Scalable Desktop Grid System[1]

Peter Kacsuk[1], Norbert Podhorszki[1], Tamás Kiss[2]

[1] MTA SZTAKI
Computer and Automation Research Institute of the
Hungarian Academy of Sciences
H-1518 Budapest, P.O. Box 63., Hungary
{kacsuk, pnorbert}@sztaki.hu

[2] University of Westminster, Cavendish School of Computer Science
115 New Cavendish Street, London W1W 6UW, UK
T.Kiss@westminster.ac.uk

Abstract. Desktop grids are easy to install on large number of personal computers, which is a prerequisite for the spread of grid technology. Current desktop grids connect all PCs into a flat hierarchy, that is, all computers to a central server. SZTAKI Desktop Grid starts from a standalone desktop grid, as a building block. It is extended to include clusters as single powerful PCs, while using their local resource management system. Such building blocks support overtaking additional tasks from other desktop grids, enabling the set-up of a hierarchy. Desktop grids with different owners thus can share resources, although only in a hierarchical structure. This brings desktop grids closer to other grid technologies where sharing resources by several users is the most important feature.

Keywords. Desktop Grid, infrastructure, BOINC, cluster computing

1 Introduction

Originally, the aim of the researchers in the field of Grid was that anyone could offer resources for a Grid system, and anyone can claim resources dynamically, according to the actual needs, in order to solve a computationally intensive task. This twofold aim has been, however, not fully achieved. Currently, we can observe two different trends in the development of Grid systems, according to these aims.

Researchers and developers in the first trend are creating a Grid service, which can be accessed by lots of users. A resource can become part of the Grid by installing a predefined software set (middleware). The middleware is, however, so complex that it needs a lot of effort to maintain. Therefore it is natural, that single persons do not offer their resources but all resources are maintained by institutions, where

[1] This research work was carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265) and by the Hungarian grants IHM 4671/1/2003, OTKA T042459.

professional system administrators take care of the hardware/middleware/software environment and ensure the high-availability of the Grid. Examples of such Grid infrastructures are the world's largest Grid: EGEE (Enabling Grids for E-SciencE), the NGS (National Grid Service) in the UK, the OSG (Open Science Grid) and TeraGrid in the USA and NAREGI in Japan. The original aim of enabling anyone to join the Grid with one's resources has not been fulfilled. Nevertheless, anyone who is registered at the Certificate Authority of such a Grid and has a valid certificate can access the Grid and use the resources.

A complementary trend can also be observed for the other part of the original aim. Here, anyone can bring resources into the Grid system, offering them for the common goal of that Grid. Nonetheless, only some people can use those resources for computation. The most well-know example, or better to say, the original distributed computing facility example of such Grids is the SETI@home [1]. In Grids, similar to the concepts of SETI@home, personal computers owned by individuals are connected to some servers to form a large computing infrastructure. Such systems are called with the terms: Internet-based distributed computing, public Internet computing or desktop grid; we use the term desktop grid (DG) from now on. A PC owner should just install one program package, register herself on the web page of the Grid system and configure the program by simply giving the address of the central server. Afterwards, the local software runs in background (e.g. as a screensaver) and the owner does not need to take care of the Grid activity of her computer. In a desktop grid, applications can be performed in the well-known master-worker paradigm. The application is split up into many small subtasks (e.g. splitting input data into smaller, independent data units) that can be processed independently. Subtasks are processed by the individual PCs, running the same executable but processing different input data. The central server of the Grid runs the master program, which creates the subtasks and processes the incoming sub-results.

The main advantage of a desktop grid is its simplicity thus, allowing anyone to join. The main disadvantage is that currently only problems computable by the master-worker paradigm can be implemented on such a system. Desktop grids have already been used at world-wide scales to solve very large computational tasks in cancer research [2], in search for the sign of extraterrestrial intelligence [1], climate predictions [3] and so on.

