PROCEEDINGS OF SCIENCE

Porting Computation and Data Intensive Applications to Distributed Computing Infrastructures Incorporating Desktop Grids[1]

Tamas Kiss
University of Westminster
115 New Cavendish Street, London, UK, W1W 6UW
E-mail: kisst@wmin.ac.uk

Ian Kelley
Cardiff University
5 The Parade, Cardiff, UK, CF24 3AA
E-mail: I.R.Kelley@cs.cardiff.ac.uk

Peter Kacsuk
MTA SZTAKI
Kende u.13, Hudapest, Hungary, H-1111
E-mail: kacsuk@sztaki.hu

Distributed Computing Infrastructures (DCIs) are widely utilised by scientists to support computation and data intensive application scenarios. The European EDGI project is working on connecting two distinctive types of DCIs, Service and Desktop Grid systems. The infrastructure is operated at production level and supports a large variety of applications. Porting and deploying an application on such combined platform raises specific challenges. This paper summarises the experiences that were gained during the porting and migration of over 30 applications to the EDGI combined service grid/desktop grid (SG/DG) platform. Computation and data intensive application scenarios are analysed, and tools and solutions are suggested to address specific requirements of application porting to a DCI that is extended with local or volunteer desktop grid resources.

[1] This work is supported by the EDGI (European Desktop Grid Initiative) project funded by the European Commission within the FP7 framework (project number RI 261556).

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

PoS(ISGC 2011 & OGF 31)060

PROCEEDINGS OF SCIENCE

The International Symposium on Grids and Clouds and the Open Grid Forum
Academia Sinica, Taipei, Taiwan
March 19 - 25, 2011

PoS(ISGC 2011 & OGF 31)060

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike Licence.

1. Introduction

Although current grid infrastructures offer significant amount of resources to run computation and data intensive applications, some scenarios overgrow the capabilities of existing production level grid systems. These applications require resources from different grids raising the need for seamless interoperation between heterogeneous platforms. The European EDGeS (Enabling Desktop Grids for e-Science) [12] and its follow up the EDGI (European Desktop Grid Initiative) project [27] have contributed to this work by developing a bi-directional bridging mechanism that seamlessly connects the two main types of grid infrastructures, service and desktop grid systems. Before this work service and desktop grid infrastructures evolved independently from each other providing no support for applications that wanted to utilize resources from both grid types.

Service grids (SG) provide reliable 24 hour service operated and looked after by professional system administrators. A resource, typically a computing cluster, becomes part of a service grid by installing and running a highly complex middleware package, for example the Globus Toolkit [1], gLite [2], ARC [3] [10] or Unicore [4]. Although these service grid systems provide guaranteed service, the number of resources offered by them is restricted to the magnitude of a few tens of thousands of processors. Examples for production level SG infrastructures include the gLite, ARC and Unicore based EGI grid [9], or the Globus based Open Science Grid [11].

Desktop grid (DG) systems, on the other hand, are capable to offer computing resources in the magnitude of millions of PCs. The resources are donated by individuals or institutions and their computing power can be utilized for desktop grid computations whenever the processors are idle. However, this also means that DG systems come with no guarantees concerning their quality of service. Examples for DG middleware include BOINC [5] that powers several large public desktop grids such as the SETI@HOME project [6] or the SZTAKI Desktop Grid [7], and XtremWeb [8].

EDGeS developed a bi-directional bridge between the gLite based EGEE Grid [13], and BOINC and XtremWeb based DG systems. The bridge allowed DG work units to utilize idle resources in specified virtual organizations of the EGEE, and also offered EGEE users the capability to select validated applications from the EDGeS Application Repository and run them on both EGEE and DG resources. EDGI further extends the gLite to DG direction to other middleware supported by the European Grid Infrastructure, namely ARC and Unicore, and aims to improve the quality of service of desktop grid systems by extending them with Cloud resources on demand, and concentrates on supporting not only computation but also data intensive applications.

The EDGeS/EDGI bridge infrastructure [14] [15] and its supporting components for distributed data handling [16] have been described in several publications. This current paper concentrates on use-case scenarios and types of applications that could effectively utilize the EDGI infrastructure. The paper summarises all those experiences that were gained through the porting and deploying of over 30 applications to combined SG/DG infrastructures.

