Extending the EGEE Grid with XtremWeb-HEP Desktop Grids

Haiwu He[1], Gilles Fedak[1], Peter Kacsuk[2], Zoltan Farkas[2], Zoltan Balaton[2],Oleg Lodygensky[3], Etienne Urbah[3],Gabriel Caillat[3], Filipe Araujo[4], Ad Emmen[5]

[1] INRIA, LIP, ENS de Lyon, 46 avenue d'Italie, 69364 LYON CEDEX 07, France
[2] MTA SZTAKI, LPDS, Kende u. 13-17,1111 Budapest, Hungary
[3] LAL,IN2P3/CNRS, University Paris-Sud 11, Bat 200, 91898, Orsay, France
[4] CISUC, Dept. of Informatics Engineering University of Coimbra, Portugal, [5] AlmereGrid, Netherlands

{haiwu.he, gilles.fedak}@inria.fr, {kacsuk, zfarkas, balaton}@sztaki.hu, {lodygens,urbah,gcaillat}@lal.in2p3.fr, flipius@dei.uc.pt, ad@almeregrid.nl

Abstract

Desktop Grids and Service Grids are widely used by scientific communities to execute high throughput application. The European EDGeS project aims at developing the technologies to bridge these two kinds of Grid technologies together.

In this paper, we present the development and the application of these new technology to extend the EGEE Grid with XtremWeb-HEP based Desktop Grids. We present the setup of the distributed infrastructure which enables EGEE users' jobs to run on one of the three various XtremWeb-HEP DGs. We describe the new volunteer computing project EGEE@Home based on XtremWeb-HEP middleware. To evaluate the capacity of our technology, we present a performance measurement of the main components and conclude that the overhead is kept reasonable despite the fact that the infrastructure is highly distributed.

1 Introduction

Various Grids have been created and run as a service for the scientific community such as EGEE[1] (***E****nabling* ***G****rids for* ***E****-scienc****E***) or ChinaGrid [1] . Researchers and developers in SGs (Service Grids) first create a Grid service that can be accessed by a large number of users. DGs (Desktop Grids) on the other hand are commonly known as *Public Computing* or *Volunteer Computing Systems* , because they often depend on the general public to contribute resources. i.e. "free CPU 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 widely spread, heterogeneous computing resources with ease. The European EDGeS (Enabling Desktop Grids for e-Science) project [2] aims at bridging the EGEE Grid with two main Desktop Grid middleware, BOINC [3] and XtremWeb [4]. One of the main achievement of EDGeS is the proposition of a 3G bridge (Generic Grid to Grid bridge).

In the paper, we will detail the extension of the 3G bridge to handle XtremWeb based Desktop Grid. We also describe the application of these new technologies to build a new infrastructure composed of three different XWHEP (XtremWeb-High Energy Physic) DGs. This new infrastructure appear as a CE (Computing Element in EGEE) can be now be used transparently by any EGEE users, using their regular glite authentication service and job submission tools. We also report on the setup of a new volunteer-like EDGeS@Home public Desktop Grid based on the XtremWeb-HEP middleware similar to BOINC project which allows the general public to participate to the EDGeS project by installing XtremWeb-HEP worker on their PCs. We show that the 3G bridge is a flexible component that, combined with the versatile 3-roles architecture of XtremWeb, allows to build highly distributed and secured computing infrastructure. The performance evaluation demonstrates a reasonable overhead.

The outline of the paper is the following. In the next section, we present related work. In Section 3, we introduce the main components of our architecture: EGEE, XWHEP and the 3G bridge. In Section 4, we describe the actual production infrastructure. In Section 5, we present some performance evaluation results. In Section 6, we give our conclusions, and plan our future work.

2 Related Work

For interoperation between several SGs, some projects have already been achieved by the joint work of several organizations, specially the OGF (Open Grid Forum) [5], the OMII-Europe (Open Middleware Infrastructure Institute), the WLCG (Worldwide LHC Computing Grid).

