Chapter

DESKTOP GRID IN THE ERA OF CLOUD COMPUTING

***Peter Kacsuk*[1,2,*], *Zoltan Farkas*[1], *Jozsef Kovacs*[1], *Adam Visegradi*[1], *Attila Marosi*[1], *Robert Lovas*[1], *Gabor Kecskemeti*[1], *Zsolt Nemeth*[1] and *Mark Gergely*[1]**
[1]MTA SZTAKI, Institute for Computer Science and Control of the Hungarian Academy of Sciences, Hungary
[2]Centre of Parallel Computing, University of Westminster, UK

ABSTRACT

BOINC desktop grids have been used for more than a decade for running grand challenge applications. In this chapter we show those technologies, and particularly virtualization and cloud solutions, that make BOINC desktop grids more generic and speed up the execution of existing grid-enabled parameter sweep applications without porting them to BOINC and eliminate the tail problem of volunteer BOINC systems. Furthermore, we show a technology by which institutional and public desktop grids can be created in a few minutes in clouds and used without any BOINC knowledge.

Keywords: gLite, BOINC, volunteer computing, desktop grid, cloud

INTRODUCTION

Desktop grids (DG) provide the cheapest possible way of building large computing infrastructures for e-science. BOINC became very popular to support grand-challenge scientific applications like climate research, anti-cancer research, etc. However, its mass usage among scientists has not happened so far although, most of the scientific applications are large parameter study simulations where BOINC-like volunteer computing could help a lot. The reason is that porting an application to BOINC is not straightforward. In SZTAKI, we have

[*] Corresponding Author address: 1111 Budapest, Kende street 13-17, Hungary. Email: peter.kacsuk@sztaki.mta.hu.

developed several tools that made the porting much easier, but the real breakthrough has happened when we introduced virtualization in the BOINC client. We have developed GBAC (Generic BOINC Application Client) [10] based on VirtualBox virtualization technology. As a result, porting does not require any effort and existing applications can be instantly used in BOINC systems without preregistering them on the BOINC server. We have also developed 3GBridge [9] that enables access to BOINC systems via a high level job submission interface, and hence users can exploit BOINC systems even from complex workflow applications. 3GBridge also enables the connection of BOINC to other types of grid middlewares such as gLite [1], ARC [2], and UNICORE [4]. In this way, for example, gLite based middleware can be extended with institutional and volunteer BOINC systems in a transparent way. As a result, gLite users can access larger number of free resources without changing their applications.

Cloud computing [12] opens a new horizon to support the DG systems and hence, can help in many ways to further disseminate the idea of desktop grids and make them easily accessible for researchers. The first aspect in which clouds can assist volunteer desktop grids is the possibility of significantly reducing the tail problem [22] (occurs when failures of unreliable resources accumulate and cause significant delay in the last 10% of batch) by extending a desktop grid with dedicated cloud resources. The number of these resources can be increased and decreased according to the number of tail tasks in the different computations running in the desktop grid. The second issue that prevented researchers from extending their existing grid infrastructure (for example, Globus [3]), with DG systems was the difficulty of creating and installing BOINC systems. Cloud computing solves this problem by providing an extremely easy and fast way to deploy the core BOINC system in a cloud, and then extend it with volunteer resources. We have created such a system to demonstrate the applicability of this technology. This approach can be used in a more generic way to create any kind of architecture in the cloud on demand.

The chapter is organized as follows. First, we shortly summarize the BOINC volunteer desktop grid technology and one of its projects, called EDGeS@home, that was created for EGI (European Grid Infrastructure) user communities. Next we explain how the gLite based VOs (Virtual Organizations) can be extended with BOINC desktop grids, and particularly with EDGeS@home. Afterwards, the chapter introduces the GBAC virtualization technology and its advantages. Then the chapter focuses on how clouds can be used to eliminate the tail problem in desktop grids. Later, we show examples on how EGI user communities access and use EDGeS@home. Subsequently, the chapter shortly describes the One Click Cloud Orchestrator (OCCO) cloud deployment tool and its usage for quickly creating BOINC desktop grid systems without BOINC knowledge. Finally, we summarize the most important aspects of related research.

EDGeS@home As a Volunteer Desktop Grid

The generic mechanism of BOINC-based volunteer desktop grids can be summarized in the following way. Volunteers join the project by downloading and installing a piece of lightweight software: the BOINC client. First, the client downloads the registered application binaries from the BOINC project. Second, it periodically fetches new input files and parameters for the application in a form of so called workunits. These downloaded tasks are then processed in the background on the volunteer resources.

Any application supported by a desktop grid must be deployed and registered in the project, thus it is not possible to submit and run arbitrary ones. Also, there is no possibility of communication between running tasks, the so called embarrassingly parallel applications are supported only. Therefore, desktop grids are best suited for bag-of-tasks or parameter study type applications, where the applications do not change often.

EDGeS@home is a BOINC [5] based volunteer desktop grid project established with the goal to support EGI (European Grid Infrastructure [19]) user communities in executing large parameter sweep applications. Running since 2008, it has been started under the Enabling Desktop Grids for e-Science (EDGeS) EU FP7 project [15]. It was maintained further by the successor of EDGeS, the European Desktop Grid Initiative (EDGI) [8] until late 2012 and is currently maintained by the International Desktop Grid Federation Support Project (IDGF-SP) [16]. EDGeS@home has currently over 13,000 registered volunteers with 21,000 registered hosts.

Traditional desktop grids usually target a single grand-challenge scientific problem. Contrary to the traditional ones, EDGeS@home is running multiple applications, making it an “umbrella” project in order to support as many EGI user communities as possible. Any volunteer has the freedom to select which application(s) he or she wants to support from the deployed ones. This means that applications are usually competing with each other for a given set of resources and each of them can typically use a fraction of the client machines registered for the project. At the time of writing, EDGeS@home hosts 10 applications from different science domains such as physics, logistics or biology, and a framework called Generic BOINC Application Client (GBAC, to be discussed in a later section) [10], which allows running arbitrary applications inside virtual machines.

EDGeS@home is part of the IDGF desktop grid Regional Operations Centre (ROC) in the EGI infrastructure. This centre collects volunteer resources, each of its sites representing a BOINC project. The ROC has been founded with two sites, one of which is EDGeS@home, in order to make the volunteer BOINC projects visible for the EGI user communities as normal gLite resources. The ROC has the same services as normal resource sites have: information system, workloads management system, job submission interface, monitoring and accounting system, and authentication. The EDGeS@home BOINC project is open for EGI scientists for submitting jobs.

EGI user communities typically use ARC [4], gLite [1] and UNICORE [2] middleware based grid systems. They have got used to these grids, and quite often they are not prepared to learn a new type of grid middleware like BOINC. Therefore, if we want to support these communities we have to solve the issue of automatically transferring parameter sweep application jobs from their grid infrastructure into the EDGeS@home BOINC infrastructure. The advantage for the users is accessing hundreds of thousands of resources which is a higher scale than ARC, gLite or UNICORE usually provides. To achieve this goal the EDGI EU FP7 project has extended ARC, gLite and UNICORE with the capability of transferring jobs to BOINC desktop grids. In the next Section, as an example, we describe this solution for gLite. The interested reader can find the description of the ARC and UNICORE integration as well in the deliverables of EDGI project [8].

