Peer-to-Peer Desktop Grids in the Real World: the ShareGrid Project *

Cosimo Anglano[1], Massimo Canonico[1], Marco Guazzone[1], Marco Botta[2], Sergio Rabellino[2], Simone Arena[3], Guglielmo Girardi[3]
[1]Dipartimento di Informatica, Università del Piemonte Orientale, Alessandria (Italy),
[2] Dipartimento di Informatica, Università di Torino, Italy,
[3] Torino Piemonte Internet Exchange (TOP-IX), Torino, Italy
email:{cosimo.anglano,massimo.canonico,marco.guazzone}@unipmn.it,
{botta,sergio.rabellino}@di.unito.it,
{simone.arena,guglielmo.girardi}@topix.it

Abstract

ShareGrid is a peer-to-peer desktop grid aimed at satisfying the computing needs of the small research laboratories located in the Piedmont area in Northern Italy. ShareGrid adopts a cooperative approach, in which each participant allows the other ones to use his/her own resources on a reciprocity basis. ShareGrid is based on the OurGrid *middleware, that provides a set of mechanisms enabling participating entities to quickly, fairly, and securely share their resources. In this paper we report our experience in designing, deploying, and using ShareGrid, and we describe the applications using it, as well as the lessons we learned, the problems that still remain open, and some possible solutions to them.*

1 Introduction

In many scientific areas, the use of computers to carry out research has become essential. The availability of computing infrastructures able to speed-up as much as possible the execution of relevant applications is therefore fundamental for the achievement of scientific outcomes. *Volunteer computing* infrastructures, set up by means of suitable middlewares (e.g. *BOINC* [1] and *XtremWeb* [9]) and able to harvest the unused cycles provided by a set of desktop-class computers owned by independent individuals, have been shown to be able to provide performance comparable to those of more traditional grids at a fraction of their cost. However, computing infrastructures based on this paradigm can grow large enough to provide significant benefits only if a large set of resource owners are convinced to install the software that will allow them to contribute their resources to the system. Furthermore, a non-negligible effort may be required to set up a control center responsible for managing the resources contributed to the system. It is no surprise, therefore, that the largest volunteer-computing communities are aggregated by projects carried out in prestigious and famous institutions, since they typically have more resources to invest in project advertisement and infrastructure management. Thus, the gap between the research that can be carried out at the few large research institutions (that are able to profitably exploit volountary computing solutions) and the one that can be conducted by the majority of small labs (that cannot) is becoming larger.

Smaller research labs, however, can overcome these difficulties if they federate their resources and use them cooperatively according to the *peer-to-peer computing* paradigm, in which each participant lets other members use its resources when it does not need them, provided that they do the same. This is the precise goal of the *ShareGrid Project* [16], that is aimed at establishing a shared computing infrastructure in the Piedmont area, in Northern Italy, by federating the computing resources of the many small research laboratories of the region that do not have enough resources to satisfy their computing needs, and sufficient visibility to coagulate a volunteer community large enough to provide an adequate amount of computing power, and enough manpower to manage it. The ShareGrid infrastructure is based on the peer-to-peer computing paradigm, and relies on the *OurGrid* [4] middleware to provide a set of core services enabling participating laboratories to easily share their resources when they do not need them, to use those of other participants easily and transparently, and to make sure that *free riders* (i.e., participants that do not donate their resources but want to use those of others) do not receive services at the detrimental of contributing users. At

*This work has been supported by TOP-IX and the Piedmont Region Agency under the Innovation Development Program.

978-0-7695-3156-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CCGRID.2008.23

the moment, ShareGrid comprises more than 200 machines (including both desktop-class and server-class computers), donated by three University research labs and one private institution, that are used to run a variety of different applications. Plans are underway to enlarge both the number of contributing laboratories and of the application base.

To the best of our knowledge, there are not many other general-purpose computing infrastructures that, as ShareGrid, are based on the peer-to-peer paradigm. The only other example we are aware of is the publicly-accessible desktop grid operated by the *OurGrid* project [14], that however – unlike ShareGrid – does not provide any support to its users, and comprises a smaller number of machines. The other similar computing infrastructures based on the volunteer-computing paradigm that have been deployed in the past few years (e.g., *Grid.Org* [12], *Compute Against Cancer* [7], *Folding@home* [11], *ClimatePrediction.Net* [6], *FightAids@Home* [10], *LCH@Home* [13], and *Distributed Folding* [8], just to name a few) support the execution of a *single application*. In contrast, ShareGrid is able to support the simultaneous execution of many competing applications.