PoS(ISGC 2011 & OGF 31)060

2. Requirements towards applications to be run on SG/DG infrastructure

Combining resources of service and desktop grid infrastructures significantly increases the number of available resources that applications are able to utilise. Larger number of resources could result in higher throughput and better performance. However, using a combined infrastructure also imposes some limitations regarding the applications. In case of a combined SG/DG infrastructure the application needs to be executed on both types of grid systems. As SG infrastructures are capable to run a wider range of applications than DGs, it is typically the desktop grid system that limits this range of target applications. If an application is capable to run on DG resources then it can typically be executed on SG resources too. In this section an overview of requirements is presented towards target applications for SG/DG infrastructures.

One of the most important restrictions is the type of parallelisation implemented by the applications. Worker nodes of a desktop grid are typically isolated allowing no direct communication between the workers. As a consequence, applications requiring inter-process communication (e.g., MPI applications) are not suitable for DGs. Although there some promising experiments for running MPI applications on desktop grids incorporating Graphical Processing Units (GPUs), these experiments are currently limited by the number and quality of scientific MPI code that is GPU enabled.

Data handling is another important and limiting factor when running applications on SG/DG resources. The bandwidth available in a public desktop grid is limited due to the centralized input source and distribution mechanisms of desktop grid middleware. Unlike in a cluster or traditional service grid, where shared disks are available with high-speed interconnects, in a desktop grid, input files need to be distributed over the Internet to each and every volunteer node. Therefore, the feasible size of input or output files to be downloaded or uploaded by a worker in a volunteer DG is typically not larger than 100 megabytes. In local DG systems, however, this file-size could be significantly larger due to faster local network connections, allowing even gigabyte sized input and output files. The Attic peer-to-peer data distribution system [17] [18], developed in the EDGeS and EDGI projects, was built to mitigate this problem of distributing larger files on volunteer desktop grids. By utilizing multiple download endpoints and file replicas to distribute data, Attic can significantly increase the feasibility of data intensive application scenarios, as will be demonstrated later in this paper.

Confidentiality of data can also be an issue in public DGs. As the data is downloaded to computers of untrusted volunteers, public DGs may not be able to provide the level of data protection required by many applications. A local desktop grid infrastructure could provide a solution for the data confidentiality problem as the data is not passing company or institutional firewalls in this scenario.

Besides the above limiting restrictions, there are also a few generic recommendations that need to be applied in order to achieve higher performance. The execution time of individual work units of a DG application need to be well balanced. Also, very short workunits, in the range of few seconds or minutes, could significantly increase the overhead and decrease performance. On the other hand, workunits that are running for several hours are likely to be interrupted by the donors. In case of workunits running for a couple of hours or longer, it is essential that the application developer implements internal check-pointing allowing the

PoS(ISGC 2011 & OGF 31)060

application to seamlessly resume after user interruption. Finally, as a very large proportion of volunteer computers are running some flavour of the Windows operating system (when compared to the typically Linux based computing clusters), in most cases it is essential to develop a Windows (and/or MacOS) based client applications.

3. Application Scenarios for a Combined SG/DG Infrastructure

Requirements towards porting to and running an application on an SG/DG infrastructure could differ depending on the primary and secondary target platform. The primary target platform is the one that is directly utilised by the end-user, while the secondary platform is used in a user-transparent way utilising the EDGI bridging mechanisms. Based on primary and secondary target computing platforms, three generic user scenarios have been identified and analysed. In this section, besides the description of these scenarios, justification for their feasibility and usefulness is given, and illustrated via examples. Differences in the application porting process regarding the different scenarios are also highlighted.

3.1 Applications Running through the DG to SG Bridge

In this scenario, illustrated on Figure 1, desktop grid applications can utilize resources from a service grid system, for example from a specific Virtual Organization (VO) of the EGI grid, via the DG to SG bridge. The scientific end-user in this scenario runs a DG application, and the utilization of SG resources is completely transparent from the users' point of view. The DG to SG bridge acts as a powerful DG worker pulling work units to the dedicated service grid VO and sending the results back to the desktop grid server.