For interoperation between SGs and DGs, some other projects explored this concept, specially the Lattice project [6] at University of Maryland (USA), the SZTAKI Desktop Grid [7] (Budapest, Hungary), the Condor project (BOINC backfill) at University of Wisconsin (USA), the Superlink [8] project at Technion (Haifa, Israel), and the Clemson University (South Carolina, USA).

Our research is a part of the work of a new European FP7

[1]http://www.eu-egee.org/

infrastructure project: EDGeS[2] (Enabling Desktop Grids for e-Science) [2], whose aim is to build a bidirectional bridge to facilitate interoperability between DGs and SGs.

3 Architecture

3.1 EGEE

EGEE makes grids available to scientists and engineers, the second phase of this project (EGEE-II) 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 40,000 CPUs and about 5PB of storage space, with a throughput of around 100,000 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[3] (Worldwide LHC Computing Grid) and VDT[4] (Virtual Data Toolkit). 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 WN (*Worker Nodes*). 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 WN. WNs are similar to the nodes of a cluster and a group of worker nodes is attached to a CE (*Computing Element*). CEs provide a gateway to the worker nodes and therefore essentially CEs provide the grid resources.

3.2 XtremWeb-HEP

XtremWeb [4] is a research project belonging to light weight Grid systems. It's a Free Open Source and non-profit software platform to explore scientific issues and applications of Global Desktop Grids, Global Computing and Peer to Peer distributed systems.

XWHEP[5] (XtremWeb-HEP) based on XtremWeb is a DG platform developped by LAL to harnest new computing power. This platform is in the Grid family, aggregating volunteer individual computers to create a new powerful computing facility.

For the volunteer communities, this platform includes a set of volunteer PCs running Linux, Windows or Mac OS X. These PCs are not dedicated to XWHEP, they just contribute their free CPU time without disturbing the daily usage. The policies of activation for computing is totally customizable by PCs owners. For example, they can participate the volunteer computing when the CPU is idle, when the screensaver is running, or when scheduled, for instance, only in the night. XWHEP can free PCs automatically. Once the activation conditions are not fulfilled, the running jobs, if any, will be stopped. The CPU , the RAM and disk space are all freed.

XWHEP enforces greatly the security of XtremWeb by introducing mechanisms aiming to secure and confine distributed resources usage. This mechanism is achieved by the implementation of notions of user and access rights. These new feature permit to extend user actions over the platform as well as to secure resource usage and confine application deployment. For more its security issue details see [9].

[2]http://www.edges-grid.eu/

[3]http://lcg.web.cern.ch/lcg/

[4]http://vdt.cs.wisc.edu/

[5]http://www.xwhep.org/spip.php?rubrique13

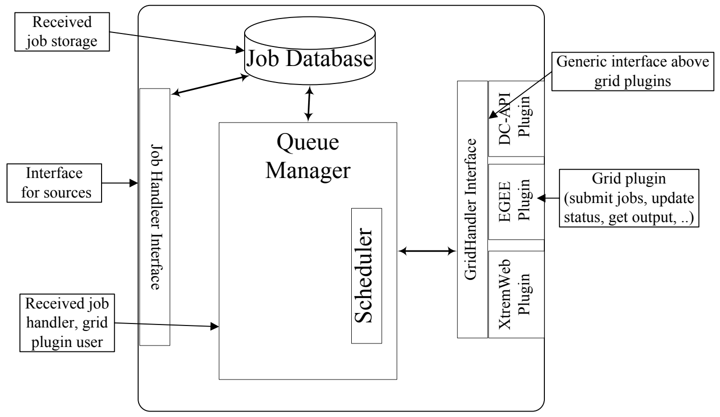


Figure 1: Overview of the 3G Bridge architecture

3.3 Generic Grid-Grid Bridge

The 3G Bridge[6] (Generic Grid-Grid Bridge) has been developed within the scope of the EDGeS project. Its aim is to provide a common bridge between different Grid systems such as EGEE, BOINC or XtremWeb in order to solve the Grid interoperability of these Grid systems. The internal architecture of the 3G Bridge can be seen in Fig.1 where the following components can be identified:

1. *Job Handler Interface* used by the Source Grid producers in order to add jobs to the 3G Bridge and to update the added jobs' status in Job Database for storing the jobs