In this paper we report our experience and the lessons we learned while setting up, deploying, and using the ShareGrid infrastructure. We start with Section 2, where we describe the architecture of ShareGrid, the configuration of the middleware on which it is based, and the development of some additional middleware components that were made necessary by the peculiarities of the user community targeted by ShareGrid. We then continue with Section 3, where we describe the set of applications that are currently executed on ShareGrid, and with Section 4, where we report the lessons we learned, the problems we encountered, and the possible solutions to them. Finally, Section 5 concludes the paper and outlines future research work.

2 The architecture of ShareGrid

As anticipated in the Introduction, ShareGrid aims at providing a shared computing infrastructure obtained by federating resources contributed by independent research institutions, typically doing research in domains different from computer science, and having limited manpower to manage their computing resources. Therefore, in order to be successful, it was crucial to design ShareGrid in such a way that it provides suitable mechanisms facilitating individual sites to participate and use the resulting infrastructure. In order to satisfy the above requirement, ShareGrid adopts a peer-to-peer paradigm in which each laboratory can independently contribute its resources anytime, without the need of signing any prior agreement with the rest of the community, and can withdraw them exactly in the same way. To implement this vision, we adopted the *OurGrid* middleware, that has been specifically conceived to provide suitable mechanisms and policies to enable small laboratories to aggregate their computing resources in a peer-to-peer fashion, so that the creation, management, and operation of the resulting computing infrastructure is greatly simplified. More specifically, *OurGrid* provides mechanisms enabling a set of independent organizations to quickly assemble a shared computing infrastructure and to profitably use it to run *Bag-of-Tasks* [5] applications (i.e., parallel applications comprising a set of independent tasks) in a secure way, and to deal with the typically large volatility and heterogeneity of the constituent resources, without requiring the availability of a large amount of manpower to manage it. Furthermore, by giving priority to users according to the amount of contributed resources, *OurGrid* provides clear incentives to donate resources to the infrastructure, thus perfectly matching the goal of ShareGrid.

The architecture of ShareGrid, schematically depicted in Fig. 1, follows a peer-to-peer paradigm in which a set of *peer nodes*, each in charge of managing a set of computing nodes, interact with each other in order to fulfill execution requests coming from a population of users. In its present

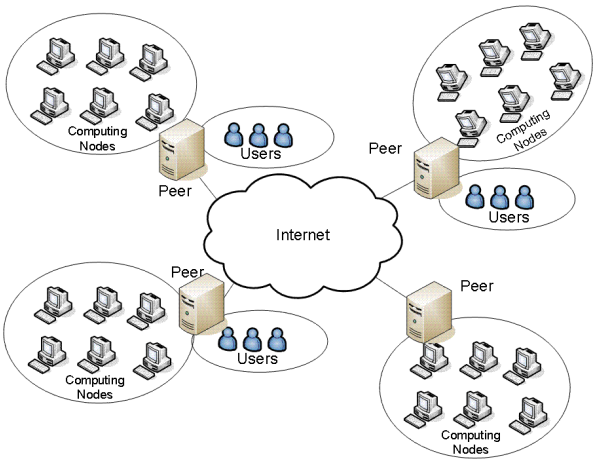

Figure 1. The Architecture of ShareGrid

incarnation, ShareGrid comprises four different sites: three of them are independent laboratories located in the University of Turin and the University of Piemonte Orientale at Alessandria, while the fourth one is a small research facility provided by the a non-profit organization (*TOP-IX* [17]) to accommodate the computational needs of selected independent researchers that do not own resources to donate. Globally, these sites provide 224 machines, equipped with Intel and AMD processors of different families, and running various versions of Linux, Windows, and Solaris. These machines are not dedicated to run grid applications, but – according to the volunteer computing paradigm – can be used by ShareGrid only when they are not used by the re-

spective owners. Therefore, the actual number of available machines may strongly fluctuate over time.

The various middleware components used in ShareGrid (the peers and the agents running on computational nodes), as well as the clients used by users to submit and manage their applications, are provided by *OurGrid*. Each site that wants to join ShareGrid deploys a peer on one of its resources, and configures the other ones to act as computing nodes under the control of the corresponding peer. This peer is provided with a list of the other peers in ShareGrid, and uses it to contact them to set the peering relationships.