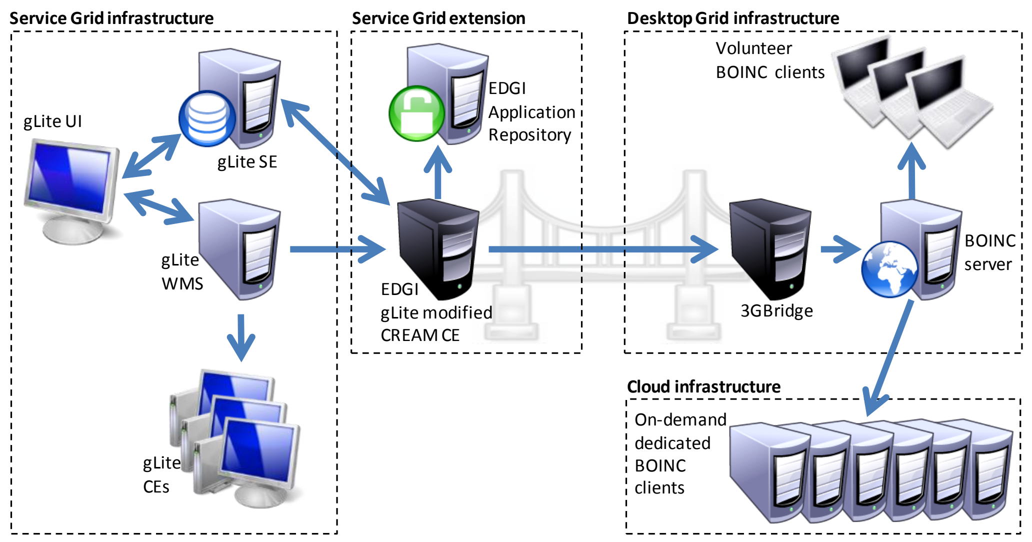

Service Grid infrastructure
Service Grid extension
Desktop Grid infrastructure
gLite UI
gLite SE
gLite WMS
gLite CEs
EDGI Application Repository
EDGI gLite modified CREAM CE
Volunteer BOINC clients
3GBridge
BOINC server
Cloud infrastructure
On-demand dedicated BOINC clients

Figure 1. EDGI Infrastructure to bridge gLite jobs to BOINC DGs.

EXTENDING GLITE VOS WITH BOINC SYSTEMS

The goal of extending gLite VOs with BOINC systems is to transparently transfer parametric jobs from a gLite VO to one or more supporting BOINC systems, and to distribute large number of job instances of parametric jobs among the large number of BOINC client resources.

In order to extend gLite VOs with BOINC systems we have designed a bridging solution. The key component is the modified Computing Element (mCE; see “EDGI gLite modified CREAM CE” in Figure 1). It extracts the job from the gLite system and transfers it to a remote desktop grid site. On the remote BOINC server a Generic Grid-Grid (3G) Bridge service [17] (“3G Bridge” in Figure 1 is running to receive the incoming jobs and to insert them into the BOINC server. These two components (modified CREAM [1] CE and the 3G Bridge) in the infrastructure represent the two pillars of the bridge.

To control which jobs can be transferred through the bridge, the EDGI application repository [18] (“EDGI Application Repository in Figure 1) has been introduced. This contains those registered applications that are validated for the gLite→BOINC execution.

Concerning the security mechanisms of the two middleware, in gLite, compute resources trust the user who is holding a certificate, and accept any kind of executable from a trusted user. In desktop grids the compute resources (i.e. the clients, or donors, who offer their resources) trust the DG project server, and accept only pre-registered and validated executables. To overcome this conceptual difference we combined the two concepts by restricting the applications that can be passed from the gLite system to the particular DG system. This means that the transfer is only realized when a trusted user submits an application that is trusted by the target DG system. Enabling the transfer is done by the central EDGI Application Repository (AR) that stores the validated/trusted applications with all of their executable binaries to be submitted, and with a list of DG systems that trust and has registered this application. Applications are registered by the admins, users can list/query or request new registrations on demand. The EDGI AR [25] distinguishes several platforms (gLite, BOINC, XtremWeb, etc.) for which developers can upload the compatible binaries. In addition to the binaries, the AR stores the description of the application, an example jdl and input files, and all the pieces of information that are necessary to perform a correct job submission. The central EDGI AR also stores the information about which VO the particular application can be submitted from and which computing element (and its queue) will be able to handle the job submission successfully. This information is needed since DG systems are configured for and can be reached through a special CREAM CE queue.

This solution was too complicated for the user communities: it required first the porting of the original grid (ARC, gLite or UNICORE) application to BOINC, then the ported application had to be validated for the various grid and BOINC combinations. In order to avoid all these complexities we have developed the GBAC virtualization framework [10] that has been introduced to enable executing jobs inside virtual machines on the BOINC clients. With the help of the virtualization environment GBAC provides, the mCE became able to automatically convert non-registered applications into a GBAC workunit. Using virtualization on volunteer resources, untrusted applications can also be executed safely under BOINC. Moreover, gLite users do not need to modify their submission jdl file when changing from normal gLite resources to desktop grid ones.

The last problem that kept grid users from using BOINC was the so-called tail problem [22]. Based on experiences of the BOINC community, the first 80-90% of the jobs belonging to one application are executed according to the expected speed-up of parallel systems but the last 10-20% are executed much slower since the unreliable volunteer BOINC clients can cause significant delay in giving back the result of the task they registered for solving. This tail problem can be solved in several ways and we proposed the extension of Desktop Grid infrastructure with on-demand dedicated cloud resources (“On-demand dedicated BOINC clients” in Figure 1) that proved to be a very efficient solution. This cloud extension will be detailed in a later section.

Before giving more detailed description of the cloud-oriented extensions of desktop grids, first we show the two most popular deployment scenarios of creating such a combined infrastructure. In the first case, extending the existing gLite VO with already gathered and maintained volunteer resources is shown, while the second case details setting up the combined infrastructure based on an institutional desktop grid system.

Case Study: Extending a gLite VO by the EDGeS@home Project

There are many gLite VOs around Europe, for which gLite administrators are taking care of the infrastructure, and not willing to take up operating a separate desktop grid site for collecting volunteer resources. EDGI (and later IDGF-SP [16]) eases their work by maintaining and offering access to the extension services (“Service Grid extension” in Figure 1) and the required volunteer desktop grid project: EDGeS@home with a large number of volunteer desktop grid resources, and cloud resources in addition. In this scenario, the gLite VO admin has nothing to do; EDGI (or IDGF-SP) takes care of the operation of the modified computing element, the 3G Bridge and all the desktop grid related tasks. After the necessary configuration is performed in the Service Grid extension, the gLite mCE appears among the CEs of the supported VO. This experiment is an ongoing and continuous activity performed by the IDGF-SP project in order to support various EGI VOs in accessing EDGeS@home.

Case Study: Extending a gLite VO by Institutional DG Site

BOINC can be used as an effective solution for aggregating the available compute capacity of desktop computers inside an institute. For example, universities can easily follow this strategy to create a campus-wide desktop grid system to provide large computational capacity for the researchers. University of Westminster is a good example, where approximately 1800 machines have been collected and utilized by a campus desktop grid and offered for their scientists to run various simulations [20]. Such a newly built campus DG site is a good candidate to be bridged to a gLite VO where the university is a member and to provide a significant capacity increase to that VO. To support this scenario, the IDGF-SP project has consolidated all the necessary software components (gLite mCE, AR, 3G Bridge, GBAC, etc.) and provides them freely for academic institutes. The project set up a website [21] for system administrators on how to setup and operate a combined gLite desktop grid infrastructure. This scenario requires very little effort from the administrators if the required components (BOINC server, 3G Bridge, gLite mCE) of such an infrastructure are deployed in a cloud system. In order to support this easy deployment and maintenance, IDGF-SP provides the necessary cloud images [21] too, as described below in detail in the section about the One Click Orchestrator.

GBAC for Avoiding Application Porting

As mentioned in the previous section, gLite jobs can be transparently transferred to a connected BOINC DG system if the application has been already ported to BOINC. Unfortunately, the porting effort can sometimes be significant, which keeps user communities from applying desktop grid technologies. SZTAKI has developed the DC-API [7] and GenWrapper [6] tools that significantly reduced the porting effort, but even this was not enough. The communities did not want to invest in any porting effort at all. This is understandable, as they have free access to managed grid resources, i.e. someone else pays for their application execution on the managed grid resources. However, if they have to pay for the resources, as in the case of commercial clouds, this attitude will change significantly. Nevertheless, we had to find a solution on how to avoid application porting for BOINC systems.

As a result of searching for such a solution, we developed the Generic BOINC Application Client (GBAC), a virtualization-based wrapper. It is aimed at a generic framework providing virtualized environments for different distributed computing infrastructures (DCIs). GBAC is implemented using the DC-API Meta API [7] and does not rely on any middleware-specific functionalities. Thus, it can be used not only with BOINC, but also on any DCIs that are supported by DC-API. GBAC currently supports VirtualBox as hypervisor. From here on, we refer to the BOINC version of GBAC for demonstrating its concepts and internals.