Desktop grids can be used efficiently and conveniently in smaller scales as well. We believe that small scale desktop grids can be the building blocks of a larger Grid. This is a new concept that can bring closer the two directions of Grid developments. It is easy to deploy desktop grids in small scale organisations and to connect individual PCs into it therefore we get a grid system that can spread much faster then heavy-weight grid implementations. On the other hand, if such desktop grids can share the resources and their owners can use others' desktop grid resources, the many user conception of the other trend is also realised. Steps towards the collaboration of desktop grids are the support of clusters – so they are easy to include as a resource –, the hierarchy of desktop grids within a large organisation with several levels of hierarchy, and the resource sharing among independent desktop grids in different organisations.

SZTAKI Desktop Grid (SZDG) starts with a standalone desktop grid, as a building block. It is extended to include clusters as single powerful PCs, while using their local

resource management system. Such building blocks support overtaking additional tasks from other desktop grids, enabling the set-up of a hierarchy of DGs. The final dream of creating a large-scale Grid from DGs as building blocks will be investigated in a forthcoming paper. In the current paper, the SZTAKI Desktop Grid is described, from the basic single desktop grid to the support of clusters and to the hierarchy of desktop grids.

1.1 Related work

Condor. Condor's approach [4] is radically different from the DG concept. Condor represents a push model by which jobs can be submitted into a local Grid (cluster) or global Grid (friendly condor pools or Globus Grid [5]). The DG concept applies the pull model whereby free resources can call for task units. The advantage of the DG concept is that it is highly scalable (even millions of desktops can be handled by a DG server) and extremely easy to install at the desktop level. The scalability of Condor is not proven yet. Largest experiments are at the level of 10000 jobs in EGEE but it requires an extremely complicated Grid middleware infrastructure that is difficult to install and maintain at the desktop level.

BOINC (Berkeley Open Infrastructure for Network Computing, see [6,7]) is developed by the SETI@home group in order to create an open infrastructure that could be the base for all large-scale scientific projects that are attractive for public interest and that can use millions of personal computers for processing their data. This concept enables millions of PC owners to install single software (the BOINC client) and then, each of them can decide what project they support with the empty cycles of their computers. There is no need to delete, reinstall and maintain software packages to change among the projects. Actually as of January 2005, the overall computational power of the more than 80.000 participants of BOINC project is about 106 TeraFLOPS, providing the most powerful supercomputer of the world, which, in contrast to the original SETI@home distributed computing facility, can run several different distributed applications.

The properties of BOINC can be used for smaller scale, combining the power of the computers at institutional level, or even at department level. The SZTAKI Desktop Grid is based on BOINC since this is a well-established open source project that already proved its feasibility and scalability. The basic infrastructure of SZTAKI Desktop Grid is provided by a BOINC server installation and the connected PCs at a given organisational level.

XtremWeb is a research project [8], which, similarly to BOINC, aims to serve as a substrate for Global Computing experiments. Basically, it supports the centralised set-up of servers and PCs as workers. In addition, it can also be used to build a peer-to-peer system with centralised control, where any worker node can become a client that submits jobs. It does not allow storing data, it allows only job submission.

Commercial Desktop Grids

There are several companies providing a Desktop Grid solution for enterprises [9,10,11,12]. The most well-known examples are the Entropia Inc, and the United Devices. Those systems support the desktops, clusters and database servers available at an enterprise. However, their cluster connection solution is not known for the research community and it is very likely that their model is based on the push model. Our goal is to develop a pull model solution since it is consistent with the current BOINC concept.

2 SZTAKI Desktop Grid

The basic idea of SZTAKI Desktop Grid is first, to provide a basic DG infrastructure that is easy to install, to maintain and to use at an organisational level. This basic infrastructure enables us to connect PCs within a department and to run (small) distributed projects on it. Second, clusters are supported as they are increasingly available at many departments of institutions and companies as well. Third, the hierarchical structure of an organisation needs the possibility of connecting such departmental desktop grids into an infrastructure where larger projects can use more resources than available within one department. Fourth, more generally, to make possible the resource sharing among desktop grids that are not related in a hierarchical way. In this way, small-scale desktop grids, which are easy to install, can be the building blocks of a large grid infrastructure.