This scenario is useful when the end-user has access to a relatively small local or institutional desktop grid system that can significantly be extended with service grid resources. An example for this scenario is the ViSAGE video stream analysis application by Correlation Systems Limited [22]. Local users can access the application via a custom user interface and can analyse pairs of frames in a video recording on the local BOINC based desktop grid system. However, the Correlation Systems DG connects approximately 20 local PCs only. Therefore, sending jobs to the EGI DG VO that contains thousands of processors can significantly speed up the computation.

The application porting and deployment process for scenario 1 is relatively straightforward. If a DG version of the application exists and runs on a public desktop grid then it is very likely that the worker applications are supporting a wide range of Linux flavours including the one applied by the target SG platform. If the application is running on a local desktop grid it may be necessary to develop Linux worker applications that can be submitted to the target SG platform.

Once an executable for the target SG platform has been compiled and tested, the host DG system need to be connected to the DG to SG bridge. Setting up this connection includes the registration of the local DG and the target application with the bridge. The owner of the DG application also requires a valid proxy certificate as the most common mechanism for user authentication in service grid systems. This certificate is utilised by the bridge when submitting the application to SG resources.

PoS(ISGC 2011 & OGF 31)060

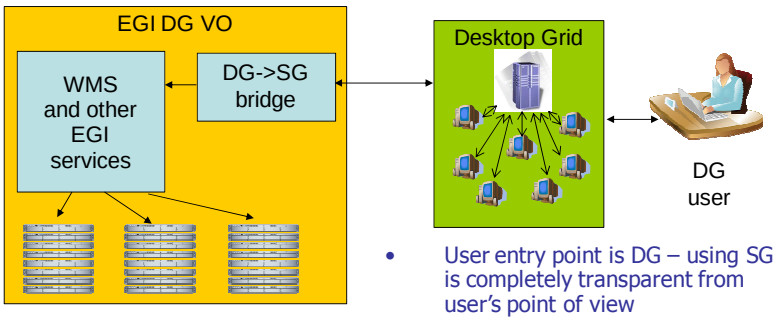

Figure 1 – Applications running from DG to SG

3.2 Applications Running through the SG to DG Bridge

The second scenario is illustrated on Figure 2. The scientific end-user in this scenario accesses the SG platform directly and submits the application to service grid resources using the standard job submission mechanism of the target platform. For example, in case of a gLite based grid the user utilizes a gLite User Interface machine and prepares a suitable JDL file for the submission. If the virtual organization where the job has been submitted includes an EDGI modified computing element then jobs sent to this computing element will be executed on DG resources using the SG to DG bridge.

Using desktop grid resources could significantly increase the processing power of service grid systems by extending them with potentially millions of worker nodes. As high percentage of current SG applications are parameter sweeps, these applications can be efficiently redirected to DG resources freeing up SG systems for more specific MPI applications, for example.

Several EGI user communities have been supported by the EDGeS and EDGI projects to port and run their applications through the SG to DG bridge. An example for these efforts is the VisIVO (Visualization Interface to the Virtual Observatory) application [23]. VisIVO is a suite of software tools for creating customized views of 3D renderings from astrophysical data tables. The application previously run on gLite based service grid systems and has been ported to the SG to DG bridge.

Porting an application that utilises the SG to DG bridge starts with the development of a desktop grid version of the application. As the application is submitted from the service grid, this side is responsible for the creation and orchestration of parameter sweep jobs or work units. It is only a desktop grid client application that needs to be developed. However, as it was analysed in section 2, this client is ideally compiled to different target platforms including Windows and MAC OS.

The development of the client is supported by several high level tools and APIs provided by the EDGI project. These tools include the Distributed Computing API (DC-API) that enables to execute the ported application on multiple desktop grid middleware (e.g. BOINC, XtremWeb or Condor) [24] without any modification, and the Generic Wrapper (GenWrapper) tool that facilitates the porting of legacy applications onto BOINC based desktop grid platforms [25].

Once the application is ported to the target DG platform it needs to be validated. Validation is required due to the different security models of SG and DG platforms. While SGs trust the user and require certificates, DG systems trust the application. The validation process aims to assure that the validated application is causing no harm to the desktop grid donors.