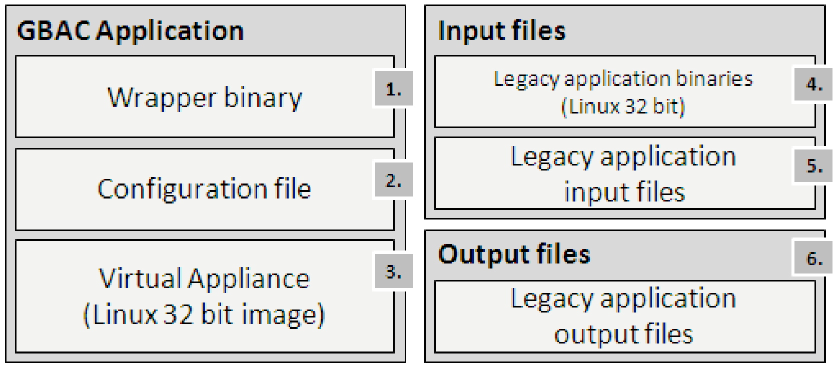

GBAC Application
Wrapper binary
1.
Configuration file
2.
Virtual Appliance (Linux 32 bit image)
3.
Input files
Legacy application binaries (Linux 32 bit)
4.
Legacy application input files
5.
Output files
6.
Legacy application output files

Figure 2. GBAC: application, inputs and outputs.

As shown in Figure 2, the GBAC wrapper consists of the following components. First, the wrapper binary itself is a BOINC enabled DC-API application that contains all BOINC related parts and handles communication with the BOINC client.

Its task is to set up the client execution environment and manage the virtual machine on the client machine. Second, a user-supplied XML based configuration file is used to set the parameters of the virtual machine: (i) the operating system type (e.g., Linux 64 bit); (ii) the size of the allocated memory for the virtual machine; (iii) whether the machine should have network access; (iv) which virtual appliance to use; (v) whether to enable a shared directory between the host and the guest (the virtual machine); and (vi) whether to enable network for the guest. The third component is a compressed virtual appliance that contains the operating systems and libraries for the virtual machine. By default it contains a 32 bit Linux installation with the Guest Tools component of GBAC installed. However, arbitrary Linux distributions, both 32 bit and 64 bit, can be used instead. The Guest Tools, which is the fourth and final component, is deployed in the virtual appliance, and handles the interaction between the GBAC wrapper and the application inside the virtual machine. It acts as an agent of the wrapper, sets up a sandboxed environment for the application, executes it, performs logging, and returns the outputs. More details about GBAC can be found in [10].

Contrary to the normal BOINC use case, where the available applications are registered at the project, GBAC provides two use cases. First, a single-purpose mode, in which GBAC is registered with a single legacy application. This conforms to the normal BOINC use case with the following additions: (a) the legacy application does not require any BOINC related porting; (b) the legacy application will be run in a homogeneous environment based on the same virtual applicance (VA) regardless of the software and hardware characteristics of the donor machines; and (c) as a consequence, only a single binary of the application compatible with the guest VM under GBAC is required. The second use case is the multipurpose mode. Here, GBAC is deployed on BOINC without any legacy applications, and arbitrary ones can be submitted as inputs for GBAC. In this case, both the legacy application binaries and their inputs are inputs of GBAC. This allows, on top of the previous additions ((a) - (c)), to overcome the limitation BOINC imposes on applications. Namely, that each of them must be deployed and registered before use, and only these trusted ones are available to the users. GBAC allows delegating this trust and, instead of the legacy application, the submitting user is trusted. As a result, beyond avoiding the application porting effort, there is another advantage of using the GBAC concept.

It eliminates the need of using the EDGI Application Repository (which filters the applications to be executed). In the case when the target BOINC server is equipped with the GBAC virtualization framework, the modified Computing Element recognizes this situation and skips the AR checking procedure, and submits the incoming job wrapped as a GBAC-job to the BOINC server. All is done automatically, no user interaction is needed for the bridging. In this case, the BOINC clients trust the GBAC application, which safely encapsulates the original job by running it inside a virtual machine.

CLOUD FOR TAIL ELIMINATION

EDGeS@home has over 21,000 registered volunteer hosts. However, the active number of hosts is always lower and varies over time depending on the volunteers' usage pattern; e.g., some volunteers only run BOINC when their computer is idle, but some turn off completely their computers when they are about to be idle for longer periods of time. Hardware failures on volunteer resources cannot be neglected either. The consequence of these is that the task failure and completion deadline-miss ratios can be high. In case of batches of tasks, a single delayed task is going to affect the completion time of the whole batch, resulting in the so-called tail problem [22].

There are different methods available for improving task completion times: (i) redundant computing, (ii) reliability and availability based scheduling, and (iii) resubmission. Redundancy (i) means that multiple copies of the same task are distributed to clients with the expectation that at least a single one is going to finish in time. Reliability and availability based scheduling (ii) means that the system will prioritize the reliable resources over less reliable and less available. Finally, resubmission (iii) identifies problematic tasks and submits them again, e.g., in case of a batch the last 10% uncompleted tasks can be resubmitted trying to avoid the tail effect. Each of these methods on its own will improve the completion time to some extent; however, usually a combination of these is used. The problem is always twofold. First, the set of resources that is considered highly available and reliable must be determined. Second, a redirection mechanism is required for the tasks. Determining reliability and availability is a challenge itself; however, there are already good mechanisms implemented, e.g. for BOINC, that can be used as a foundation [23].

One of the goals of EDGI was adding certain Quality of Service (QoS) capabilities to desktop grids to improve task completion times. The project investigated different alternatives like redundancy with resubmission [24], however the final method chosen was resubmission with reliability based scheduling. BOINC provides information about reliability and availability of the connected (volunteer) resources that can be used to determine the most reliable resources.

These resources can be used as complementary ones, but primarily resources from private Infrastructure-as-a-Service (IaaS) clouds provided by the consortium members were utilized, with the option to scale out to public providers, like Amazon Web Services. It has been designed as a non-interactive solution, meaning that the system automates the process of task redirection for the users without any interaction from the users. Table 1 describes the available IaaS cloud resources for EDGeS@home. These are single virtual core resources, thus each resource provides a worker with a single core CPU. Each resource runs a special Virtual Appliance that contains the BOINC Client. These instances run 64 bit version Debian Linux 6.0 and have at least 2GB free space for BOINC tasks. The instances are instructed via contextualization data to attach to a given BOINC project with given user credentials.

As Table 1 shows (in its 3rd column) each private cloud has a dedicated user at EDGeS@home. The dedicated users are used to group together the resources coming from the different clouds for the reliability based scheduling. Also, the profiles of these users are public, i.e. their performance can be checked on the website of EDGeS@home.

As shown in Figure 3, the cloud resources are managed from a dedicated virtual machine ("EDGI Cloud Management") from the LPDS cloud. This management VM instructs all configured clouds. It uses an enhanced version of the solution described in [10]. The management node is responsible for starting and stopping instances as required. It is able to interact with different IaaS cloud middleware, namely OpenNebula, OpenStack, Eucalyptus and Amazon Web Services using the de facto standard Amazon EC2 interface.

Table 1. IaaS cloud resources available for EDGeS@home

Cloud Acronym	Cloud Provider	EDGeS@home User	Middleware	Number of Resources (up to)
CICA	Centro Informático Científico de Andalucía	CloudCICA	OpenNebula	100
LPDS	MTA SZTAKI LPDS	CloudLPDS	OpenNebula	64
UniMainz	University of Mainz	CloudMAINZ	OpenStack	32
UNIZAR	University of Zaragoza	CloudUNIZAR	OpenStack	50
UoW	University of Westminster	UoW	OpenStack	52
EC2	Amazon EC2	CloudAmazon	Amazon WS	∞
Total				300 +

When a new worker is requested, the URL of the BOINC project and a specific authenticator unique to each configured cloud is used as contextualization. This allows them to connect to the BOINC project. The instances remain attached to the BOINC project and continue processing tasks until they are terminated by the manager.

