SZTAKI Desktop Grid – a Hierarchical Desktop Grid System

P. Kacsuk, A. Marosi, J. Kovács, Z. Balaton, G. Gombás, G. Vida, Á. Kornafeld

MTA SZTAKI
Computer and Automation Research Institute of the
Hungarian Academy of Sciences
H-1518 Budapest, P.O. Box 63., Hungary

Abstract

Desktop grids are an emerging trend in grid computing. Interoperability and resource sharing between different desktop grid projects however is not yet solved. This paper presents an approach of organizing desktop grids in a hierarchy that makes it possible for lower-level desktop grids to acquire work from the higher level if they have spare resources available, thus building large-scale grid systems from small local desktop grid installations. An industrial application that can make use of such a setup is also described.

1 Introduction

Desktop grids are an emerging trend is the field of grid systems. The most well-know example, or better to say, the original distributed computing facility example of such grids is the SETI@home [2]. In grids similar to the concepts of SETI@home, personal computers owned by individuals are connected to some central servers to form a large computing infrastructure. Such systems are called either Internet-based distributed computing, public Internet computing or desktop grid; we use the term desktop grid (DG) from now on.

Contrary to the "many users – few resource provider" paradigm of previous grid systems, desktop grids employ a "many resource providers – few users" model, meaning that anyone can join and offer resources, but only a selected few users can make use of those resources.

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 [9], in search for the sign of extraterrestrial intelligence [2], climate prediction [8] and so on.

Desktop grids can be used efficiently and conveniently on a smaller scale 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 desktop grids closer the direction of "traditional" grid developments. The easy deployment of desktop grids in small organizations can lead to a grid system that spreads much faster than 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 concept of the other grid trend is also realised. A major step towards the collaboration of desktop grids is the support of hierarchy of desktop grids within a large organization or community.

SZTAKI Local Desktop Grid (LDG) is designed and implemented to realize this idea. LDG is based on BOINC [1][3] but significantly extends the client concept of BOINC in order to enable the creation of hierarchical desktop grids.

Imagine a university where the needs of departments can be satisfied by using the basic LDG. All PCs 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). 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 of developing some new component (e.g. a broker) to control over the different desktop grids, there is the possibility to build a hierarchy of desktop grids.

Of course, when different resources are all provided by different individuals, you'd just ask them to join the DG projects separately as the projects come and go, and therefore the above mentioned hierarchy would not be needed. But when the resources belong under a single administrative domain, attacking the problem at the server side instead of constantly reconfiguring the clients makes the system more robust and makes tasks like policy enforcement or inter-department accounting much easier.

In 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). SZTAKI Desktop Grid supports the pull mode, as this is the original way how desktop grids work. The control of work at the higher level can be realised with priority handling at the lower level. A basic LDG can be configured to participate in a hierarchy, that is, to connect to a higher-level instance of LDG (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.

To support hierarchy, 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 achieved by just introducing a new component without having to modify existing components.

Workunits are generated by the running applications and they are stored in a database. Whether a workunit 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, and injects approrpiate workunits when there is a shortage detected. 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 workunits. The daemon behaves towards the parent as a BOINC client, asking for work and reporting results. However, it puts all those workunits into the database used by 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.

The rest of the paper is organized as follows. The principles and architecture of the hierarchical desktop grid is described in in Section 2. Section 3 shows an enterprise application of the LDG. Section 4 describes related research and finally, section 5 concludes the paper.

2 Architecture of the hierarchical LDG

In order to help the spreading of the desktop grid concept, SZTAKI has created the SZTAKI Local Desktop Grid. The LDG is based on BOINC software [1] and is tailored for easy deployment at small organizations.

One of the major enhancements of the LDG is hierarchy that allows the use of desktop grid projects as building blocks for larger grids. For example, divisions of a company or departments of a university can form a company- or faculty-wide desktop grid. Every DG project has a classical parent-child relationship with the others. They may request work from a project above (*consumer*) or may provide work for a project below (*producer*). The project server can enter a hierarchical mode, when one of it's consumers require more work than it has for disposal. It will then contact one of it's producer nodes and request more work.