PoS(ISGC 2011 & OGF 31)060

Once validated, the application is published in the publicly available EDGI Application Repository (EDGI AR). This repository is accessed by three different types of actors. Desktop grid administrators can browse the repository and download the DG version of the applications that they aim to support. service grid users can find and download the SG version of the application. Finally, the SG to DG bridge uses the reference provided by the user to one of the deployed applications in the repository, and enables the bridging of this application to the target DG systems.

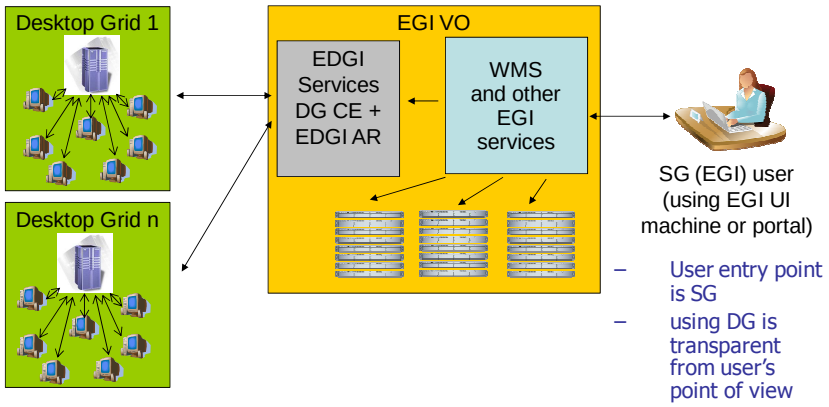

Figure 2 – Applications running from SG to DG

3.3 Applications Using Specific Job Submission or Scheduling Systems to Utilize to SG/DG Resources

There are use case scenarios when the target application already utilises specific high level user environments or lower level middleware and job submission frameworks. In these cases application developers and end-users may wish to exploit both desktop and service grid resources without compromising the current user experience. Exploiting the EDGI bridges directly may not be suitable in these scenarios because of some preliminary assumptions regarding the way SG or DG jobs are submitted.

Besides the production bridges, EDGI also provides building blocks and components of these bridging solutions that could be integrated into other grid middleware or user environments. This integration assures that the user community of the target middleware or user environment will utilise the combined SG/DG infrastructure in a transparent way without changing the frameworks they accustomed to.

An example for this scenario is illustrated on figure 3. The Wisdom production environment [26] has been extended with a DG submitter module. The WISDOM project is an international initiative to enable a virtual screening pipeline on a grid infrastructure. The project has developed its own meta-middleware that utilises the EGI production infrastructure, and capable to submit and execute very large number of jobs on EGI resources using a pilot job submission mechanism. The newly developed DG submitter uses EDGI components, such as the 3G Bridge and its WS Submitter to access desktop grid Resources. The DG submitter pulls WISDOM jobs from the WISDOM task manager exactly the same way as in case of EGI jobs. Therefore, the DG submission did not require modification of the original WISDOM architecture and completely transparent from the end-users' point of view.

Application porting in this case includes the implementation and validation of a desktop grid version of the application, similarly to scenario 2. Instead of the EDGI application repository, the middleware may use its own repositories to store and submit applications and work units. In the presented example the WISDOM Job and Task managers are responsible for this functionality.