This way, the different configured private clouds contribute reliable resources for EDGeS@home. However, this solves only the first part of the problem. For the second part, EDGI developed a complementary Scheduler component for BOINC that reassigns problematic tasks to multiple sets of resources. This component assigns the oldest non-finished tasks in the system to one of the configured reliable resource group. In this case, a resource group is the set of resources made available from each connected private cloud.

The scheduler keeps track of the number of assigned tasks for each resource group and assigns new tasks based on the number of resources available in that group. It also provides an interface extending the administrator interface of BOINC for querying the status of assignments as shown in Figure 4. It allows filtering the jobs based on the different configured clouds (e.g. "LPDS", "UNIZAR") according to Figure 4 and applications (e.g. "DSP" in the figure) or all tasks ("All" in our figure). Reassignment is performed in round-robin order, where each configured cloud has a weight and capacity assigned and a global limit for the assignments total. The Scheduler keeps track of the total number of assigned jobs and the currently processed ones for each resource group. It continuously keeps the number of assignments at the global limit while removing finished assignments and adding new ones based on the group limit and prioritizing between resources based on the group weights.

UTILIZATION OF EDGeS@HOME BY EGI USER COMMUNITIES

After the technical solutions shown in the previous sections every obstacle of using the desktop grid technology has been eliminated and indeed EGI user communities have started to use the EDGeS@home volunteer desktop grid more and more intensively. Furthermore, these communities use EDGeS@home via different interfaces, demonstrating that the user interface flexibility we provide is indeed an important feature of supporting the user communities. There are three ways of accessing the services of EDGeS@home as shown in Figure 5.

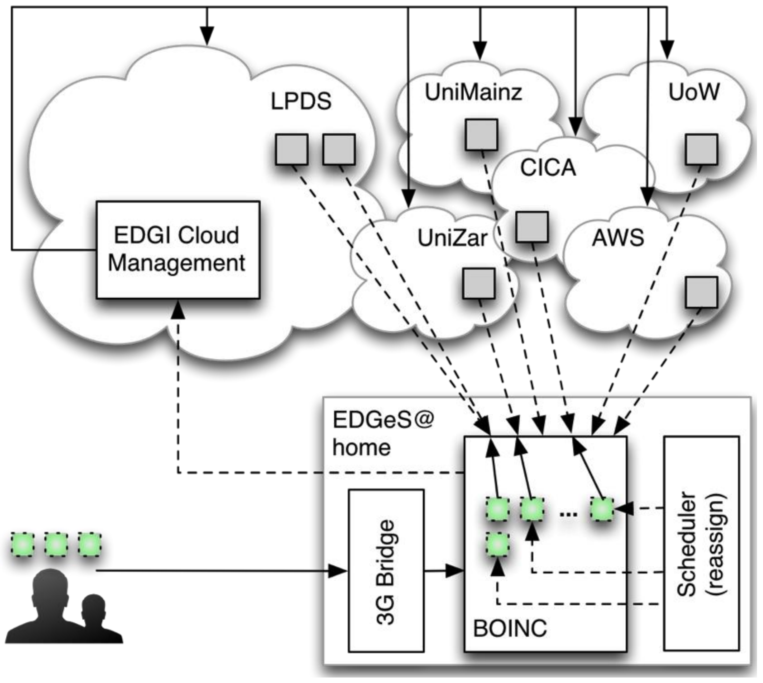
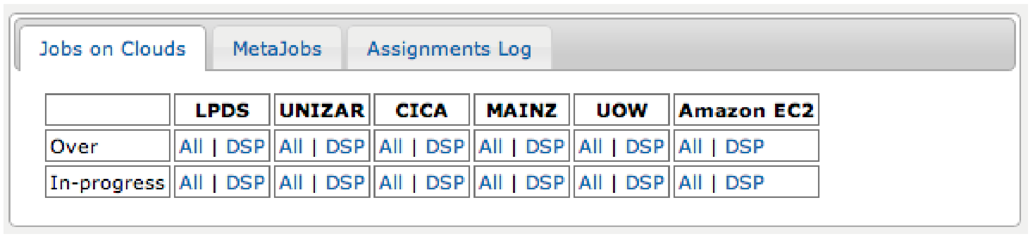


Figure 3. Task reassignment in the non-interactive QoS process.



	LPDS	UNIZAR	CICA	MAINZ	UOW	Amazon EC2
Over	All | DSP	All | DSP	All | DSP	All | DSP	All | DSP	All | DSP
In-progress	All | DSP	All | DSP	All | DSP	All | DSP	All | DSP	All | DSP

Figure 4. Information interface for querying the different desktop grid tasks (by application) assigned to the configured clouds.

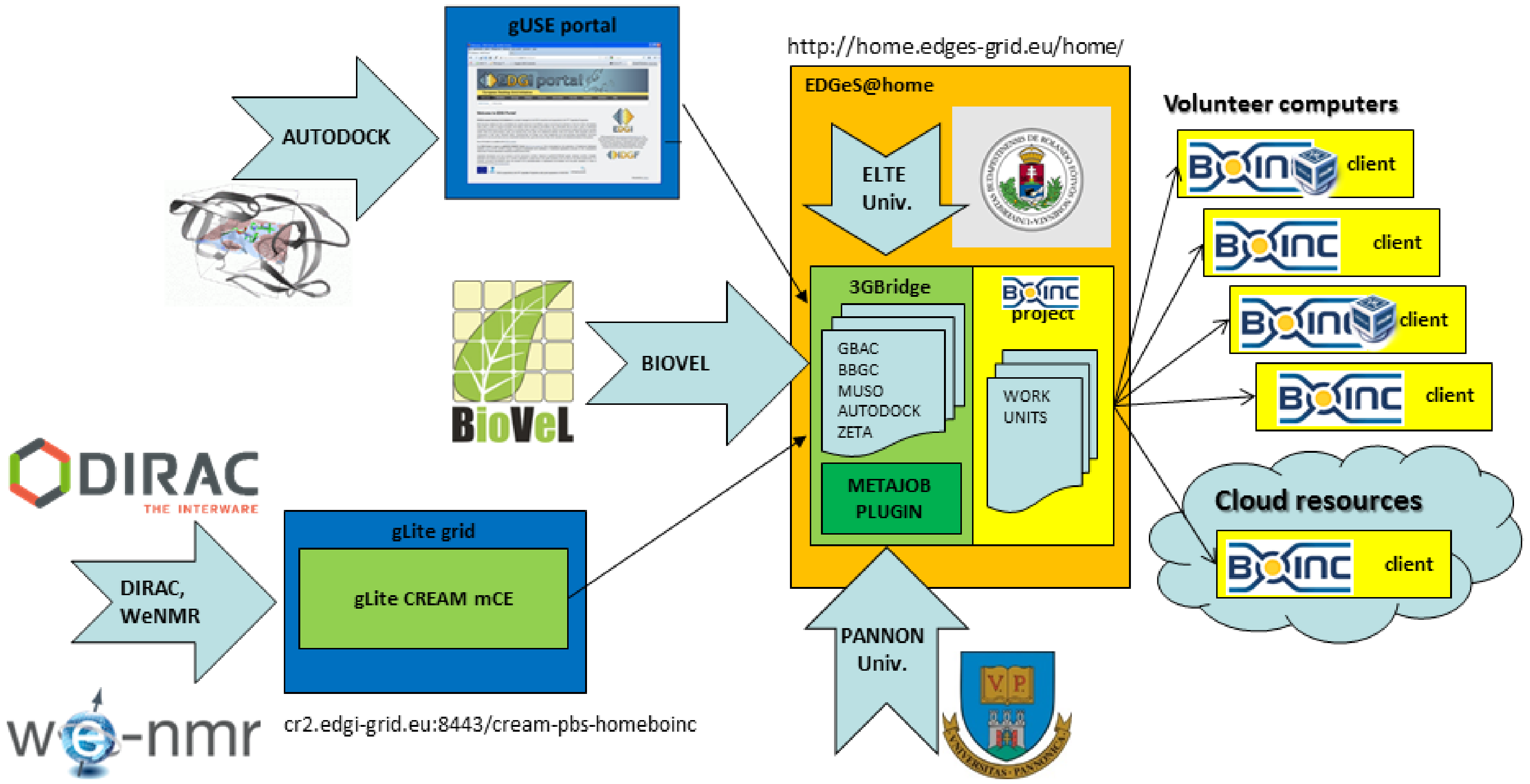


Figure 5. Overview of various access modes of EDGeS@home volunteer resources.