2. *Queue Manager* for handling the jobs in the database by using a very simple scheduler for calling Grid plugins

3. *GridHandler Interface* that provides a generic interface above the Grid plugins

4. *Grid plugins* that are responsible for handling the jobs in the different destination Grids, thus instances of these plugins are the destination Grid consumers

The 3G Bridge is generic. In fact it is used in the EDGeS project to interconnect Grids that are based on gLite, BOINC and XtremWeb technology. However, an important usage scenario of the 3G Bridge is the parameter sweep application support. Moreover, it is important to note that 3G Bridge has a Web-Service submitter frontend through which it is possible for web service clients to submit jobs. The WS submitter is currently used in the EDGeS project to implement job submission among remote sites having different Grid types (e.g., from EGEE to DG).However, it is also a convenient way to submit jobs to a Desktop Grid similarly to other Grid types where command-line submission utility is provided.

[6]http://edges-3g-bridge.sourceforge.net/

3.4 XtremWeb Handler

The XtremWeb Handler for the 3G Bridge is a plugin which permits to create jobs, to monitor the jobs status, and to retreive results of of jobs executed by an XtremWeb DGs. The XtremWeb Handler takes advantage of the flexibility and the genericity of the 3G Bridge to implement a connection to any kind of XtremWeb Desktop Grid systems. It is sufficient for an XtremWeb administrator to provide an XtremWeb client, configurated with the correct rights for job submission to the 3G Bridge to allow transparent job management.

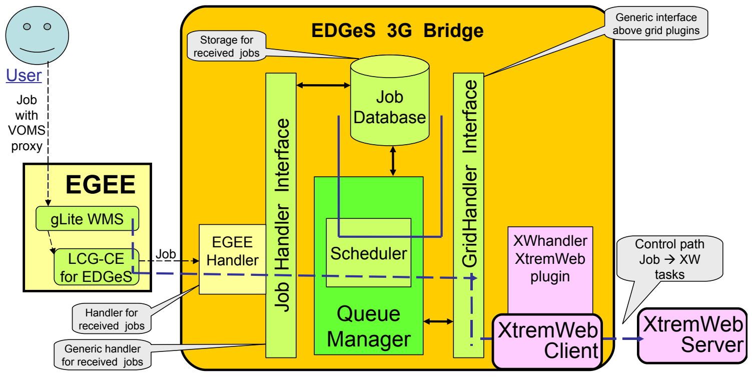


Figure 2: The 3G Bridge with XWHandler

The XtremWeb Handler consists in two parts :

- The *XtremWeb Client*. This Java software, part of the XtremWeb distribution allows remote submission of data and jobs, as well as probing for job status from an XtremWeb server. Typically, each XtremWeb installation provides one or more XtremWeb clients, which are self-contained jar archives, already configured with the user's right management. In essence, the XtremWeb Client interfaces with an XtremWeb Server. The XtremWeb Client and Server are designed to run on different machines, a typical setup would be to run the 3G Bridge and XtremWeb client on a machine with EGEE access and the XWHEP server on a machine without rights to access the EGEE infrastructure. The XtremWeb client is used as a scriptable command-line tool.

- The *XWHandler* is a C++ class serving as an interface between the 3G Bridge and the XtremWeb Client. XWHandler inherits from the generic GridHandler classes of 3G Bridge. These classes provide a list of functions (`submitJobs`, `updateStatus`, `poll`, `getInstance`) to implement, which ensures compatibility with the 3G Bridge. These functions are periodically called by the 3G Bridge. Implementing this set of functions ensures that the XtremWeb Desktop Grid will be able to receive jobs and to provide results to the Bridge.

XWHandler provides following features:

- *Job creation:* The 3G Bridge periodically calls the `submitJobs` function, providing a list of new jobs to insert. According to this list, the XWHandler scan this list and call the XtremWeb Client for every job that has not been yet submitted to XtremWeb. The result of the submission to XtremWeb is a new job identifier, which, once properly registered within the 3G Bridge database, is later used to bind the different Grid jobs together.