PoS(ISGC 2011 & OGF 31)060

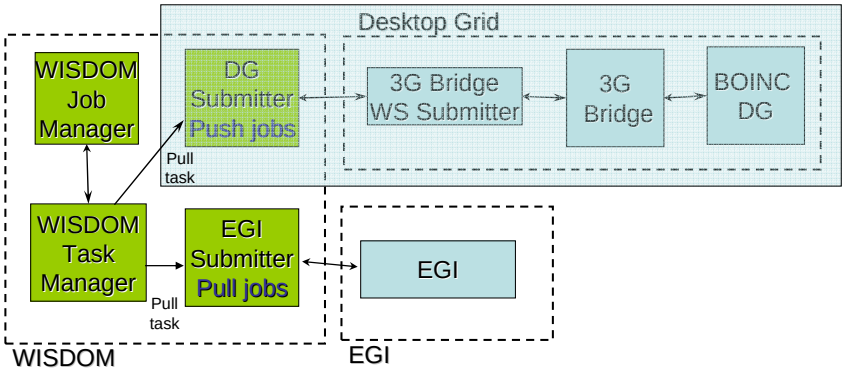

Figure 3 - Extending the WISDOM environment with a DG submitter

4. Supporting Data Intensive Application Scenarios

In current desktop grid scientific volunteer computing software infrastructures, such as BOINC and XtremWeb, data is distributed centrally from a project's coordinating nodes or servers. In BOINC, this is achieved through a set of HTTP mirrors, each providing clients with full copies of data input files. Similarly, in XtremWeb, clients are given the URIs of data input files. These centralized systems require projects not only to have the necessary network capacity needed to provide data to all volunteers, but also to have data readily available and persistent on their servers at all times to fulfil client requests. Further, the network throughput requirements of serving so many client machines can prove to be a hindrance to projects wishing to explore new types of data-intensive application scenarios that are currently prohibitive in terms of their large data transfer needs. For example, the SETI@Home project, which has very small work unit sizes (340KB per work unit, with an average processing time of two hours) averages about 50 Mbp/s of download and serves over 16 Petabytes a month. Each SETI@Home job is replicated to two unique workers, meaning the bandwidth consumption could theoretically be cut in half if files were shared among the network participants.

4.1 Desktop grid data distribution using Attic

An alternate approach to the centralized systems currently employed by desktop grids is to make use of peer-to-peer (P2P) techniques to implement data distribution. The application of using P2P in volunteer computing as a means to offload the central network has been explored in [19]. There are many ways this could be implemented, ranging from a BitTorrent-style network [20], where data is centrally tracked and all participants share relatively equal loads, to KaZaa-like super-peer networks [21] where select nodes are assigned greater responsibility in the network.

Attic, as shown in Figure 4, provides a solution where data is *pushed* to a P2P environment by the service grid nodes that have the necessary security credentials to access the data from the their servers. This not only alleviates the security problems associated with somehow allowing untrusted users access to service grid storage elements, but it also offloads the data distribution, making the integration of service and desktop grids more accessible.

PoS(ISGC 2011 & OGF 31)060

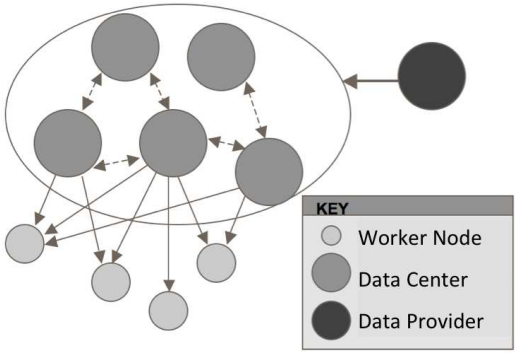

Figure 4 - Attic network roles

Attic network agents can be logically broken up into three key entities: *data providers*, *data centers* and *workers (data consumers)*. This 3-tiered, bridged architecture involves the following high-level interactions. The *data provider* pushes files to an overlay network of *data center*s, which self-organize using P2P techniques to propagate data amongst itself. This data center layer then serves pull requests for data from the *workers* that process jobs on the network. A worker node discovers data servers by sending a request to known access points on the network (likely members of the data center overlay) that keep track of which data centers have registered replicas of the file.

4.2 Using Attic to distribute service grid data on desktop grid nodes

The EDGI project requires a system that can adapt to varying input file sizes and replication factors without unduly stressing or exposing the service grid layer. Service grid security infrastructure and policies prevent access to local files from foreign and untrusted hosts. Anonymous access is generally not an issue for most BOINC projects, as they are able to have dedicated and network-isolated data servers.

By having a more robust data distribution framework, applications can have input files in the gigabytes, not the megabytes scale, which is required for many of the EDGI applications (e.g., analyzing medical imagery and searching large protein databases).

To adapt a current (EDGI-supported) service grid job to use Attic, the input files are published to the Attic system using the Attic tools, either a direct Java library, or a command-line interface that only requires *curl*, a common tool found on most clusters, be installed. When a file is published, certain metadata is attached to it, such as the MD5 [28] sums, its size, the project it is associated with, an initial seed endpoint from where the file can be downloaded, and the desired replication factor. The Attic software then registers the file, returns a unique attic ID and URL, and begins to replicate the file on the network.