The peer of a given site fulfills two purposes: (a) it accepts the submission requests coming from users of the site, and decides whether to dispatch them on local resources or to remote ones, and (b) accepts submission requests coming from external users, queues them locally, and dispatches them on local resources when they become available. Peers prioritize submissions according to the standard *OurGrid* policy, in which local users are given priority with respect to remote ones: when a local application is submitted for execution and there are no local resources available for execution, remote applications that are being run are terminated, and the corresponding resources are allocated to the local applications. When choosing which one among its local resource will run a given task, the peer will pick one at random among those that are idle and that possibly match the requirement expressed by the user with the characteristics of the machine. If there are not enough local resources, the peer contacts other peers of ShareGrid and select the one(s) to which it will ship the local application according to the *network of favors* principle [4]. In practice, each peer P_i maintains for each other peer P_j in the system the amount of *favors* it received, a variable that accounts for the amount of time that P_j devoted to execute P_i's tasks (suitable weighted to take into account the computing power of the involved machines). When selecting remote applications for the execution on local resources, P_i gives preference to the applications coming from the peer with the highest favor value. By coupling the network of favors mechanisms with a proper accounting mechanism [15], *OurGrid* ensures that *free riders* (i.e. users that do not donate resources to the system but use those donated by other ones) receive little or no service.

In addition to the basic *OurGrid* components, the deployment of ShareGrid required us to develop additional mechanisms to tailor the specific needs of the targeted user community.

3 ShareGrid Applications

ShareGrid, being based on the *OurGrid* middleware, supports only the execution of Bag-of-Tasks applications.These applications typically consist in a set of independent tasks that do not communicate among them. Despite their simplicity, Bag-of-Tasks are used in a variety of domains, such as parameter sweeps, simulations, fractal calculations, computational biology, and computer imaging.

At the moment of this writing, ShareGrid is used by five applications, belonging to very different domains, that have been implemented as Bag-of-Tasks, namely:

- *Distributed rendering*: Scene rendering is a typical compute-intensive activity, where a set of scenes of a given movie must be rendered via software in order to add in static and dynamic features (like bitmap or procedural textures, lights, bump mapping, etc.), and to stitch them together for making the final animation. The inherent nature of scene rendering, where different frames belonging to the same animation can be rendered independently from each other, makes it particularly suited to distributed processing. A distributed version of scene rendering can indeed be obtained by processing each frame independently, and then merging them together to build the complete movie. Distributed rendering drastically reduces the rendering time (and the production time too) and the investment costs needed for buying and managing personal computing resources. For ShareGrid, we developed a Bag-of-Tasks version of *blender* [3],in which bags correspond to scenes, and tasks of a bag to the different frames of the same scene. This application is being used by professionals working in the animation field to accomplish their activities.

- *Simulation of economic systems*: Researchers in the Department of Economic and Financial Sciences of the University of Turin have developed *Parei*, an agent-based simulation system modelling the over 500,000 production units composing the economy of Piedmont, and use it for the understanding and the esteem of economic dynamics, particularly when they are difficult to be observed in vivo, and for the evaluation, in vitro, of economic effects of proposed public policies. Given the large size of the model, and the large set of parameter values that must be studied, the analysis of the *Parei* model requires very large amounts of computing power to obtain results in an acceptable time, that far exceed those available to the researchers that developed it. However, such a simulation naturally fits the Bag-of-Tasks paradigm, since each set of model parameters, as well as each distinct scenario, corresponds to an independent task. A Bag-of-Tasks implementation of this application, executed on ShareGrid, has been able to produce very interesting results in an amount of time much smaller than that taken by its sequential version. Furthermore, as an additional benefit, the simultaneous exploration of the solution space al-

lowed by running several tasks on the same parameter set and scenario made possible to neutralize deterministic effects in random number generation and to analyze the solution space.

- *Simulation of molecular systems*: The simulation of the dynamic behavior of molecular systems is a computationally intensive task, especially when biological macromolecules are concerned, and solvent molecules are modeled explicitly. The availability of large computational resources is thus mandatory to accomplish reliable esteems of the free energy of binding between a ligand and its target. Techniques like thermodynamic integration require running several, independent, molecular dynamics simulations, and can naturally be implemented as Bag-of-Tasks applications, since each simulation corresponds to an individual task that do not communicate with the other ones.An additional application belonging to this scientific domain consists in performing the simultaneous virtual screening of a series of potential drug candidates by means of molecular docking and subsequent molecular dynamics simulation refinement, thus making in silico methods competitive with traditional in vitro wet-lab screening. Researchers of Department of Drug Science of the University of Turin have developed Bag-of-Tasks versions of the above implementation, and have extensively used them on ShareGrid. As result, they have obtained innovative and significant results in a relatively short time, that would instead have required much longer time on traditional computing platforms.

