Future Generation Computer Systems 26 (2010) 1092–1103

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Interoperability of BOINC and EGEE[☆]

Z. Farkas[*], P. Kacsuk, Z. Balaton, G. Gombás

MTA SZTAKI, 1518 Budapest, P.O. Box 38, Hungary

ARTICLE INFO

Article history:
Received 4 September 2009
Received in revised form
28 December 2009
Accepted 6 May 2010
Available online 20 May 2010

Keywords:
Grids
Service grids
Desktop grid
Volunteer computing
BOINC
EGEE
gLite
Interoperability

ABSTRACT

Today basically two types of grid systems are in use: service grids and desktop grids. Service grids offer an infrastructure for grid users, thus require notable management to keep the service running. On the other hand, desktop grids aim to utilize free CPU cycles of cheap desktop PCs, are easy to set up, but the availability towards users is limited compared to the service grid. The aim of the EDGeS project is to create an integrated infrastructure that combines the advantages of the two grid concepts. A building block of this infrastructure is bridging between the different grid types. In this paper, we first focus on bridging from BOINC-based desktop grids towards EGEE-like service grids, i.e., making desktop grids able to utilize free service grid resources. The solution is based on a generic grid to grid bridge, called as, 3G Bridge. In the second part of the paper we show how the 3G Bridge and EDGeS Bridge services can be used to realize the reverse direction interconnection of BOINC and EGEE grids, i.e., sending EGEE jobs in a user transparent way to BOINC systems that are connected to EGEE VOs. This is the first paper in which we publish the full two-directional bridging between BOINC and EGEE grids.

1. Introduction

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

Researchers and developers in Service Grids (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. The middleware is, however, so complex that it often requires extensive expert effort to maintain. It is therefore natural, that individuals do not usually 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 US TeraGrid. 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 144.000 PCs [1]. 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 (DG) 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 compute resources, or “spare cycles”. 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 and 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 example is the SETI@HOME project [2], in which approximately four million PCs have been involved.

☆ 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).

* Corresponding author. Tel.: +3613297864; fax: +3613297864.
E-mail addresses: zfarkas@sztaki.hu (Z. Farkas), kacsuk@sztaki.hu (P. Kacsuk), balaton@sztaki.hu (Z. Balaton), gombasg@sztaki.hu (G. Gombás).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.05.009

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 Desktop Grid systems, cannot use the system for their own goals. Because of this limitation, the Grid research community considers desktop Grids 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 to exploit their individual advantageous features in a unified environment. However, with the objective to support new scientific communities who 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.

The European EDGeS project [3] aims at establishing an integrated SG–DG infrastructure where the DG $\rightarrow$ SG bridge enables desktop grids to submit task units as jobs into the service grids and the SG $\rightarrow$ DG bridge ensures the job movement into the opposite direction. In this paper, we describe research on how such DG $\rightarrow$ SG and SG $\rightarrow$ DG bridge can be created for BOINC and EGEE.

2. Related work in bridging service and desktop grids

According to [4], grid interoperability solutions achieved so far are based on short-term solution. The paper focuses on different grid components (like job management, data management, information systems), and urges the usage of standards-based solutions.

The interoperability solution for UNICORE and Globus described in [5] uses a translation mechanism to allow the execution of jobs submitted to UNICORE on Globus.

In paper [6] authors show two solutions for using Condor resources in an OGSA-based infrastructure. The first option hides the details of the Condor system: job submission, job execution management and resource information providers are offered for the OGSA grid to hide Condor details. On the other hand, the second option embeds OGSA within the Condor framework in order to provide controlled access to the Condor resources.

Grid interoperation can also be solved at a higher level, using additional tools like described in [7] or [8]. The solution presented in [7] offers a tool that is able to submit jobs to different types of grid middlewares based on user requirements. On the other hand, paper [8] presents a meta-broker that is able to send jobs to different types of grid middlewares, thus the middleware selection is solved using a brokering mechanism.

In paper [9] authors describe different approaches and levels for solving grid interoperability:

Virtual organization driven in this approach VOs are responsible for integrating the different grid infrastructures into their framework,

Parallel deployment in this approach sites deploy interfaces of different grid infrastructures above their computing resources,

Gateways in this approach a gateway is used to show an additional grid infrastructure as a resource of a well-known grid infrastructure.

The paper also urges grid middleware developers to move towards using standards based on experienced gained during working within the Grid Interoperability Now (GIN) efforts of OGF.

There are basically two main concepts to connect different grid systems in terms of sending jobs transmitting jobs between the grid systems: the *pilot job* method and the *job wrapping* approach. In this section we will present these two approaches from different view points.

2.1. The job wrapping method

The job wrapping method proposed by the Lattice [10] project and the SZTAKI Desktop Grid [11], is based on the following idea: from the source grid's point of view, a very powerful machine (*gateway*) is responsible to catch jobs originating from the source grid, and transfers them to the destination grid. This machine is responsible for the bidirectional communication with both the source and the destination grid. Its most important tasks are the following:

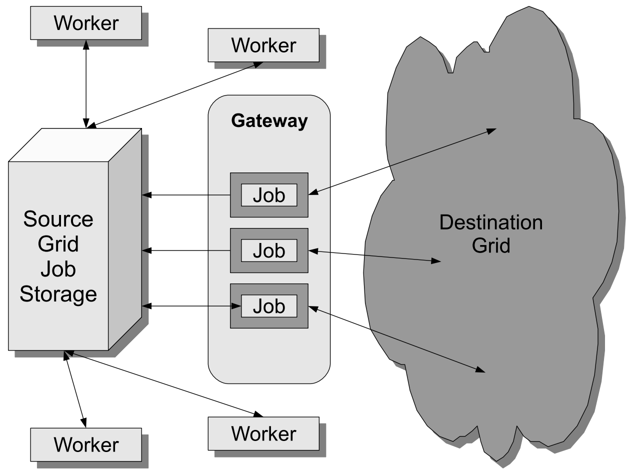

Fig. 1. The job wrapping method.

- fetch work from the source grid,
- wrap incoming jobs, and send them to the destination grid,
- observe the execution of destination grid jobs, get their results once finished,
- report results of completed destination grid jobs towards the source grid.

The problems of this solution are:

- the gateway can become a bottleneck when more and more source grid jobs should be executed in the destination grid and introduces a single point of failure in the system affecting a lot of jobs in case of errors, which lessens fault-tolerance,
- the turnaround time of a source gird job is increased because it has to be managed by the gateway, and in case of a destination grid using a resource broker, the destination grid job has to be scheduled onto an available resource.

On the other hand, the job wrapping solution provides better security properties, concerning the integration with the destination grid. First, the gateway does not require the modification of the infrastructure, it can run under any user identity as long as the user has the right to submit jobs on destination grid. Next, source jobs are wrapped by the gateway, they run under the user identity that conforms with the regular security usage, in contrast with the approach described in the following paragraph. Finally, if the gateway implements a job filtering policy, then the set of jobs that can be executed might be restricted.

An overview of the job wrapping method can be seen in Fig. 1, where traditional workers process jobs of the source grid, and the gateway wraps jobs originating from the source grid into destination grid jobs, and runs them on the destination grid.

2.2. The pilot job method

The basic idea of the pilot job method is to run workers of the source grid (pilot jobs) in the destination grid. As the result, a "super-cluster" of the source grid workers is created above the destination grid resources. This approach to cluster resources spread in different Condor pools using the Global Computing system (XtremWeb) was first introduced in [12]. The main principle consists in wrapping the XtremWeb worker as regular Condor task and submitting this task to the Condor pool. Once the worker is