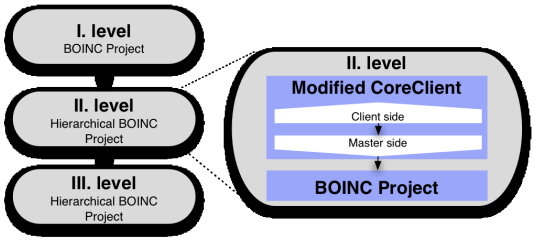

Figure 1: The split architecture of the Hierarchy prototype. Inside the *Core Client* the *Client side* is acting as a consumer by requesting work and the *Master side* as a producer by providing work for the project.

It is allowed for a project to have multiple consumers and/or producers. We use the simple layout in Figure 1 for presenting the enhancements of the architecture. A modified Core Client (what is normally used on the clients to participate in a project) is running on the machine hosting the consumer project.

BOINC terminology uses the platform expression for the specific combinations of architectures and operating systems. We modified the Core Client in a way that we may specify what platform it should pretend to be using. This allows us to query all the predefined platforms for applications. However, the deployment of the application at the lower levels is not handled by the Core Client, it is the task of the project administrator. The way to overcome this deficiency and to enable automatic propagation of application code will be addressed in future research.

The modified Core Client has a split architecture. When the number of unsent workunits runs below a specified threshold the *client side* of the Core Client will contact a producer for work. A project may have more producers configured each with a priority assigned. First, the producer with the highest priority will be contacted for work, if it fails to provide work then the next one is queried and so on. When a workunit is successfully acquired from one of the producer projects, the *master side* of the Core Client injects it into the local project's database in the form of a local workunit. When this new workunit has been completed, it is also the modified Core Client's responsibility to delegate the result back to the originating producer.

Hierarchical desktop grids present some new problems that do not exist in stand-alone destop grids. These problems are not fatal but the DG operators must be aware of them to avoid sub-optimal behaviour.

The first problem arises due to redundant computing. Redundancy ensures every workunit will have a correct result by simply sending the same piece of work to multiple clients and comparing the results to filter out corrupt ones. Figure 2 shows a three level layout with the redundancy of three at each level. In this case each producer at each level creates three copies of any workunit received. By the third level there will be nine redundant ones. This means that nine clients will compute the same workunit instead of the supposed three (which was the requested redundancy at the first level). If more levels are added to the hierarchy this number will exponentially grow. It can be easily solved by forcing redundancy to be disabled at all but the first level. This way exactly the requested number of redundant workunits will be distributed.

The other important problem arises from the way BOINC handles workunit deadlines. A deadline is applied in DG systems in order to prohibit workunit-hijacking by clients. When a workunit is donwloaded, a timer is started. If the timer expires before a result is received, the workunit is considered invalid and is resent to another client. Since every level of the hierarchy creates a new local workunit, the deadline of the original workunit at the top level is not propagated. If a consumer level in the hierarchy does not have enough resources to process the workunits it has acquired from its producers in a timely manner, the producer level may invalidate the workunits. If however the consumer level

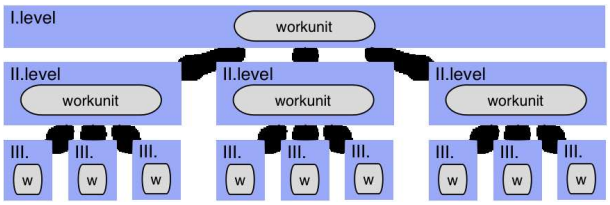

Figure 2: Growing number of redundant workunits in the hierarchy demonstrated with a simple three level layout.

does not request enough workunits from the producers, some of the clients will be left without work. Predicting the performance is not the subject of this paper, but for demonstration purposes we created a monitoring and statistics tool, which monitors the performance, number of users, hosts, sent and unsent workunits and many more. This tool allows us to manually tune the number of workunits the consumer requests from the producer. Automatic performance tuning will require further research.