- *Job monitoring:* The 3G Bridge requires regular updates of the job's status. XWHandler probes the XtremWeb dispatcher for job's status and makes an association between the state of the job with the XtremWeb dispatcher and the state of the job in the 3G Bridge. At the moment, the association implemented follows the other Grids handler implementation in the sense that every job inserted in XtremWeb has the RUNNING status, regardless of the actual internal status as it appears in XtremWeb (e.g., COMPUTING, and PENDING).

- *Data Management:* Input files, which are stored on Grid nodes, cannot be directly accessed from the Desktop Grid computing resources for obvious security issues. Thus, files are first copied locally to the node running the 3G Bridge daemon. The XtremWeb client registers files on the XtremWeb server. Then it can be accessed in the server by a URI, in a form similar to xw://xwdg.lal.in2p3.fr/43D3A445F53E19338ABBA3. This reference can be used by several jobs which share the same input file.

4 XtremWeb-HEP – EGEE Infrastructure

EDGeS has several goals. One of them is connecting deployed DGs to the new EDGeS infrastructure. Another goal, detailed in this section, is to interconnect XWHEP to the new global EDGeS infrastructure.

4.1 Connecting XWHEP Desktop Grids

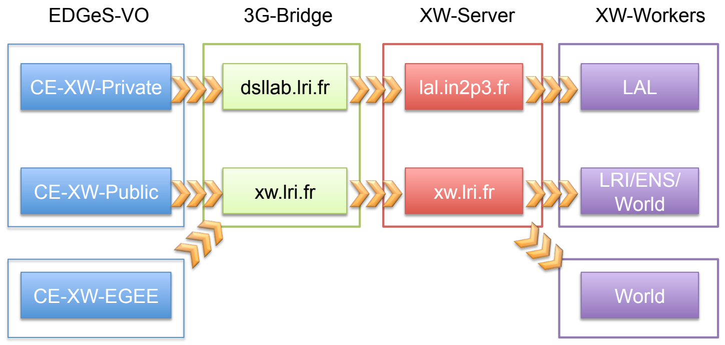


Figure 3: Three XWHEPs are connected to EDGeS infrastructure

In Figure.3, we demonstrates the different XWHEP DGs connected to the EDGeS infrastructure :

- XW-PRIVATE is the local XWHEP run by the LAL/IN2P3 Laboratory. This DG contains High Energy Physics applications and its users are the scientists working at the LAL laboratory. Computing resources are mainly Desktop PCs and Apple notebook computers, as well as a few cluster nodes, all belonging to the LAL institute and managed by the LAL staff.

- XW-PUBLIC is the public XWHEP operated jointly by the LAL and LRI (University of Paris XI). This DG is open to any public or private users who want to create a DG pool of computing resources. At this moment, users mainly belong to three French research instituitions: LAL, LRI and ENS (University of Lyon), although the platform is open to anyone. Of course, both XW-PRIVATE and XW-PUBLIC can run EDGeS applications in addition to their own applications and thus, are able to run EDGeS jobs.

- XW-EGEE is the new public XWHEP dedicated to run EGEE applications and specially created and operated for the EDGeS project. This new DG, which we will describe in this document, only runs EGEE applications.

Before describing the setup of XW-EGEE, it is necessary to give some background about how XWHEPs are connected to EGEE in the EGEE→DG direction. From the EGEE user's point of view, different DGs appear as queues of EDGeS bridge CE (Computing Elements), thus in the connected VO, a CE exists with queues that connect to the different XWHEPs. The complete procedure for a job submitted by an EGEE users from the VO to the volunteer PC is the following:

- The job first goes from the CE queue to the connected 3G Bridge.

- On the 3G Bridge, the destination grid is checked and accordingly, the corresponding XWHEP Client is selected and used to submit the job to the right XWHEP server.

- At the moment, two 3G Bridges are operated in the EGEE→DG direction: one running on the machine `dsllab.lri.fr` which connects the CE-XW-Private CE queue to the XWHEP server running on machine `lal.in2p3.fr`, and one running on the machine `xw.lri.fr` which connects the CE-XW-PUBLIC CE queue to the XWHEP server running on machine `xw.lri.fr`.

- After the job goes to one of the XWHEP server, one of the attached workers will get and process the job.

