.

4 Bridging XtremWeb to EGEE

Our goal is to safely aggregate EGEE worker nodes in a single XtremWeb network. In this network, we assume that the users connect to the dispatcher administration domain to submit tasks. XtremWeb has the responsibility to ensure user authentication, hosts (workers) integrity, application and results protection and user execution logging.

In the rest of the paper, we based our study of the security model on the gliding-in technology. XtremWeb Workers are submitted to EGEE computing elements using JSDL wrappers.

4.1 User Authentication and Execution Logging

The coordinator site manages a list of authorized users as ACLs. It is the responsibility of the system administrator to register new users (and revoke non desired ones) on the coordinator. After registration, the coordinator provides a key to be used by the user for each subsequent connection. For each connection, a challenge is ran in order to ensure that the user is registered on the coordinator. All communications between the user XW client and the coordinator are encrypted using SSL. Then the coordinator works as a proxy for the user: all tasks are submitted to the workers through the coordinator credential. All executions on the workers are logged in the security perspective: all tasks contain a descriptor with the actual user credential so that workers and coordinator can take appropriate corrective action (user revocation), in case of security problem.

The design does not currently rely on certificates and presents a certain degree of risk for "Man is the Middle" (*MIM*) attacks but risks are very limited since 1) attacks should origin from within EGEE subclusters only (due to TCP protocols), and 2) workers and clients software include coordinator public key, then if one is able to securely ensure worker and client binaries installation to dedicated pools, the full system is not subject to *MIM* attacks since key exchanges will not be necessary any more.

A certification system based on X.509 certificates is under integration in XtremWeb. Subsequent experiments and futures XtremWeb installations will implement one, based on Open-SSL, allowing extension of clients and workers authentication by the coordinator.

4.2 Applications, Parameters and Results Protection

EGEE subclusters belonging to different administration domains fetch applications and tasks, and store results on the central coordinator. The only security issue concerning applications, parameters and results transfers is then about the connections between EGEE worker nodes and the coordinator. To ensure connection security between domains, 1) every connection from any client and worker to the coordinator is encrypted through *SSL* tunnels; 2) workers can only connect to the coordinator for which they have the public key. These two mechanisms prevent malicious participants to be able to intercept and read any connection, to connect to the coordinator and EGEE worker nodes to connect to a wrong XtremWeb coordinator.

4.3 Node Integrity

If, for any reason, a malicious user succeeds on accessing the system and launching an aggressive application, XtremWeb workers still protect their host by implementing *sand-boxing*[AKS99, AR99, GWTB96] for binary applications. This is a secure way to execute applications, providing rights to do some actions and denying some others. One should note that Java applications are always executed inside a virtual machine which includes security[GMPS97]; XtremWeb uses this functionality in two levels, one for the worker itself and a more restrictive one for the downloaded Java byte code. On the contrary, binary (or *native*) applications have access to the full hosting system by nature; workers are configured to run any task of that type inside a sand-box which is fully customizable, from memory usage to file system operations.

Java and sand-boxes, have performance costs[BSPF01]; one can then disable this functionality on highly secured systems, such as clusters under a fully closed firewall.

4.4 Access Confinement

XtremWeb 2.0.0 introduces mechanisms aimed to secure and confine distributed resources usage; this is done thanks to the notion of user and access rights. These

new features permit to extend user actions over the platform as well as to secure resource usage and confine application deployment.

Access rights must be understood as linux file system ones (e.g. 0x755, uog+r etc.) and are used to define data (which is also a new paradigm in XtremWeb 2.0.0 but not discussed in this deliverable), application and job accesses. The default access is *0x755* which grants full access (read, write, execute) to owner, and limited access (read, execute) to users belonging to the owner group, as well as other users. Denying access to non group users, for example, consists to set access rights to *0x750*. The middleware includes the *xwchmod* command to modify access rights.

Any user can insert its own applications in the platform. This feature could lead to security hazards since this could allow users to insert malicious applications. This is solved by access rights. A *standard* user can only insert *private* applications; any submitted jobs referring private applications are private too. There is no way to modify this; even *xwchmod* cannot help to modify access rights of private entities (applications or jobs). A private entity has its access rights set to *0x700* which grants owner access only. A private entity will only be managed by private workers which are described below.

Inserting group (i.e. access rights *0x750*) entities needs advanced privileges. All users belonging to the group can access group entities. Submitting a job for a group application creates group jobs (i.e. access rights *0x750*). A group job can only be run by group workers which are described below.

Finally, inserting public (i.e. access rights *0x755*) entities needs advanced privileges too. All users can submit jobs for public applications. This creates public jobs (i.e. access rights *0x755*). A public job can only be run by public workers which are described below.