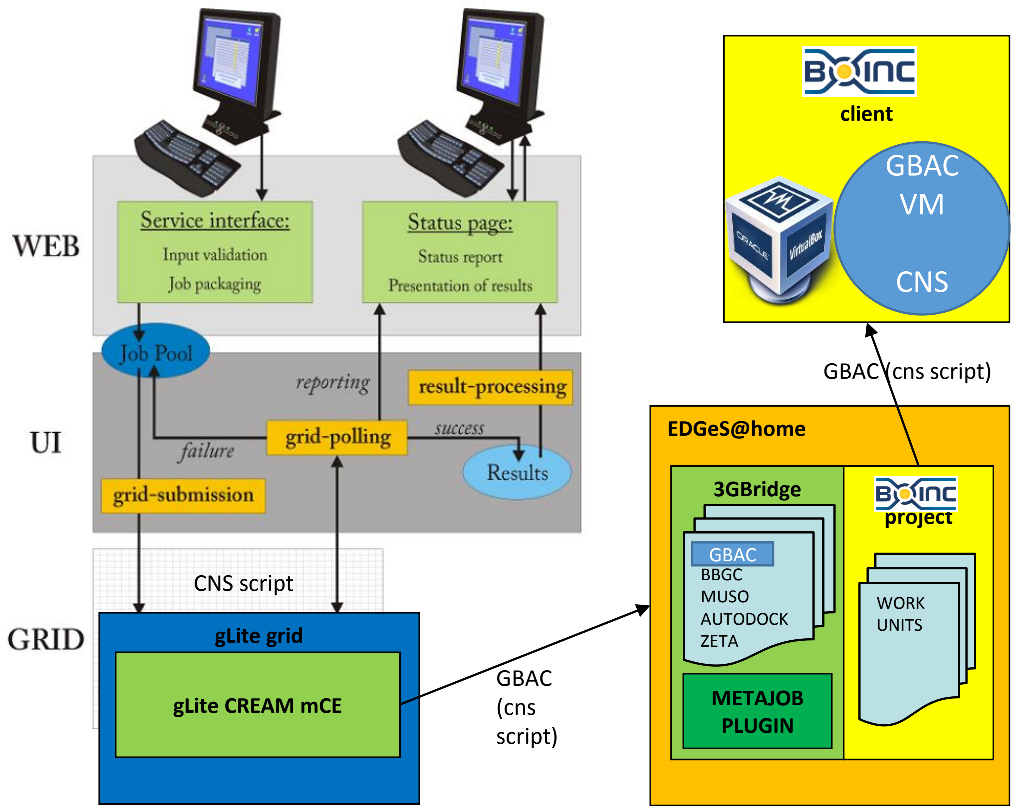


Figure 6. WeNMR HADDOCK portal: modeling of biomolecular complexes, and its connection to EDGeS@home.

One possibility of accessing EDGeS@home is from a gLite VO as we introduced before. This interface is applied by the WeNMR project [30]. They use the WeNMR HADDOCK portal for modeling of biomolecular complexes by submitting their jobs to the WeNMR VO. This VO is extended with the gLite CREAM mCE provided by the IDGF-SP project. So, the transfer of the WeNMR jobs to EDGeS@home is transparent for the WeNMR gateway users. Currently the WeNMR VO decided to send every 10th job to EDGeS@home to utilise volunteer resources beyond their own infrastructure. Notice that the utilisation of volunteer resources did not require any application porting, or any Windows version development since most of the applications are written for Linux OS, but most of the volunteer resources are MS Windows based. The architecture of supporting the execution of WeNMR jobs in EDGeS@home is shown in Figure 6.

The second option to access EDGeS@home is from a gateway that is directly connected to the 3G Bridge component of EDGeS@home. This interface possibility is used by the AUTODOCK gateway of the SCI-BUS project that was set up jointly by MTA SZTAKI and the University of Westminster. Biologists and chemists can easily do docking experiments via this gateway that is tailored for this application type. All the docking jobs defined by the users of the gateway are directly sent to the 3G Bridge component of EDGeS@home and results returned by EDGeS@home are directly shown on the gateway.

The third possible access mode to EDGeS@home is via the 3G Bridge API. This approach has been used by three projects: the EU FP7 BioVEL project [31], the Riemann zeta research project of the Eötvös Loránd University (ELTE, Budapest) [36] and the Optimiser for Linear Programming project of the Pannon University Veszprem [37]. The first one searches for values of t on the critical line, where the Riemann zeta function Z(t) is larger than a predetermined threshold, in order to get a better understanding of the behavior of the distribution of primes. In the second one they are trying to find optimal values for system solvers of Linear Programming problems by doing parameter sweep of a large number of runtime parameters. Due to the lack of space, these projects are not detailed here. The interested reader can find their descriptions in the following literature [36, 37].

CLOUD FOR AUTOMATIC DEPLOYMENT OF DESKTOP GRIDS

As described beforehand, grid VOs can easily be extended with new institutional desktop grids. Institutional desktop grids have the advantage to use resources that are already available in the institutes and hence, without investment of new hardware significant computing resources can be put together. However, to set up such institutional desktop grid requires BOINC knowledge that is usually not available in universities and other academic institutes. In order to overcome this problem we have developed a new tool called as One Click Cloud Orchestrator (OCCO) and a web-based cloud installation environment that enables for institutes and universities to quickly (in several minutes) set up their own desktop grid in a cloud without any BOINC knowledge.

One Click Cloud Orchestrator (OCCO)

Here we give a short overview of the OCCO architecture and components (see Figure 7) that allows the prompt creation of various infrastructures in the cloud including the required BOINC infrastructure, too. OCCO has the following components:(i) Automated Infrastructure Maintenance, (ii) Infrastructure Processor, (iii) Cloud Handler, (iv) VM Reshaper, and (v) Information Dispatcher. We have defined the Automated Infrastructure Maintenance component as the one responsible to understand the customized deployment descriptors. But this component does not only provide the descriptor processing capabilities but it also offers dependency resolution (so the nodes of the particular instantiated infrastructures are instantiated in a natural order), scalability and error resilience rule evaluation and enactment (so the end user does not have to intervene in its infrastructure’s internal operations). The Infrastructure Processor component of OCCO is used to ensure that the definitions of the infrastructure nodes are propagated to the VM Reshaper (which allows runtime reconfiguration of a virtual machine to meet a particular node description). In addition, the Infrastructure Processor sends such virtual machine requests to the Cloud Handler that ensures the intended role of the virtual machines after their creation. Next, the Cloud Handler is responsible for selecting a cloud infrastructure that will host a particular virtual machine, and interfacing with the infrastructure provider in a unified manner. Finally, the Information Dispatcher allows the Automated Infrastructure Maintenance component to determine the current state of the infrastructure to be used during the scaling and error resolution rule evaluation process.

In order to enable the creation of a BOINC infrastructure in a cloud we have to provide the Infrastructure Deployment Descriptor (IDD) of the target BOINC infrastructure for OCCO. The preparation of such an IDD requires some expertise and therefore we have created the required IDD. We have also created a web based UI through which user community or university representatives can initiate the creation of their BOINC infrastructure. The current version of the BOINC infrastructure we provide through this web interface is a demonstration infrastructure that connects the Autodock gateway of SCI-BUS to a BOINC infrastructure via the 3G Bridge as shown in Figure 8. As an extra functionality, the BOINC project is associated with a public IP address; therefore, the user can attach his/her own BOINC client to the server. Using automatically deployed and configured BOINC clients in virtual machines, computational resources are automatically attached to this BOINC project. Our descriptor template allows the customization of the number of computational resources. Computing jobs arrive to the BOINC project as work units with the help of the WS-PGRADE/gUSE science gateway (also automatically deployed as a node of the virtual infrastructure). Overall, the prototype shows how a complete gateway plus DCI with resources can be deployed by OCCO and how the components attach to each other. Detailed description of a similar infrastructure is shown at http://doc.desktopgrid.hu/doku.php?id= scenario:unidg with a different application.