- The results collect will follow the reverse path.

Now both XW-PRIVATE and XW-PUBLIC run in production. It is possible to leverage this infrastructure for connecting the new XWHEP in order to save development and operating costs thanks to the multi-users and multi-application capability of XWHEP.

XW-EGEE relies on the same server infrastructure as XW-PUBLIC, however, XW-EGEE is a completely new DG, totally separated and independent from XW-PUBLIC. We have created a new *User* and a new *Group* called EGEE which are private. This means that all data, applications, jobs and workers from the XW-EGEE DG will be isolated from the XW-PUBLIC DG and vice versa, XW-EGEE *Users* and *Workers* will not have access to application and data hosted on XW-PUBLIC. Thus, the two DGs are separated and isolated, even if the same XWHEP server is used. A new XWClient software `XWClient-EGEE.jar` is generated, which enables to manage XW-EGEE: job submissions, results retrieval, data management, application insertions etc. In particular, this new client is configured in the 3G Bridge hosted on `xw.lri.fr`, so that the 3G Bridge can forward jobs to either XW-EGEE or XW-PUBLIC depending on the queue which have received the job. Jobs submitted through CE-XW-EGEE will be submitted with EGEE membership and thus executed by XW-EGEE computing resources, while jobs submitted through CE-XW-PUBLIC will have a public membership allowing them to be processed by public resources. A new XWWorker software `XWWorker-EGEE.jar` is generated, which enables to process XW-EGEE jobs. This worker is private and runs under the EGEE identity. This allows participants who run the XWWorker-EGEE to process EGEE jobs only, ignoring the jobs from the XW-PUBLIC DG. As a consequence, deployments are confined and computing nodes can not belong to both infrastructures.

4.2 Deploying applications from the Application Repository

For this infrastructure, only validated applications in the EDGeS Application Repository can be executed. Currently the *DSP: Digital Signal Processing* application is validated and deployed which analyses digital signals in one of the following domains: time domain (one-dimensional signals), spatial domain (multidimensional signals), frequency domain, autocorrelation domain, and wavelet domains. An implementation of DSP application on an academic production Grid :UK National Grid and BOINC Desktop Grid can be found in [10]. It is programmed and highly optimised in C language. It is a typical parameter sweep application. And its algorithm of large data processing can be well parallelised and distributed.

The `xwsendapp` tool was used to install the DSP application downloaded from the Application Repository including binaries for Linux i686, Linux x86_64 and Windows 32 bit platforms, respectively as follows:

```
$ xwsendapp DSP ix86 linux DSP_linux_ix86
$ xwsendapp DSP x86_64 linux DSP_linux_x86_64
$ xwsendapp DSP ix86 win32 DSP_win32_ix86
```

4.3 Monitoring Architecture

The current version of our monitoring architecture comprises of the following parts: 1. XWHEP probe, 2. Data Collector daemon, 3. Ganglia stack, 4. User Interface.

We depict the block view of the interactions between these components in Figure 4. The XWHEP probes must have access to the MySQL database of the XWHEP server. These are at the heart of all the data we collect. We do not modify any data downstream these probes. In the case of this bridge direction (EGEE → XWHEP), we must query the database for jobs submitted by the specific user identifier used by the XWHandler in the 3G Bridge. This works, because when we query for jobs related to this user, we are always sure that such jobs must have crossed the 3G Bridge. Regarding monitoring data that has crossed the bridge, we currently collect the success rate of running jobs, waiting jobs and past jobs.

Probes run periodically to read parameters on a regular basis (e.g., every 5 minutes). From the probes, data follow to the Data

Collector Daemon. This daemon acts as a hub, receiving, then filtering, (optionally) storing and sending data to whatever devices want to receive it. We use Java Enterprise Edition technologies, like Java Message Service and Enterprise Java Beans to implement this component.

The user interface, which displays system information for administrators, is backed by the Ganglia stack that stores and manages data for plots (daily, weekly, monthly or yearly). In fact, the most important component for us (also used and included in Ganglia) comes from the periodic database provided by the RRD Tool[7]. This displays the periodic plots, coming from data sent by the probes.