User rights, associated with access rights, permit to define public, groups and private levels, which grant allowed user actions. The three main level rights are *standard*, *worker* and *advanced*. The *standard* level grants data management, job submission and private application insertion as defined earlier. The *advanced* level grants full access to the platform including private, group and public entities, as well as users, user groups and workers management.

To understand the *worker* right level, one must understand that workers run using a registered identity. When a worker communicates with the coordinator, it presents its identity. This identity defines user rights, among others.

The *worker* level right defines *public* workers. This level right permits to delegate user rights to those public workers so that they can access entities as if they were the entity owner. A public worker can bypass entities access rights in order to update them even if their access rights do not allow that action. This is used to update jobs status to *COMPLETED* when it has successfully been computed, or to store jobs results and set results owner to the job owner. That last can then download its results. At installation time, the platform includes a specific user defined with worker level right, aiming to deploy public workers. Workers can also be group ones. Group workers are public ones restricted to a group. They use an identity belonging in a group, with *worker* user rights level.

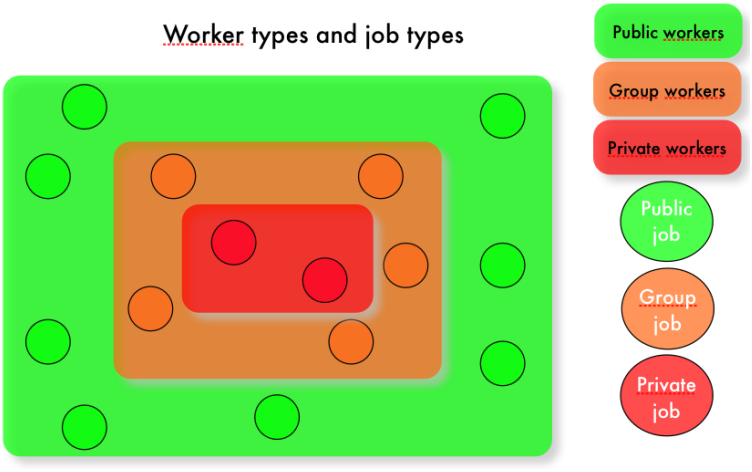

Fig. 2. Worker levels and jobs levels

Group workers will compute jobs submitted by any user of their group only. Finally, workers can be private. Private workers are identified whithout *worker* user rights level. They can only compute private jobs.

The Figure 2 summarizes worker levels and the types of job they can run. We see that private jobs are run on private workers only, groups jobs on group workers only, and public jobs on public workers only.

5 Performance Evaluation

To demonstrate the full system, we ran an application over our SG/DG platform. The application consists in a multi-parameters computation requiring a large set of independent tasks. We submitted 185 tasks XtremWeb aggregating volunteers resources and EGEE worker nodes.

Table 1 summarizes resource aggregated from XtremWeb and from EGEE.

Table 2 details these resource and shows four different types according to CPU speed.

Figure 3 shows a good distribution of the jobs among resources; differences comes from availability and resource types themselves. We can see that some resources have only computed one job; this is because the bridge configures

Table 1. Available resources per platform

	XtremWeb	EGEE	Total
Linux	1	10	11
Win32	1	0	1
Mac OS X	32	0	32
Total	34	10	44

Table 2. Types of available resources

OS	CPU	CPU speed	Total
Linux	AMD64	2.3GHz	11
Win32	ix86	2.0GHz	1
Mac OS X	ix86	2.0GHz	1
Mac OS X	ix86	2.0GHz	10
Mac OS X	PPC	1.0GHz	5
Mac OS X	PPC	1.5GHz	6
Mac OS X	PPC	2.0GHz	10
Total	3	4	44