In the prototype's welcome- and request submission page (see Figure 9) the user is not only requested to fill in the list of customization options, but he/she must also provide some details about him/her for identification and for justification. After a request is submitted, the prototype first asks for approval by the SZTAKI cloud administrators then initiates the infrastructure's creation with the Automated Infrastructure Maintenance component. Once the infrastructure is created, the notification service generates an email with all the authentication and access details to the new infrastructure (e.g., url of the science gateway and of the BOINC project plus user/password for login). With these details, users just need to login to the gateway, submit a scientific workflow (implementing molecule docking simulation based on the autodock tool) with their inputs and inspect the operation (i.e. how the jobs are flowing through the infrastructure and processed by the BOINC clients). To prevent SZTAKI's IaaS from overloading the OCCO created virtual infrastructures have a limited lifetime. Our notification service sends an email to the infrastrucure's user before the shutdown procedure is initiated. Limitation for the lifetime of the infrastructure is only applied for the demonstration infrastructure.

Notice that this service is used just for demonstration and trial purposes but the same technology can be used to set up real BOINC infrastructures with generic purpose WS-PGRADE/gUSE gateways in the cloud to which the user has access. The created BOINC infrastructure is using the GBAC technology and, as a result, no application porting is needed: the gateway/BOINC infrastructure created can immediately be used by the university or other user communities to submit and run applications in the created BOINC system.

RELATED WORK

SpeQuloS [22] aims to shorten the completion time of a batch (collection of jobs) running on a desktop grid by redirecting the workunits to cloud resources. This is similar to our proposed solution however, it requires user interaction as users must explicitly request the speed-up of their batches and must pay for it by virtual credits that can be earned by volunteering their own machine for the target desktop grid site. The more capacity they offer, the more credit they collect. The more credit they have, the more jobs can be redirected to cloud resources and the less completion time the batch will reach.

Another work targeting the elimination of the tail-effect has been introduced by the University of Westminster. The system introduced in [26] is similar to the one presented in this chapter; however, they use a different batch system (PBS) for executing the replicas of the delayed workunits in BOINC. Therefore, the delayed jobs are not handled in the frame of BOINC, but resubmitted to a separate PBS cluster that is set up on-demand when a tail has to be handled. Another difference is that they use an institutional BOINC system, not volunteer resources.

Lei Ni and Aaron Harwood [27] propose the "next generation Volunteer Computing systems on top of well studied Peer-to-Peer techniques to fully take advantage of its decentralized characteristic and its very large, shared data storage capacity". Their proposed system is based on a P2P-Tuple system where the execution of a job is done by a kind of cooperation of the peers. The paper proposes to handle the tail effect by peers pushing unfinished jobs to each other. Based on this, completion time of a job can be much lower than in a normal BOINC system.

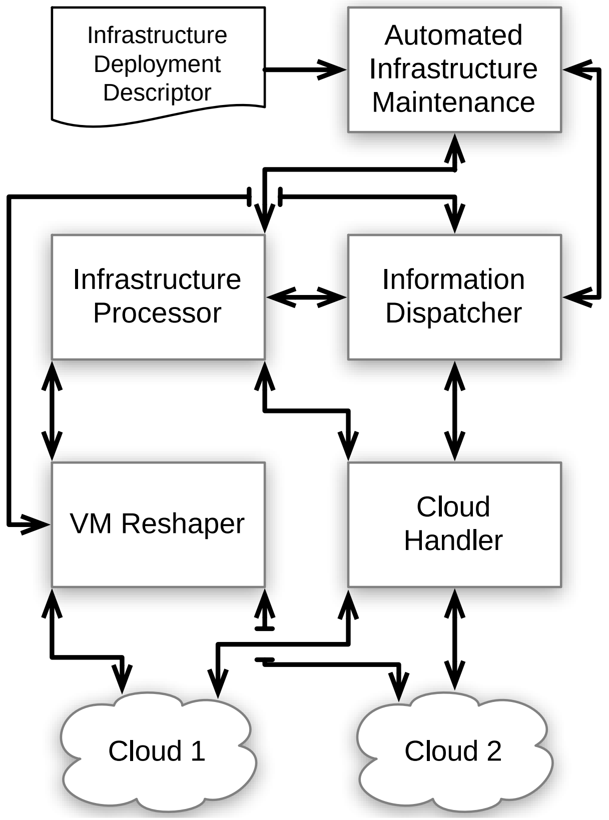


Figure 7. Architecture of OCCO.