SZTAKI Desktop Grid is based on the BOINC infrastructure, as we believe that it provides everything that is needed for a basic desktop grid with one (running on a single machine or on multiple machines) server and many workers. The infrastructure for executing computational tasks and for storing data sets is used only. Its support for user credits, teams and the web-based discussion forums are not relevant for an organisation but, of course, all these features are available if needed.

A desktop grid within an organisation (institution, or just a department) enables us

- to connect PCs in the organisation into the desktop grid,
- to install several distributed computing projects on the desktop grid, and
- to use the connected PCs to compute subtasks of those projects.

As Fig. 1 shows, there is a Scheduler Server and a Data Server in the BOINC infrastructure, however, they can be simply installed on one computer but also they can exist in multiple instances as well, depending on the central processing needs of a project. Scheduler Server stores all information about available platforms, application programs, subtasks, connected machines (and users) and results for subtasks. Data Server stores all executables, input and output files. On each PC, a core client is running that downloads application client executables, subtasks (describing actual work) and input files to perform the subtasks. The main application on the top level has to generate the sequential subtasks and to process subresults. BOINC gives tools and support for generic distributed projects to do that, however, SZTAKI provides a much simpler and easier-to-use API, the DC-API. The use of this API enables scientist just concentrate on task generation and processing results without knowing

even what grid infrastructure is serving the processing needs. Of course, the use of the API is not obligatory, one can use BOINC's tools as well.

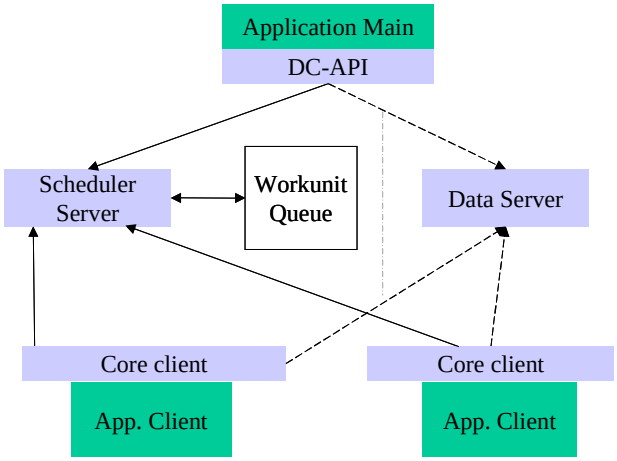

Fig. 1. BOINC-based Desktop Grid infrastructure

2.1 Supporting clusters within SZTAKI Desktop Grid

BOINC in itself does not provide any support for clusters. It has a server that generates work and there are clients that do the work (actually several ones on an SMP node, one subtask per CPU). The need for cluster support is clear. No one would like to develop a sophisticated distributed application that uses partly the desktop grid and partly a cluster, all with different concepts, APIs and syntaxes. Cluster's job management concept is more general than the execution of work units (subtasks) within a desktop grid therefore, the latter one can be mapped onto the previous one. There are five possibilities in extending the BOINC infrastructure for cluster support.

1. A desktop grid client is installed on all machines of the cluster and connected to the server of the desktop grid of the given organisation, i.e. all machines of the cluster participate individually, as a normal PC in the desktop grid.
2. A complete desktop grid is installed on the cluster, with the server on the front-end node, and all machines connected to it. This way, the cluster can participate in a larger desktop grid as one leaf element in a hierarchy, see section 2.2.
3. An independent, higher-level broker distributes work among clusters and desktop grids.
4. The server of the desktop grid should be aware of the presence of a cluster and submit jobs instead of work units,

5. An extended version of a single desktop grid client is installed onto the cluster's front-end, which converts desktop grid work units into traditional jobs and submits them to the cluster's job management.

The *first possibility* is easy to achieve, only the desktop grid client should be installed on the machines, see Fig. 2. The configuration of BOINC core client consist of defining a registered user's ID and the project server URL. Settings for the user's preferences are defined on the project web server, and settings are propagated to all clients with the same user ID. BOINC provides easy install on multiple machines based on one installation therefore, the whole procedure is very easy. Compare this with the installation and configuration of the LHC Grid middleware (of course, the latter providing more functionality).