To test our system we have deployed a seven-level hierarchical environment with clients attached to the lowest level. Six of the servers were running on Debian Linux 3.1/Intel, one was using Mac OS/X 10.4/PowerPC. Six clients were attached, three using Debian Linux 3.1/Intel, two Windows XP and one Mac OS/X 10.4/PowerPC. We have also created a simple application for all platforms, with the only purpose to produce high load and run exactly for the time given in the workunit. Our goal with this diverse environment was not performance analysis, but simply to test the environment for possible problems and bottlenecks. Using the prototype we were able to provide basic hierarchical functionality without modifying existing projects. Only the modified Core Client was needed for work request and distribution.

We have created Debian packages for LDG. This makes the deployment of LDG servers fast and easy. After installation, a new project can be created with a single command, and then only the application executables should be set into operation on that machine. Client machines need just the standard BOINC Core Client installed. The set-up and operation of the SZTAKI Local Desktop Grid infrastructure is so simple that this is a strong reason in itself to use such an infrastructure instead of a complex grid middleware, provided it fits our applications' needs.

3 Drug discovery using ADMEToxGrid

The ADMEToxGrid project [7] takes the SZTAKI Local Desktop Grid as the middleware for building an enterprise-level grid infrastructure. This project demonstrates that LDG can be used even as a professional Grid environment inside a company.

The main goal of the ADMEToxGrid system is to create the distributed version of selected packages of the Pallas software, thereby serving for large scale ADMETox parameter predictions.

ADMEToxGrid consists of at least two servers and several Personal Computers (see Figure 3). One of the servers is responsible for the management of the Grid system the other is the central database server handling prediction results.

The central server supervises the connected computers, provides user interface to the Grid resources and executes ADMEToxGrid specific services such as chemical data collection and handling, and the executable version control. The database server provides chemical data for parameter calculation and handles prediction results as well as data related to the entire prediction environment and grid system.

A web portal is applied to provide controlled and secure access and user control to the ADMEToxGrid system. The user selects the input data of the predicting calculation with a predefined or a custom query. The Input Engine processes the data request, collects results from the connected database server and creates the calculation job called Project. The Project consists of molecule structures in chemical data formats, such as Smiles and SD formats and all the parameters, which are defined by the user and necessary for the calculations. The Project containing all the input data is processed by the workunit producer component and stored in the Data server. The Workunit producer component splits the Project into workunits with defined size, and format and passes them to the LDG via DC-API [6]. The chemical data format is transformed into the internal standard XML format and each XML file is temporarily stored.

The results and metadata of the prediction are stored temporarily on the Data server of the LDG and processed continuously by the Assimilator via the DC-API. The processed results are transferred to the Persistor or saved into result files in a defined file format depending on the Desktop Grid user preference. The Persistor is responsible for the insertion of the results into the Database server.

Each client machine has different resources, different CPU, memory size, and hard disk capacity, therefore it cannot be ensured that all kind of calculation programs can be executed on each machine. The LDG keeps record of all Client Machine resources detected by the BOINC Core Client program on the Client Machine during the logging in procedure. Similarly, the description of the work packages contains information needed for the relevant calculation. This mechanism ensures that workunits transferred to the client machine with appropriate computational resources are executed.

Basically, the ADMEToxGRID system relies on open source software products and tools, except the database server, which is an MS SQL server in our case. The Central server (Debian Linux), coordinating the GRID system, consists of SZTAKI Desktop Grid server compiled on the same machine, a MySQL database server, and an Apache web server. For the web portal, the Apache web server is installed with PHP, Perl and Python language supports.

The Core Client software, controlling the executable package on the client

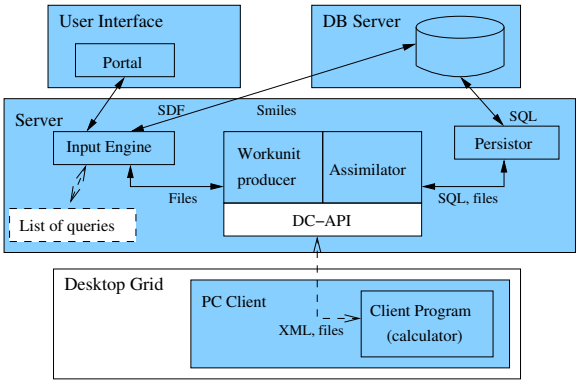

Figure 3: Architecture of the ADMEToxGRID system.