The entire flow of monitoring data goes through the following hops:

1. XWHEP probe reads the XWHEP database
2. Sends data to the Data Collector Daemon see Figure.5, which spawns one `gmetric` process.
3. From `gmetric`, data go through the usual path in the Ganglia architecture to reach `gmetad`, which writes data in the `RRD Tool` database.
4. When the user visits the appropriate page and requests a given parameter, a Java servlet in the Java Server Faces (JSF) framework will issue a request to the `RRD Tool` and produce a plot on-the-fly (if needed).

This monitoring architecture also provides monitoring web pages[8] as shown in Figure.6. At the time we write this paper, we are in the process of updating the web site to reflect the novel architecture and bridge implementation we are describing.

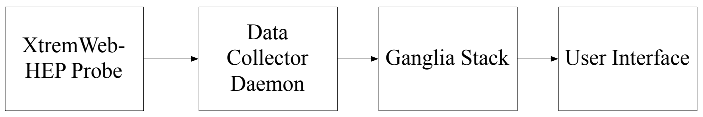


Figure 4: XW-EGEE Monitoring Architecture

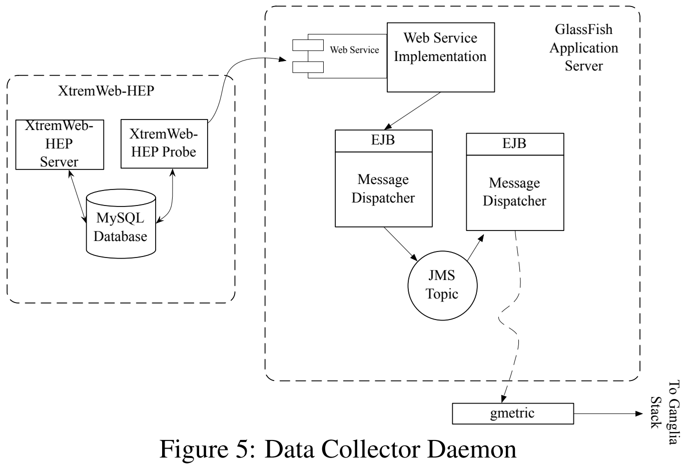


Figure 5: Data Collector Daemon

Figure.6 gives an overview of the EGEE→XWHEP page. The default view contains four plots per application per parameter, corresponding to the last 24 hours, last week, last month and last year. Additionally, we can replace the plots by numerical values, corresponding to averages (or to sums). To switch on or off some of the plots, we use Javascript, thus reducing the interaction with the web server.