However, if the cluster is not a brand new one or the owners do not want to use it exclusively for the desktop grid, a job manager is surely installed and used on that cluster. This means, that the job manager and the desktop grid clients are competing for the spare cycles of the computers. The job manager's role is to coordinate the resources within a cluster and to balance the load on it. Desktop grid clients and subtasks coming from the desktop grid server are out of the view for the job manager therefore, it is not able to function properly.

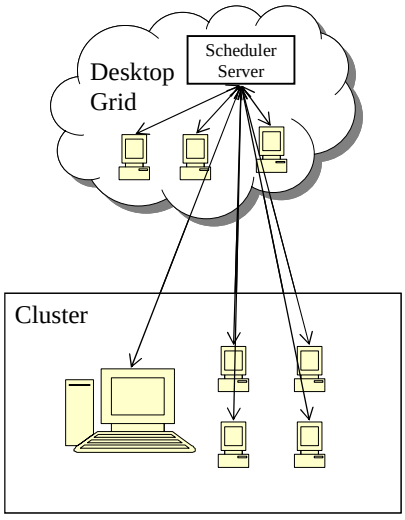

Fig. 2. Clusters 1. All machines are clients

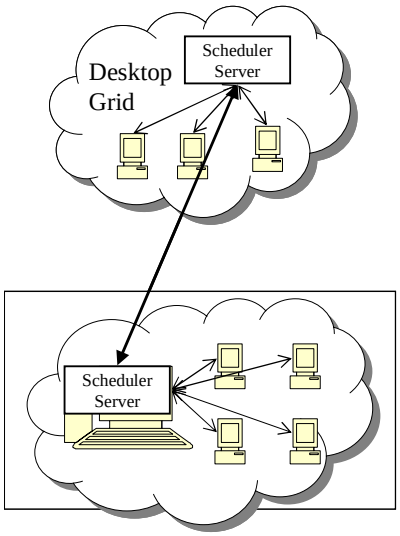

Fig. 3. Clusters 2: stand-alone desktop grid

The *second possibility* (see Fig. 3) by-passes the job manager as well, having the same drawback and therefore, it is not recommended. However, if the hierarchy of desktop grids are a reality, this option can be considered as a free solution for connecting a cluster into an existing desktop grid.

Usually, we may think at first that if different things are to be connected and to work together, there is a need for a higher-level actor that distributes work among

those things and takes care of the good balance, as in the *third possibility*. That is, in our case, an appropriate broker is needed that is able to gather information about the status of the different entities (desktop grids and clusters), to decide where to send the next piece of work and to convert subtasks into work units or jobs according to the target system, see Fig. 4. Such an approach is followed in the Lattice project [13], which is developing a community-based Grid system that integrates Grid middleware technologies and widely used life science applications. This system deals with traditional jobs, i.e. executables, input data and definition of requirements and preferences. Jobs are submitted to a modified version of the Condor-G broker [14] that sends a job either to a Globus-based grid or to a BOINC-based desktop grid.

In this case, a desktop grid is just one element among others. Different grid implementations can be connected together this way if appropriate conversion between the different concepts, representations and syntaxes can be managed.