PC, is available for different Windows and various flavours of Linux operating systems. In summary, the overall system has been designed to be a flexible and easily maintainable corporate Grid system.

4 Related Research

BOINC (Berkeley Open Infrastructure for Network Computing, see) is developed by the SETI@home [2] 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. 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.

XtremWeb [5] is a research project, 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.

There are several companies providing a Desktop Grid solution for enterprises. 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. Basically, they are Windows-based solutions, however, understanding the needs of the commercial users, clusters and mainframes can be connected into the desktop grid as well, mostly based on the Globus

toolkit. Entropia [4] can completely seclude the execution of the desktop grid applications from other processes running on the PC (sandboxing) thus, ensuring that grid applications cannot access data on the client machines. Strong cryptography ensures also privacy of application data: encoding protects from stealing data on the network, digital signatures provides safe identification and protects from intentional falsification of data.

Besides the drug discovery application used in the ADMEToxGRID project, the local SZTAKI Desktop Grid technology forms the base infrastructure of other research, development and educational activities as follows.

A Hungarian project in data mining aims the development of the prototype software using LDG technology, which 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. 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 but the project members will also consider the commercial deployment of a data mining grid-based product.

One of the goals of the project in meteorology and climate modelling is to elaborate a hierarchical local Desktop Grid systems based on the achievements of LDG and to provide a grid execution environment for numerical weather prediction and climate models developed by Hungarian Meteorological Service.

The WestFocus GridAlliance [10] 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 Data Signal Processing (DSP) application deals with designing periodic non-uniform sampling sequences for digital alias free signal processing. This is a computationally intensive problem, in which sequential solutions could easily run for days or even weeks. In order to reduce computation time, the sequential algorithm needed to be parallelized. The LDG based version of the DSP application has been demonstrated in 2006/Q1 with 100 PCs located at the two Universities in London.

5 Conclusion

This paper demonstrated how can stand-alone desktop grid installations combined to form a large-scale grid system. The technique demostrated does not require extensive modifications to the existing system therefore it is easy to deploy even on existing systems.

It was also demonstrated that grid (and especially desktop grid) systems have grown up in the sense that they are now ready to be used in the corporate world, outside of the academic community. We have shown a drug discovery application that runs on top of a desktop grid system and is actively used by the industry.

Future directions and open issues were also discussed, the most important

ones being the automatic deployment of application code on consumer nodes in a desktop grid hierarchy and the tuning of the modified Core Client that provides the link between the members of the hierarchy.

This research work was carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

References

1. D. P. Anderson, "BOINC: A System for Public-Resource Computing and Storage", *Proc. 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, November 8, 2004.
2. 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
3. BOINC Home Page: `http://boinc.berkeley.edu`
4. A. 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.
5. G. Fedak, C. Germain, V. Néri and F. Cappello, "XtremWeb: A Generic Global Computing System", *Proc. CCGRID2001 Workshop on Global Computing on Personal Devices*, IEEE Press, May 2001.
6. P. Kacsuk, N. Podhorszki, T. Kiss, "Scalable Desktop Grid System", *High performance computing for computational science*, VECPAR'06, Rio de Janeiro, Brazil, 2006, July 10-13. pp. 1-13.
7. R. Lovas, G. Gombás, N. Podhorszki, G. Pöcze, P. Hliva, Z. Gulyás, A. Petz, *ADMEToxGRID: First Experiences with an Enterprise Grid for Drug Discovery*, MIPRO 2006/Hypermedia and Grid Systems, Croatian Society for Information and Communication Technology, Electronics and Microelectronics, 2006. Opatija, Croatia, pp. 194-200.
8. 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.*
9. United Devices Cancer Research Project: `http://www.grid.org/projects/cancer`
10. The WestFocus GridAlliance: `http://www.gridalliance.co.uk`