The service grid user (or middleware) can then submit their service grid job through the EDGI 3G Bridge to run on a desktop grid, as normal. The only change is that *attic://* input file locations are also sent with their corresponding MD5 hashes. For XtremWeb jobs, no further information or adaptation is needed. XtremWeb can download Attic files without modification to the core client, due to its inclusion of the native-Java Attic protocol handler, which only required a few additional lines of code to register the URL-handler.

PoS(ISGC 2011 & OGF 31)060

For BOINC, however, the process is slightly more complex. In lieu of modifying the BOINC client to directly recognize *attic://* files, which would have required installation of a new version of BOINC on every client machine in a project wishing to utilize Attic, a new "Attic Proxy" project was created that intercepts Attic requests from BOINC projects, retrieves the Attic files, and passes the resulting file-stream directly back to the BOINC project. All a BOINC project or user has to do, is "subscribe" to the Attic Proxy project in addition to their other BOINC projects. The Attic Proxy then runs in the background (within BOINC) and translates the Attic file requests as they come in.

This proxying mechanism is made possible by first having the 3G Bridge rewrite the attic:// URLs it receives to be the standard HTTP URLs that BOINC expects and can retrieve. However, the HTTP URLs are re-written to reference a local webserver on the client machine running on an unused port. The Attic Proxy project starts a lightweight webserver on this port that intercepts these http://localhost/atticURL requests, translates them into their fundamental attic:// requests and downloads the files. Since the Attic Proxy (dubbed libafs, and written in ANSI C) knows how to interpret the *attic://* protocol, the burden of having Attic libraries is then moved to the Attic Proxy project, and doesn't become a requirement of BOINC or the BOINC projects. Libafs downloads the files, and then returns the resulting file stream directly to the BOINC project via the localhost HTTP request. In essence, the Attic Proxy acts exactly like a traditional web proxy – it retrieves the given files, and sends them back to the client as requested.

Using the Attic Proxy for BOINC, or the native Java-protocol handler in XtremWeb, it is possible to publish large files to Attic, launch a job through the 3G Bridge, and run these jobs on the volunteer desktop grid nodes, without any modification to the desktop grid middleware. By leveraging this new technology, projects can now distribute large input files to many volunteer nodes, without stressing their centralized systems. The burden of data distribution is moved to Attic, which can more easily balance network loads, allowing projects to add nodes that act as partial replicas rather than full mirrors, and, depending on their security policies, to utilize volunteer resources to even host the Attic *data centers.*

5. Conclusion and future work

Within the framework of the EDGeS and EDGI projects, several applications have been ported to combined SG/DG infrastructures. The experience gained, methodologies utilised and tools required for this specific application porting scenario is summarised in this paper. The results presented are a good starting point for future application developers when porting applications to this specific platform.

The application support service of the EDGI project will continue its work on supporting user communities and porting their applications to this infrastructure. The emphasis will slowly be shifted from only computationally intensive applications to more data intensive scenarios, as it was described in this paper. Also , the EDGI platform is currently being extended with Cloud Computing based quality of service guarantees which also raises specific challenges to target applications.