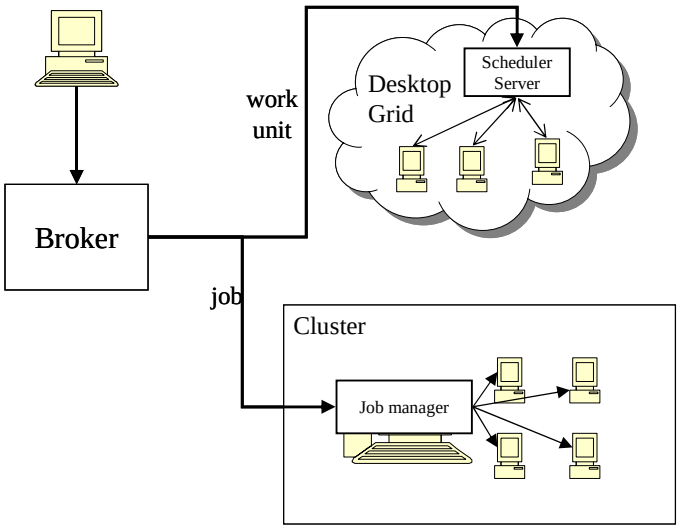

Fig. 4. Cluster 3: High-level brokering of jobs

The *fourth possibility* keeps the leading role of the desktop grid server, see Fig. 5. In this scenario, there is a desktop grid as "the grid", in which clusters are connected from "below". The server should be configured in a way that it knows about the cluster, its static status information (size, benchmark information) and its dynamic status information (number of available machines) – the same way, as the broker of the third option should do. As in the basic desktop grid, work is distributed by the server; however, it can decide to send some work to the cluster. In this case, the work unit representation should be converted to the job representation, which can be submitted to the job manager of the cluster.

This solution needs lot of development of the server's implementation. A monitoring system should be used to get status information about the cluster, such information should be stored and handled somehow, decision logic should be altered

– all these tasks are also part of the third option. Besides that, the internal work unit should be converted into a traditional job and the server should be able to contact the job manager of the cluster remotely and submit jobs. As we mentioned, work unit representations can be mapped onto job representations therefore, this is quite a simple task.

The *fifth possibility* is the most elegant way of including clusters into the desktop grid, see Fig. 6. In a desktop grid, client machines are connecting to the server and ask for work; this is called pull-mode. In contrast, job managers and grids of the first trend mentioned in the introduction submit work (jobs) to selected resources (push-mode). In this option, clusters can participate in the pull-mode execution of the desktop grid. A desktop grid client originally asks for a given amount of work to be processed on the given machine. However, with some modification, it can ask for many work units, transform them into jobs and submit them into a cluster. The desktop grid server can see it as a normal, but somewhat very powerful client. In this solution, only the client should be modified, and since it is running on the front-end node of the cluster, information gathering and job submission are easy to perform.

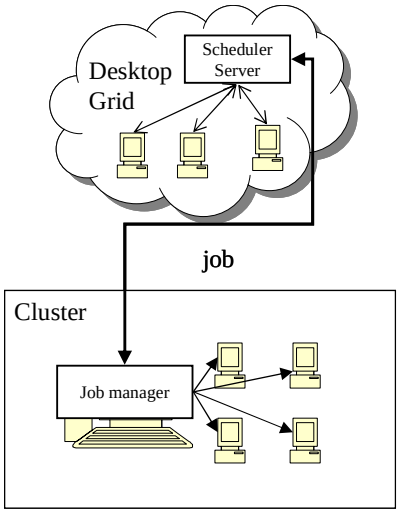

Fig. 5. Clusters 4: Submit jobs from server

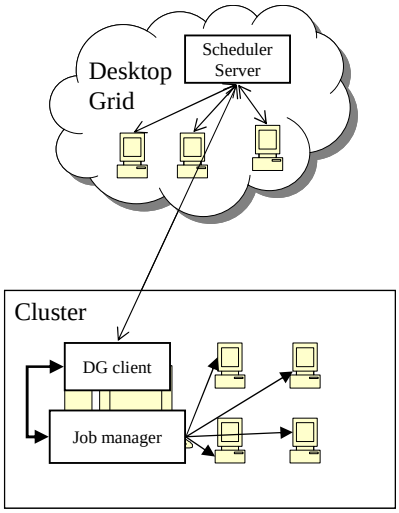

Fig. 6. Clusters 5: Special DG client on the front-end

We have chosen the fifth possibility for SZTAKI Desktop Grid, because this way clusters are seamlessly integrated into it. This concept keeps the role of the job manager of the cluster and it requires less number of modifications than the others. The current implementation can connect clusters managed by the Condor job manager. The cluster extended version of SZDG is currently used for education purposes at the University of Miskolc and is under installation at the University of Westminster for speeding-up the execution of large DSP applications.