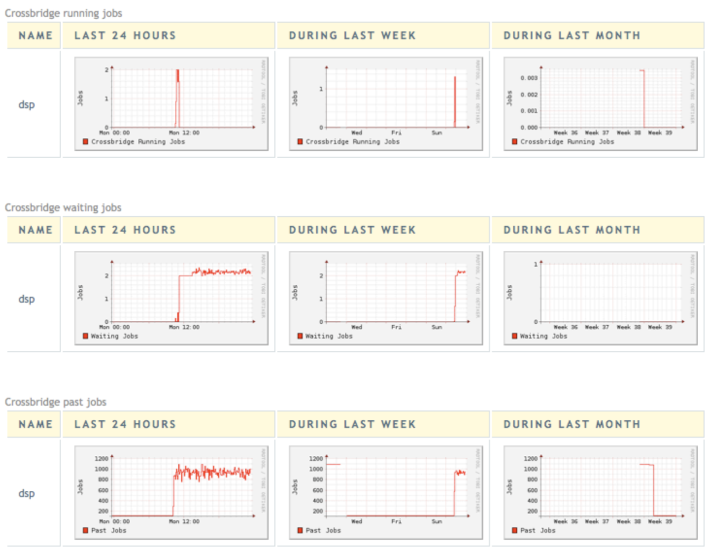


Figure 6: DSP monitoring view

5 Performance Evaluation

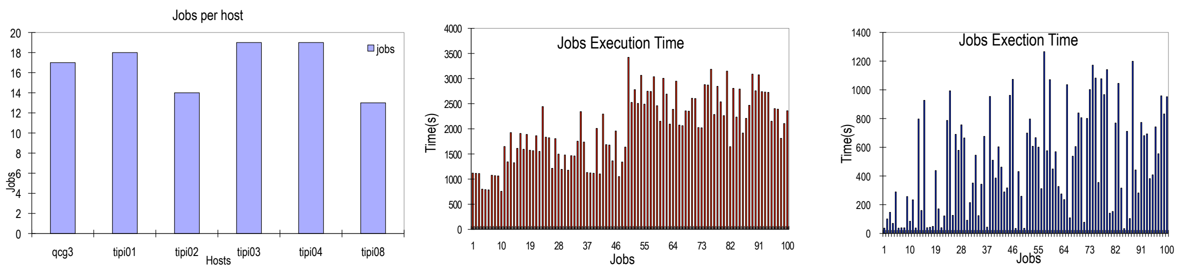
We used QCG, a cluster of 4 multi-core-based nodes with dual-core Intel Pentium D (2.8GHz2x1MB L2 cache) processors interconnected by a 100MB Ethernet network, TIPI, a cluster of 50 multi-core-based nodes with dual-core AMD Opteron (1.8GHz4x1MB L2 cache) processors interconnected by a 100MB Ethernet network. We have a multi-core platform based on these two clusters to provide our XWHEP workers resources, for the details of this built platform see Table.1. Nowadays, such a multi-CPU and multi-core platform becomes the main stream system for desktop computing. We also note that when we run our experiments jobs, these resources are at the same time shared by other users. Jobs are executed according to local resource allocation policies of XWHEP.

Host	CPU	CPU Speed	CPUs	Cores
tipi01	AMD Opteron 265 DuoCore	1.8G	4	8
tipi02	AMD Opteron 265 DuoCore	1.8G	4	8
tipi03	AMD Opteron 265 DuoCore	1.8G	4	8
tipi04	AMD Opteron 265 DuoCore	1.8G	4	8
tipi08	AMD Opteron 265 DuoCore	1.8G	4	8
qcg3	Intel Pentium D DuoCore	2.8G	2	4
Total	2	2	22	44

Table 1: Resources of XWHEP workers

At present, totally more than 11,000 jobs have been sent and executed in our infrastructure. Here, we just present the results of benchmark for a bunch of jobs. We submitted the same 100 DSP application jobs mentioned in Section 4.2 from EGEE node of SZTAKI site. There is a good distribution of jobs which bridged from EGEE to XWHEP shown in Figure.7a. Figure.7b and Figure.7c show respectively the execution time of these 100 bridged jobs measured by EGEE and XWHEP server. We can see the execution time varies for the same job because of the different availability and heterogeneity of each computing resource.

[7]http://oss.oetiker.ch/rrdtool
[8]http://edges.dei.uc.pt/EDGeSMonitoring



(a) Jobs per host bridged from EGEE to XWHEP (b) Execution time on EGEE (c) Execution time on XWHEP

Figure 7: Performance of bridged jobs from EGEE to XWHEP

The execution time of each job measured directly by XWHEP is much shorter that that measured by EGEE. The reason is that the overhead for jobs bridged from EGEE to XWHEP is not negligible.

In Table.2, we present the average time of each steps from submission to completion of 100 bridged jobs. We recorded the used time step by step: the accepted time recorded by Network Server, the match time in the Workload Manger, the transfer time from JobControler to Log Monitor, the accepted time recorded by Log Monitor, the transfer time from Log Monitor to LRMS (Local Resources Management System in EGEE), and finally the running time. We can see the majority of time is used for running jobs, the transfer of data and program is less time consuming in this case of a relative small DSP application jobs (1.1M with input and output files). The cost includes jobs registration and match time, scheduling time, transfer time from EGEE to XWHEP etc, which is necessary for EGEE jobs to run in XWHEP.