PoS(ISGC 2011 & OGF 31)060

References

[1] Foster, I., Globus Toolkit Version 4: Software for Service-Oriented Systems. Journal of Computer Science and Technology 21, no. 4, 513–520, 2006.
[2] Laure, E. et al., Programming the Grid with gLite. Computational Methods in Science and Technology 12, no. 1, 33-45, 2006.
[3] The ARC Grid Middleware, http://www.nordugrid.org/middleware/
[4] D. Erwin: UNICORE - A Grid Computing Environment Concurrency, Practice and Experience Journal, 14, 2002, pages 1395-1410
[5] D. P. Anderson: BOINC: A System for Public-Resource Computing and Storage. 5th IEEE/ACM International Workshop on Grid Computing, November 8, 2004, Pittsburgh, USA.
[6] D.P. Anderson et al.: SETI@home: An Experiment in Public-Resource Computing, Communications of the ACM, Vol. 45 No. 11, November 2002, pp. 56-61
[7] Z. Balaton, et al.: SZTAKI Desktop Grid: A Modular and Scalable Way of Building Large Computing Grids, Conf. Proc. of the Workshop on Large-Scale and Volatile Desktop Grids (PCGrid 2007), 30th March 2007, Long Beach, California U.S.A, pp1-8, ISBN: 1-4244-0910-1.
[8] G. Fedak, et al.: XtremWeb: A Generic Global Computing System. CCGRID2001 Workshop on Global Computing on Personal Devices, May 2001, IEEE Press.
[9] EGI – European Grid Infrastructure, http://www.egi.eu/
[10] The NorduGrid Website - http://www.nordugrid.org/
[11] The Open Science Grid Website - http://www.opensciencegrid.org/
[12] Z. Balaton et al.: EDGeS, the Common Boundary between Service and Desktop Grids, Parallel Processing Letters, Vol. 18, No. 3, September 2008, ISSN: 0129-6264, pp. 433-445
[13] The EGEE – Enabling Grids for E-sciencE – Website, http://www.eu-egee.org/
[14] E. Urbah et al.: EDGeS: bridging EGEE to BOINC and XtremWeb, Journal of Grid Computing, Vol 7, Issue 3, pp 335-354, 2009.
[15] Z. Farkas, et al.: Utilizing the EGEE Infrastructure for Desktop Grids, In: Distributed and Parallel Systems In Focus: Desktop Grid Computing, Peter Kacsuk, Robert Lovas and Zsolt Nemeth (Editors), Springer, 2008, pp 27-35.
[16] Ian Kelley and Ian Taylor. Bridging the Data Management Gap between Service and Desktop Grids. In: Distributed and Parallel Systems In Focus: Desktop Grid Computing, Peter Kacsuk, Robert Lovas and Zsolt Nemeth (Editors), Springer, 2008.
[17] Attic Website – http://www.atticfs.org
[18] AbdelHamid Elwaer, Andrew Harrison, Ian Kelley and Ian Taylor. Attic: A Case Study for Distributing Data in BOINC Projects. 5th Workshop on Desktop Grids and Volunteer Computing Systems (PCGrid 2011). May 20th, 2011, Anchorage, Alaska, USA.
[19] Costa, F., Kelley, I., Silva, L., Taylor, I., 2008. Peer-To-Peer Techniques for Data Distribution in Desktop Grid Computing Platforms. In: To be published in a special volume of the CoreGRID Springer series.
[20] Cohen, B., June 2003. Incentives Build Robustness in BitTorrent. In: Work- shop on Economics of Peer-to-Peer Systems (P2PEcon'03). Berkeley, CA.
[21] Pasquale Cozza, Carlo Mastroianni, D. T., Taylor, I., August 28-29 2006. A Super-Peer Protocol for Multiple Job Submission on a Grid. In: Pro- ceedings of the CoreGRID Workshop on Grid Middleware Workshop in conjunction with Euro-Par, Dresden. To be Published. Springer.
[22] ViSAGE: Video Stream Anaysis in a Grid Environment, http://edges-grid.eu/c/document_library/get_file?folderId=63854&name=DLFE-1504.pdf
[23] U.Becciani, C.Gheller, M. Comparato, A. Costa: VisIVO, a VO enabled tool for Scientific Visualization and Data Analysis, http://wiki.eurovotech.org/bin/view/VOTech/VisIVO, 2005.
[24] A. Marosi et al.: Enabling Java applications for BOINC with DC-API, In Distributed and Parallel Systems, Springer US, August 2008, pp 3-12.
[25] A. Marosi, Z. Balaton, P. Kacsuk: GenWrapper: A generic wrapper for running legacy applications on desktop grids, IPDPS 2009, IEEE.
[26] Jacq N. et al.: Grid-enabled virtual Screening against malaria, Journal of Grid Computing, 6 (1), 29-432008, Springer Netherlands, 2007, DOI 10.1007/s10723-007-9085-5.
[27] EDGI Project Website, http://edgi-project.eu
[28] W3C MD5 recommandations, http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0

PoS(ISGC 2011 & OGF 31)060