2.2 Hierarchical Desktop Grid

Departments can be satisfied by using the basic SZTAKI Desktop Grid with cluster support. All PCs and clusters of a department can be connected into one local (department level) DG system and distributed projects can use all these resources. It is natural to ask, what if there are several departments using their own resources independently but there is an important project at a higher organisational level (e.g. at a school or campus level of a university or at university level). Having the previous set-up in the departments, only one of the departments can be selected to run the project. Of course, the ideal would be to use all departments' resources for that project. Besides again developing something new component (e.g. a broker) to control over the different desktop grids, there is the possibility to build a hierarchy of desktop grids – if the building blocks can enable it as shown in Fig. 7. In such a hierarchy, desktop grids on the lower level can ask for work from higher level (pull mode), or vice versa, desktop grids on the higher level can send work to the lower levels (push mode).

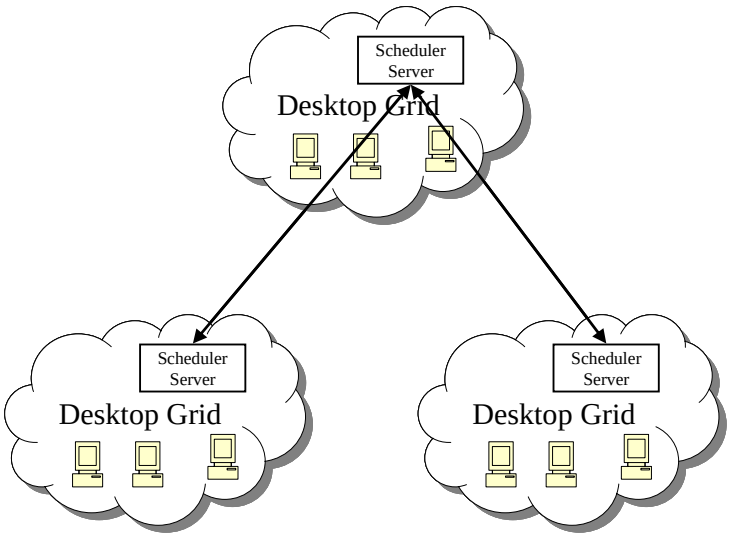

Fig. 7. Hierarchy of desktop grids

SZTAKI Desktop Grid supports the pull mode, as this is the original way how desktop grids work. The control of important work on the higher level can be realised with priority handling on the lower level. A basic SZTAKI Desktop Grid can be configured to participate in a hierarchy, that is, to connect to a higher-level instance of SZTAKI Desktop Grid (parent node in the tree of the hierarchy). When the child node (a stand-alone desktop grid) has less work than resources available, it asks for work from the parent. The parent node can see the child as one powerful client, exactly as in the case of a cluster, which asks for work units.

Of course, the BOINC-based server has to be extended to ask for work from somewhere else (i.e., behave similarly as a client) when there is not enough work locally. Fortunately, this can be done separately in the case of BOINC. Work units are generated by the running applications and they are put into a database of the BOINC server. Whether a work unit arrives from outside or from a local application, it does not matter. Therefore, it is enough to create a new daemon on the server machine that observes the status of the desktop grid. When client machines' requests for work are rejected – or when the daemon predicts that this will happen soon – the daemon can turn to the parent desktop grid and ask for work units. The daemon behaves towards the parent as a BOINC client, asking for work and reporting results. However, it puts all those work units into the database of the local server thus, client machines will process them and give the results. The daemon should also wait and look for the incoming results and send them back to the parent.

However, there is the issue of applications when we want to connect two BOINC-based desktop grids. In the BOINC infrastructure, application executables should be registered in the server and signed with a private key (of the project). Clients always check if the downloaded executable is registered and valid thus, avoiding the possibility of spreading arbitrary code by hackers. A parent desktop grid is an alien to the child in this sense; executables registered in the parent desktop grid should be registered before work units using that executables can be processed.

In BOINC, for security reasons, the private key of a project should be stored on a machine that is separated from the network. Application client executables should be signed by the administrator of the projects and only the signature should be copied from that separated machine. The signature is checked by using the public key at the client level. If a client machine receives work units from projects belonging to different levels of the DG hierarchy, the client should know the public keys of all the servers placed above it in the DG hierarchy. When a work unit arrives it should contain the source level's identifier based on which the desktop will know which public key to use for checking the signature of the code.

The prototype version of the hierarchical SZDG has been developed and tested. However, the prototype can support only one level of the hierarchy and hence current work aims at solving the problem of multi-level hierarchy.