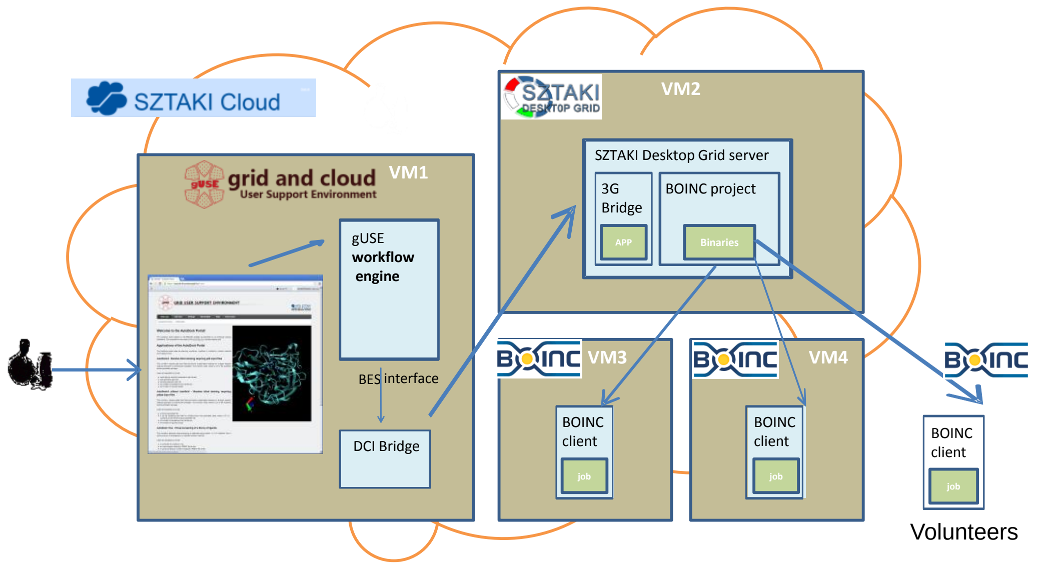


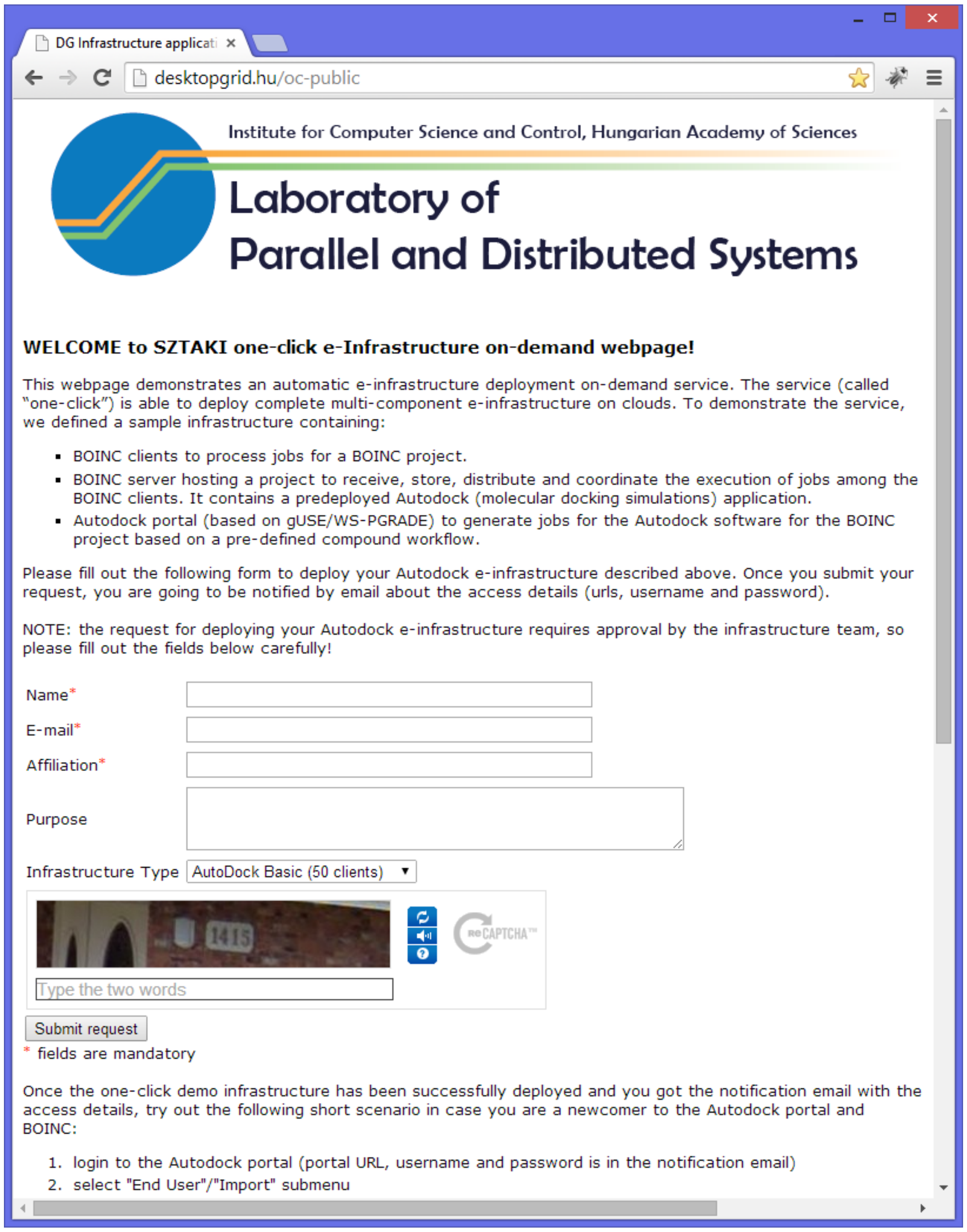
Figure 8. Infrastructure created by OCCO.

Various pilot systems like DIRAC [28] or Diane [29] are widely used by scientists these days. In these systems, a single pilot job is submitted through gLite, as a placeholder, to a given CE. The pilot job pulls real jobs from the pilot system's job repository for execution on the CE, and transfers information and results back. Pulling jobs and all communication are executed bypassing the gLite infrastructure, reducing the overhead of job submission from linear to constant time. This is also a way of decreasing the overhead of the gLite system; however, it is not targeted to use volunteer resources, and hence the number of accessible resources will be much smaller than in a volunteer system.

The notion of dynamic resource provisioning/orchestration and automatic deployment similar to the OCCO concept appears in other works, too. These solutions are general and are not aimed at supporting or extending desktop grids particularly. Chen et al. [32] present Sulcata, an on-line virtual cluster provisioning. They assume a pool of various physical resources that act as VM containers. Upon an user's request the system deploys a virtual cluster on-the-fly involving VM image preparation, VM creation and configuration and VM reboot. They solve basic functionalities of dynamic infrastructure provisioning; furthermore, the solution largely focuses on minimizing the overhead of VM image handling and deployment and proposes a resource mapping scheme. These solutions are general and are not aimed at supporting or extending desktop grids particularly.

Dörnemann et al. [33] introduced an on-demand on-the-fly deployment scheme to avoid peak loads. In a later work [34], this solution was advanced to support workflows and eliminate some shortcomings with respect to throughput and cost. To find a trade-off between task based scheduling (imprecise) and graph based scheduling (complex) of workflows, they propose a critical path based scheduling where the graph is annotated with information on anticipated run times and data transfers to calculate the makespan. Possible allocations for critical paths are predicted by genetic algorithms and the process is iterated until a mapping with minimal estimated runtime is reached. As it can be seen, in this case the goal is improving the quality of experience (as opposed to general resource provisioning in our case) by deploying services on the fly with focus on workflows (as opposed to embarrassingly parallel applications in our case).

Vukojevic et al. [35] present a similar solution to OCCO in a service oriented scenario to support simulation workflows. Their aim is to provide and redeem services on-demand according to the progress of the workflow. The core of the solution is dynamic binding with software stack provisioning. Their aims are similar to ours, the technical realization is entirely different due to the service oriented approach.


DG Infrastructure applicati
desktopgrid.hu/oc-public
Institute for Computer Science and Control, Hungarian Academy of Sciences
Laboratory of Parallel and Distributed Systems

WELCOME to SZTAKI one-click e-Infrastructure on-demand webpage!

This webpage demonstrates an automatic e-infrastructure deployment on-demand service. The service (called "one-click") is able to deploy complete multi-component e-infrastructure on clouds. To demonstrate the service, we defined a sample infrastructure containing:

- BOINC clients to process jobs for a BOINC project.
- BOINC server hosting a project to receive, store, distribute and coordinate the execution of jobs among the BOINC clients. It contains a predeployed Autodock (molecular docking simulations) application.
- Autodock portal (based on gUSE/WS-PGRADE) to generate jobs for the Autodock software for the BOINC project based on a pre-defined compound workflow.

Please fill out the following form to deploy your Autodock e-infrastructure described above. Once you submit your request, you are going to be notified by email about the access details (urls, username and password).

NOTE: the request for deploying your Autodock e-infrastructure requires approval by the infrastructure team, so please fill out the fields below carefully!

Name*	
E-mail*	
Affiliation*	
Purpose	
Infrastructure Type	AutoDock Basic (50 clients)

Type the two words

Submit request

* fields are mandatory

Once the one-click demo infrastructure has been successfully deployed and you got the notification email with the access details, try out the following short scenario in case you are a newcomer to the Autodock portal and BOINC:

1. login to the Autodock portal (portal URL, username and password is in the notification email)
2. select "End User"/"Import" submenu

Figure 9. Autodock gateway with BOINC prototype's welcome- and request submission page.

CONCLUSION

BOINC-based desktop grid systems are excellent in collecting large number of volunteer resources worldwide. Despite their cost-effectiveness, there were several problems that prevented their widespread use in e-science environments. Recent innovations in virtualization and cloud technology helped us to overcome these major problems. Overall, we can claim that all the major obstacles witnessed in the former use of BOINC systems have been eliminated. Our GBAC technology eliminates the need of porting applications to BOINC and the integration of clouds with BOINC systems solves the tail problem in volunteer desktop grids. As a result, more and more user communities start to use the EDGeS@home BOINC desktop grid either via a gLite VO (e.g. WeNMR), or via a gateway (e.g. biologists and chemists using autodock), or directly (e.g. BioVEL project). We expect more radical increase in use of BOINC systems in the near future, when the recently introduced one-click creation method of BOINC systems in clouds will be widely applied by scientists.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no 261556 (EDGI), no. 283481 (SCI-BUS) and under grant agreement no 312297 (IDGF-SP).

REFERENCES

[1] Cristina Aiftimiei, Paolo Andreetto, Sara Bertocco, Simone Dalla Fina, Alvise Dorigo, Eric Frizziero, Alessio Gianelle, Moreno Marzolla, Mirco Mazzucato, Massimo Sgaravatto, Sergio Traldi, Luigi Zangrando, *Design and implementation of the gLite CREAM job management service*, Future Generation Computer Systems, Volume 26, Issue 4, April 2010, Pages 654-667, ISSN 0167-739X, 10.1016/j.future.2009.12.006.

[2] Streit, P. Bala, A. Beck-Ratzka, K. Benedyczak et al., *Unicore 6: Recent and future advancements*, Berichte des Forschungszentrums Julich, vol. 65, 2010.

[3] Globus Toolkit Version 4: Software for Service-Oriented Systems. I. Foster. *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, pp 2-13, 2006.