1. Accepted Time(Network Server)	1.40
2. Match Time(Workload Manger)	0.93
3. Tranfer Time(Job Controler to Log Monitor)	1.49
4. Accepted Time(Log Monitor)	2.86
5. Tranfer Time(Log Monitor to LRMS)	10.63
6. Running Time	1965.38
Total Time	1982.69

Table 2: Average time of each steps of 100 bridged jobs (seconds)

6 Conclusions and Future Work

In this paper we have presented our EGEE → XWHEP infrastructure using Generic Grid-Grid Bridge. We described the architecture of XWHEP, Generic Grid-Grid Bridge, the XtremWeb Handler for this bridge, and the technical details of our EGEE → XWHEP infrastructure. We have shown that these software components are flexible and allow to build distributed and complex architecture. Our production infrastructure allows now EGEE users to dispatch work to XWHEP, and is tested using real-world applications.

Finally, we present as well the performance evaluations done in our infrastructure. Although, the overhead for jobs going through the bridge is not negligible, we exploit the potential huge number of resources of DGs. Our present work is now to support more application validated by the EDGeS Application Repository. For the future work, we plan to connect our infrastructure to Grid'5000 which is a French experimental Grid to investigate how Grid resources could contribute to the EDGeS project.

As EGEE is based on gLite middleware, our bridge technology can be used for all gLite-based Grids. With minor improvements, our solution can be adapted to the Globus or Unicore based Grids. We would like to extend this infrastructure to all kinds of Grids in order to turn Grid computing, which is almost only used in universities and institutions, into Public computing, which is popular on a lot of PCs at home.

Acknowledgements

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

References

[1] Hai Jin. ChinaGrid: Making Grid Computing a Reality. In *Lecture Notes in Computer Science, Volume 3334*, pages 13–24, Springer-Veralag Berlin Heidelberg, 2004.

[2] Etienne Urbah, Peter Kacsuk, Zoltan Farkas, Gilles Fedak, Gabor Kecskemeti, Oleg Lodygensky, Attila Marosi, Zoltan Balaton, Gabriel Caillat, Gabor Gombas, Adam Kornafeld, Jozsef Kovacs, Haiwu He, and Robert Lovas. Edges: Bridging egee to boinc and xtremweb. *Journal of Grid Computing*, 7(3):335–354, September 2009.

[3] David Anderson. BOINC: A System for Public-Resource Computing and Storage. In *proceedings of the 5th IEEE/ACM International GRID Workshop*, Pittsburgh, USA, 2004.

[4] Franck Cappello, Samir Djilali, Gilles Fedak, Thomas Herault, Frdric Magniette, Vincent Nri, and Oleg Lodygensky. Computing on large scale distributed systems: Xtremweb architecture, programming models, security, tests and convergence with grid. *Future Generation Computer Science*, 2004.

[5] Riedel, M, et al. Interoperation of world-wide production e-science infrastructures. In *Concurrency and Computation: Practice and Experience*, 2008.

[6] Daniel S. Myers, Adam L. Bazinet, and Michael P. Cummings. Expanding the reach of Grid computing: combining Globus- and BOINC-based systems. In A. Zomaya, editor, *Grids for Bioinformatics and Computational Biology*, Book Series on Parallel and Distributed Computing, pages 71–85. John Wiley & Sons, New York, 2008.

[7] 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 *the 21st International Parallel and Distributed Processing Symposium*, Long Beach, California, USA, 2007.

[8] Geiger D Fishelson M. Exact genetic linkage computations for general pedigrees. *Bioinformatics. 2002;18 Suppl 1:S189-98.*, 2002.

[9] Gabriel Caillat, Gilles Fedak, Haiwu He, Oleg Lodygensky, and Etienne Urbah. Towards a Security Model to Bridge Internet Desktop Grids and Service Grids. In *Proceedings of the Euro-Par 2008 Workshops (LNCS), Workshop on Secure, Trusted, Manageable and Controllable Grid Services (SGS'08)*, Las Palmas de Gran Canaria, Spain, August 2008.

[10] Andrzej Tarczynski, Tamás Kiss, Gábor Terstyánszky, Thierry Delaitre, Dongdong Qu, and Stephen C. Winter. Application of grid computing for designing a class of optimal periodic nonuniform sampling sequences. *Future Generation Comp. Syst.*, 24(7):763–773, 2008.