2.3 The SZTAKI Desktop Grid Service

In order to demonstrate the strength and usability of the DG concept for Hungarian institutes SZTAKI has created the global version of SZTAKI Desktop Grid [15] that is a new BOINC-like DG service. SZDG has been running since July 2005 and extracted more than 7 500 participants and more than 18 000 machines from all over the world. The sustained performance of SZDG is about 800 GFlops. The task to be solved by SZDG is a math problem of generating 11- and 12-dimension binary number systems. These can contribute to develop new encryption algorithms for safer security systems.

Though SZDG works the same way as the other global DG systems its basic role is to provide an experimental system for Hungarian institutes and companies to learn the technology and its possible usage as local DG system. We have found that institutes

are very cautious with the usage of Grid technology and hence in order to convince them about the usefulness and safety of the DG systems they can test the DG technology in three phases:

Phase 1: Test the client side. Staff members of institutes can connect their PCs to the demo project thus, participating in one large-scale computing project; similarly, as people all over the world participate in BOINC, XtremWeb and Grid.org based projects. In this way they can be convinced that the client components of SZDG are safe enough and do not cause any harm to their desktop machines.

Phase 2: Test the server side with their own application. If an institute has a problem that needs large computing power to solve, SZTAKI helps to create a new project on SZDG and provides the central server for that project. The institute can provide the PCs and clusters for SZDG to work on that project. In this way the desktops of the institute will work on the institute's project separated from other projects running on the SZDG.

Phase 3: Finally, if the institute is convinced on the usefulness of the SZDG concept SZTAKI can help them to set-up and maintain their own local DG system based on the SZDG concept.

3 Applications of SZTAKI Desktop Grid

The success of the SZDG concept is proven by several applications. One of the basic issues of modern drug discovery is the exclusion of chemically unstable, biologically inactive and toxic compounds from the research process in the early stages, thereby reducing the cost and the time period of the drug development. The main purpose of the ADMEToxGrid project [16] is to develop an enterprise Grid system that is suitable for predicting these chemical parameters of millions of compounds in a short time and in a secure manner, while also exploiting the free capacity of the office computers located at the different site of the company. In this project the local version of SZTAKI Desktop Grid serves as the base of the enterprise Grid framework deployed in Comgenex company.

A Hungarian data mining Grid project [17] aims at the development of the prototype of data mining software using SZTAKI Desktop Grid technology. The software enables the user to select the algorithm and to make scheduling decisions, as well as the generation of higher quality data mining models by automating these permits. The innovative element of the project is the optimization in the scheduling of data mining algorithms enabled by meta-level learning. The prototype supports the documentation and verification of data mining projects, while it remains expandable thanks to its architecture. Special attention is paid to data privacy issues. After the termination of the project, the prototype and its subsequent versions will be available for non-profit research purposes. Following up on the results of the project, the members will consider the commercial deployment of a data mining grid-based product.

One of the goals of the Hungarian climate modeling Grid project [18] is to elaborate a new generation of Desktop Grid systems based on the achievements of SZTAKI Desktop Grid and to provide a Grid execution environment for numerical weather prediction and climate models developed by the Hungarian Meteorological Service. Participants are about to create a so-called Global Desktop Grid (GDG) environment in Hungary, which is the first attempt to apply the Desktop Grid technology not only for academic/research purposes, but using the intranet infrastructure of companies. In the project, a GDG system will be built involving large amount of computational resources from three sites: SZTAKI, Hungarian Meteorological Service and econet.hu. Later, this GDG will be the prototype for a national GDG service which aims to integrate home PC owners, whose interest will be challenged in financial manner. In this way, a service provider based Hungarian Grid market will be born, which leads to a new kind of internet service in the long-term.