[4] M. Ellert, M. Grønager, A. Konstantinov, B. Konya, J. Lindemann, I. Livenson, J. Nielsen, M. Niinimaki, O. Smirnova and A. Waananen, *Advanced resource connector middleware for lightweight computational grids*, Future Generation Computer Systems, vol. 23, no. 2, pp. 219-240, 2007.

[5] David P. Anderson 2004. BOINC: A System for Public-Resource Computing and Storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID '04)*. IEEE Computer Society, Washington, DC, USA, 4-10. DOI=10.1109/GRID.2004.14 http://dx.doi.org/10.1109/GRID.2004.14.

[6] A. C. Marosi, Z. Balaton & P. Kacsuk (2009). GenWrapper: A generic wrapper for running legacy applications on desktop grids. In *2009 IEEE International Symposium on Parallel & Distributed Processing* (pp. 1-6). IEEE. doi:10.1109/IPDPS.2009. 5161136.

[7] A. C. Marosi, G. Gombás, Z. Balaton & P. Kacsuk. (2008). *Enabling Java applications for BOINC with DC-API.* In P. Kacsuk, R. Lovas, & Z. Németh (Eds.), Distributed and Parallel Systems SE-1 (pp. 3-12). Springer US. doi:10.1007/978-0-387-79448-8_1.

[8] *The EDGI EU FP7 project:* http://edgi-project.eu.

[9] P. Kacsuk, J. Kovacs, Z. Farkas, A. Marosi and Z. Balaton Towards a powerful european DCI based on desktop grids. *Journal of Grid Computing*, 9:219-239, 2011.

[10] Attila Marosi, József Kovács, Peter Kacsuk, *Towards a volunteer cloud system, Future Generation Computer Systems,* Volume 29, Issue 6, Pages: 1442-1451, 2013, ISSN 0167-739X, http://www.sciencedirect.com/science/article/pii/S0167739X12000660, 10.1016/j.future.2012.03.013.

[11] Ádám Visegrádi, József Kovács, Peter Kacsuk: *Efficient extension of gLite VOs with BOINC based desktop grids,* Future Generation Computer Systems, Volume 32, March 2014, Pages 13-23, ISSN 0167-739X, http://dx.doi.org/10.1016/j.future.2013.10.012, (http://www.sciencedirect.com/science/article/pii/S0167739X1300229X).

[12] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Comp. Syst.* 25(6): 599-616 (2009).

[13] C. U. Soettrup, A. Waananen, J. Kovacs, Transparent execution of ARC jobs on Desktop Grid resources, MIPRO 2012, *Proceedings of the 35th International Convention IEEE Conference Publications Croatian Society for Information and* Communication Technology, Electronics and Microelectronics, MIPRO Rijeka, Croatia 2012 pp. 271-276, ISBN: 978-953-233-072-4.

[14] M. Keller, J. Kovacs, A. Brinkmann Desktop Grids Opening up to UNICORE, *Proceedings of UNICORE Summit 2011*, IAS Series 09, Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag, Torun, Poland 2011 pp. 67-76 ISBN: 978-3-89336-750-4.

[15] *The EDGeS EU FP7 project:* http://edges-grid.eu.

[16] *The IDGF-SP EU FP7 project:* http://idgf-sp.eu.

[17] Z. Farkas, P. Kacsuk, Z. Balaton, G. Gombás, *Interoperability of BOINC and EGEE, Future Generation Computer Systems*, Volume 26, Issue 8, October 2010, Pages 1092-1103, ISSN 0167-739X, 10.1016/j.future.2010.05.009. (http://www.sciencedirect. com/science/article/pii/S0167739X10000890).

[18] Gabor Terstyanszky and Tamas Kiss and Tamas Kukla and Zsolt Lichtenberger and Stephen Winter and Pamela Greenwell, and Sharron McEldowney and Hans Heindl (2012) Application repository and science gateway for running molecular docking and dynamics simulations. Healthgrid applications and technologies meet science gateways for life sciences. *Studies in health technology and informatics* (175). IOS Press, pp. 152-161. ISBN 9781614990536.

[19] *The European Grid Initiative,* http://www.egi.eu.

[20] *WMIN DG:* http://wgrass.wmin.ac.uk/index.php/Desktop_Grid:Westminster_Local_ DG.

[21] *The online documentation site:* http://doc.desktopgrid.hu.

[22] Simon Delamare, Gilles Fedak, Derrick Kondo, Oleg Lodygensky, SpeQuloS. A QoS Service for BoT Applications Using Best Effort Distributed Computing Infrastructures, in *International Symposium on High Performance Distributed Computing* (HPDC'2012), Delft, Nederlands, 2012.

[23] Javadi, B., Kondo, D., Vincent, J.-M., Anderson, D. P., *Discovering Statistical Models of Availability in Large Distributed Systems:* An Empirical Study of SETI@home, Parallel and Distributed Systems, IEEE Transactions on, vol. 22, no. 11, pp. 1896-1903, Nov. 2011. doi: 10.1109/TPDS.2011.50.

[24] Pataki, M., Marosi, A. C. (2012). Searching for Translated Plagiarism with the Help of Desktop Grids. *Journal of Grid Computing,* 1-18. http://dx.doi.org/10.1007/s10723-012-9224-5.

[25] Gabor Terstyanszky, Tamas Kiss, Tamas Kukla, Zsolt Lichtenberger, Stephen Winter, Pamela Greenwell, Sharron McEldowney and Hans Heindl, Application Repository and Science Gateway for Running Molecular Docking and Dynamics Simulations, in Sandra Gesing et al., *Editors Health Grid Applications and Technologies Meet Science Gateways for Life Sciences, Studies* in Health Technology and Informatics, Vol. 175, pp152-161, IOS Press, 2012, ISSN 0926-9630 (print), ISSN 1897-8365.

[26] Reynolds, C. J., Winter, S., Terstyanszky, G. Z., Kiss, T., Greenwell, P., Acs, S., Kacsuk, P., Scientific Workflow Makespan Reduction through Cloud Augmented Desktop Grids, Cloud Computing Technology and Science (CloudCom), 2011 IEEE *Third International Conference* on, pp. 18-23, Nov. 29 2011-Dec. 1 2011, doi: 10.1109/CloudCom.2011.13.

[27] Lei Ni, Aaron Harwood, P2P-Tuple: Towards a Robust Volunteer Computing Platform pdcat, *International Conference on Parallel and Distributed Computing,* Applications and Technologies, 2009, pp. 217-223.

[28] Tsaregorodtsev et al., DIRAC: a community grid solution, *J. Phys. Conf. Ser.* 119 (2008) 062048.

[29] Vladimir V. Korkhov, Jakub T. Moscicki and Valeria V. Krzhizhanovskaya, 2009. *Dynamic workload balancing of parallel applications with user-level scheduling on the Grid. Future Gener. Comput. Syst.* 25, 1 (January 2009), 28-34. DOI=10.1016/j.future.2008.07.001 http://dx.doi.org/10.1016/j.future.2008.07.001.

[30] *The WeNMR community,* https://www.wenmr.eu.

[31] *The BIOVEL community,* http://www.biovel.eu.

[32] Yang Chen, Tianyu Wo and Jianxin Li. An efficient resource management system for on-line virtual clusterprovision. *In IEEE CLOUD,* pages 72-79, 2009.

[33] Tim Dörnemann, Ernst Juhnke and Bernd Freisleben. *On-demand resource provisioning for bpel workflows using amazon'selastic compute cloud.* In IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGrid), pages 140-147, 2009.

[34] Tim Dörnemann, Ernst Juhnke, Thomas Noll, Dominik Seiler and Bernd Freisleben. *Data flow driven scheduling of bpel workflows using cloud resources.* In IEEE CLOUD, pp. 196-203, 2010.

[35] Karolina Vukojevic-Haupt, Dimka Karastoyanova, Frank Leymann., On-demand Provisioning of Infrastructure, Middleware and Services for Simulation Workflows. *SOCA* 2013: 91-98.

[36] http://riemann-siegel.com/.

[37] Péter Tar, István Maros, Parameter sweep of a linear programming solver on distributed computing infrastructures, presented at: *17th Spring Wind Conference,* Debrecen, 2014. March 21-23.

P. K.