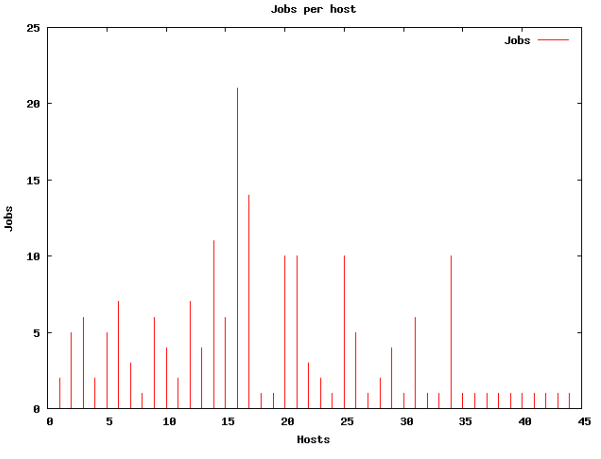

Fig. 3. Jobs per host in a SG/DG platform

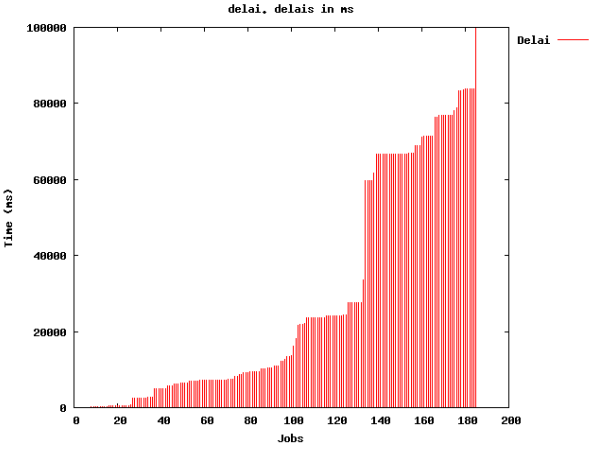

Fig. 4. Jobs execution delays

resources aggregated from SG to run a single job. This ensures that DG applications do not overload SG resources.

The impact of the resource heterogeneity is even more visible on figure 4 which presents the job execution time, sorted in increasing order.

6 Conclusion

Bridging Service Grids and Desktop Grids raises many issues. To enable actual infrastructure, gathering both Grid and Internet users, application, computing and storage resources requires new security model. In this paper we have reviewed the security requirements of DG and SG. The discussion about bridging technologies convince us to select the gliding-in solution even if it stresses the security requirements. Thus we have proposed a security model which distinguishes between *anonymous* users who are typically Internet volunteers and *certified* users who are the Grid users with a valid X.509 certificate delivered by the EGEE author. We have implemented this security model within the XtremWeb framework and showed that, when integrated with the EGEE infrastructure, this ensures a high security level for both the Grid and the volunteers' PC.

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

[ACK+02] Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: Seti@home: an experiment in public-resource computing. Commun. ACM 45(11), 56–61 (2002)

[AKS99] Alexandrov, A., Kmiec, P., Schauser, K.: Consh: a confinied execution environment for internet computations. In: Proceedings of the Usenix annual technical conference (1999), `http://www.usenix.org/events/usenix99/`

[And04] Anderson, D.: BOINC: A System for Public-Resource Computing and Storage. In: Proceedings of the 5th IEEE/ACM International GRID Workshop, Pittsburgh, USA (2004)

[AR99] Acharya, A., Raje, M.: Mapbox: using parameterized behavior classes to confine applications. Technical report TRCS99-25, University of California, Santa Barbara (1999)

[BGK+07] 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 21th International Parallel and Distributed Processing Symposium, Long Beach, California, USA, March 26-30 (2007)

[BSPF01] Bull, J.M., Smith, L.A., Pottage, L., Freeman, R.: Benchmarking java against c and fortran for scientific applications. In: ISCOPE Conference. LNCS, vol. 1343, pp. 97–105. Springer, Heidelberg (2001)

[CMEM+08] 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: Proc. of the 2nd Iberian Grid Infrastructure Conference, University of Porto, Portugal, May 12-14 (2008)

[FGNC01] Fedak, G., Germain, C., Néri, 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, Brisbane, Australia, May 2001, pp. 582–587. IEEE/ACM (2001)

[GMPS97] Gong, L., Muller, M., Prafullchandra, H., Schemers, R.: Going beyond the sandbox: an overview of the new security architecture in the java development kit 1.2. In: Usenix Symposium on Internet Technologies and Systems (1997)

[GWTB96] 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)

[LFC+03] 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 2003), Tokyo, Japan, pp. 382–389. IEEE/ACM (2003)

[MBC08] Myers, D.S., Bazinet, A.L., Cummings, M.P.: Grids for Bioinformatics and Computational Biology. In: Expanding the reach of Grid computing: combining Globus- and BOINC-based systems. Wiley Book Series on Parallel and Distributed Computing (2008)

[RZD+07] Raicu, I., Zhao, Y., Dumitrescu, C., Foster, I., Wilde, M.: Falkon: a fast and light-weight task execution framework. In: IEEE/ACM SuperComputing (2007)

[Sar02] Sarmenta, L.F.G.: Sabotage-Tolerance Mechanisms for Volunteer Computing Systems. Future Generation Computer Systems 18(4), 561–572 (2002)

[SKV+07] Sfiligoi, Koeroo, O., Venekamp, G., Yocum, D., Groep, D., Petravick, D.: Addressing the Pilot security problem with gLExec. Technical Report FERMILAB-PUB-07-483-CD, Fermi National Laboratory (2007)

[TGSR04] Tsaregorodtsev, A., Garonne, V., Stokes-Rees, I.: Dirac: A scalable lightweight architecture for high throughput computing. In: Fifth IEEE/ACM International Workshop on Grid Computing, GRID 2004 (2004)

[TL04] Thain, D., Livny, M.: Building reliable clients and services. In: Foster, I., Kesselman, C. (eds.) The Grid: Blueprint for a New Computing Infrastructure, 2nd edn. Morgan Kaufman, San Francisco (2004)