The WestFocus GridAlliance between Brunel University and the University of Westminster is dedicated to raising the profile of Grid computing in the West London region and to facilitate real Grid-solutions in the industry. One of their application deals with designing periodic non-uniform sampling sequences for digital alias free signal processing [19]. This is a computationally intensive problem, in which sequential (single computer based) solutions could easily run for weeks or even months. In order to reduce computation time, the sequential algorithm was parallelized, making it possible to execute parts of the calculations on different nodes of computational Grids at the same time. This in turn reduces the overall runtime of the application. The SZTAKI Desktop Grid based version of the DSP application has been demonstrated with 100 PCs located at the two universities in London. The typically one-month computation time was reduced to two days by the local SZDG.

4 Conclusion

In this paper, SZTAKI Desktop Grid's structure is presented, discussing the possibilities of the support of clusters within a desktop grid. SZTAKI Desktop Grid uses the BOINC infrastructure as a basic building block for connecting PCs to solve large scale distributed programs. It is extended by the support of clusters by installing a modified version of the PC client that converts incoming subtasks into traditional jobs and submits them to the cluster's job manager. Such a desktop grid, as a building block, is then used to build a hierarchy of DGs in an institute or company to provide individual desktop grids to the lower level organisational units but also to provide a larger infrastructure to solve problems on the higher level. The ability to propagate work from one desktop grid to the other (but only in a hierarchy) is a step towards a grid infrastructure that is easy to install and has several users that share resources. This means that in the future DG based grid systems these two features will not exclude each others as they currently do in today's grid systems.

5 References

1. D.P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D. Werthimer: SETI@home: An Experiment in Public-Resource Computing, Communications of the ACM, Vol. 45 No. 11, November 2002, pp. 56-61
2. United Devices Cancer Research Project: http://www.grid.org/projects/cancer
3. D. A. Stainforth et al.: Uncertainty in the predictions of the climate response to rising levels of greenhouse gases, Nature, 27 January 2005, vol 433.
4. D. Thain, T. Tannenbaum and M. Livny: Condor and the Grid.
   Grid Computing – Making the Global Infrastructure a Reality. Ed. F. Berman, A. Hey and G. Fox. John-Wiley & Sons, Ltd. Chapter 11. 2003
5. Foster, C. Kesselman: Globus: A Metacomputing Infrastructure Toolkit.
   Intl J. Supercomputer Applications, 11(2):115-128, 1997.
6. 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. Available at: http://boinc.berkeley.edu/grid_paper_04.pdf
7. BOINC Home Page: http://boinc.berkeley.edu
8. G. Fedak, C. Germain, V. Néri and F. Cappello: XtremWeb: A Generic Global Computing System. CCGRID2001 Workshop on Global Computing on Personal Devices, May 2001, IEEE Press.
9. Grid MP, United Devices Inc. http://www.ud.com
10. Platform LSF, Platform Computing. http://www.platform.com
11. A. Chien: Architecture of a commercial enterprise desktop Grid: the Entropia system. *Grid Computing – Making the Global Infrastructure a Reality.* Ed. F. Berman, A. Hey and G. Fox. John-Wiley & Sons, Ltd. Chapter 12. 2003
12. DeskGrid, Info Design Inc. http://www.deskgrid.com
13. Myers, D. S., and M. P. Cummings. Necessity is the mother of invention: a simple grid computing system using commodity tools. Journal of Parallel and Distributed Computing, Volume 63, Issue 5, May 2003, pp. 578-589.
14. James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke: Condor-G: A Computation Management Agent for Multi-Institutional Grids, Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10) San Francisco, California, August 7-9, 2001.
15. SZTAKI Desktop Grid: http://szdg.lpds.sztaki.hu/szdg/
16. ADMEToxGrid project: www.admetoxgrid.hu
17. Data mining Grid project:
    http://www.sztaki.hu/search/projects/project_information/?uid=00025
18. Climate modeling Grid project:
    http://www.sztaki.hu/search/projects/project_information/?uid=00188
19. A. Tarczynski, T.Kiss, D. Qu, G. Terstyanszky, T. Delaittre, S. Winter, Application of Grid Computing for Designing a Class of Optimal Periodic Nonuniform Sampling Sequences, Conf. Proc. of the Grid-Enabling Legacy Applications and Supporting End Users Workshop, within the framework of the 15th IEEE International Symposium on High Performance Distributed Computing , HPDC'15, Paris, France, June 19-23, 2006.