- *Simulation of scheduling algorithms for distributed systems*: Discrete-event simulation is often used in Computer Science to study the behavior of certain systems before actually implementing them. Job scheduling for distributed systems is one of the research areas in which discrete-event simulation is often the tool-of-choice. The study of the behavior of a given scheduling algorithm for different scenarios, or the comparison of different algorithms for the same set of scenarios, can be naturally performed by simultaneously executing many independent simulations in parallel. A Bag-of-Tasks discrete simulation engine has been developed by researchers of the Department of Computer Science at the University of Piemonte Orientale, and is used to perform parameter sweep studies of scheduling algorithms for desktop grids. The computing infrastructure provided by ShareGrid has enabled these researchers to enlarge the set of scenarios and scheduling algorithm that could be studied in a reasonable amount of time, thus significantly increasing their possibility of investigating interesting avenues of research that could not have been possible otherwise.

- *Evaluation of Classifier Systems*: Classification is the task of recognizing an object or event as an instance of a given class and represents one of the problems most frequently found in computer applications. Medical and fault diagnosis, prognosis, image recognition, text categorization, adaptive user profiling are weel known instances of classification tasks. The task of automatically inferring a classification program (*classifier*) from a set of previously classified data has been investigated for more than two decades in pattern recognition, statistics, and in machine learning and produced a large number of powerful algorithms. When new application domains and/or new requirements are sought, existing algorithms as well as new ones need to be tested and evaluated extensively to assess their performances. This evaluation activity consists in running the algorithm to acquire a classification program and test the learned model on, so called, test data, and is very time-consuming, because, several runs are performed by varying configuration parameters of the algorithm and the datasets on which the algorithms are tested. Each run of an algorithm can then be seen as a separate task and, thus, a Bag-of-Tasks paradigm can be applied to organize a large number of experiments and get advantage of a grid computing environment. In particular, the researchers at the Department of Computer Science at the University of Turin used the computing infrastructure provided by ShareGrid to test the performances of an SVM classifier on a user identification task. The experimental setup consists in running the SVM classifier on 64 different datasets, by varying three parameters of the algorithm for a total of about 1200 runs. Each run takes about 1 hour cpu time on average to run. This means that it would take more than a month to terminate all the experiments using a single cpu system. By using the ShareGrid computing power, the full experiment took only a couple of days.

4 Lessons learned

In order to make ShareGrid suitable to the execution of such a variety of applications, we had to face a set of issues, that had not been anticipated, and whose solution – in some cases – is under development, that made us learn some important lessons, that are summarized below.

Lesson 1: Software dependencies do exist. The first problem we encountered is concerned with the software dependencies characterizing applications. In many (if not all) cases, each application requires the availability of specific libraries that are not part of the standard software packages installed on the machines. This problem gets even worse

for Windows-based machines. While a straightforward solution is – at least in theory – available, namely the static compilation of the application executable, there are cases when such a compilation produces an excessively large executable file, or when statically-compiled libraries are not available for all the platforms present in ShareGrid. Therefore, we had to adopt a quick-and-dirty solution to this problem by installing, where possible, dynamic versions of the libraries required by each application. Unfortunately, this has not been possible for all machines and libraries, so the set of resources available to a given application is actually smaller than the number of machines in ShareGrid. A better solution, that however requires a large development effort, consists in encapsulating all the execution environment required by each application into a virtual machine, and ship it to the computing nodes of ShareGrid. This, however, puts some constraints on the computational nodes that, in order to profitably run virtual machines, must be equipped with hardware support for virtualization. Fortunately, this does not seem to be a real problem anymore, since practically all the newer processors incorporate this feature, but for older machines the problem does exist.

Lesson 2: Input and output data may be harder to deal with than anticipated. The second major problem we encountered was caused by applications requiring large amounts of input data. The version of *OurGrid* used for ShareGrid provides no support to remote I/O, so all the data required and produced by an application must be staged in and out for each task. When the size of the input data is very large, copying them on the target machine may require a very large amount of time, that greatly delays the completion of the corresponding task. The performance penalty caused by the need of staging in the input data can be dramatic, especially considering that such a copy must be practically performed for each task of a bag. In some cases, when a task needs the same input of another one, *OurGrid* is able to schedule them on the same resource, but this only partially solves the problem. We have not yet devised a solution for this problem, although we are considering some ways of integrating I/O support in *OurGrid*. One of these possibilities consists in coupling a BitTorrent-like file transfer system, like for instance done in [18], or an infrastructure like the *File Mover* [2].

Lesson 3: Data privacy is fundamental. The third problem, raised by those applications that process confidential or sensitive data, consists in guaranteeing suitable privacy levels for their input or output data. Unfortunately, this seems to be a problem that cannot be easily solved on a purely technical basis. One possible partial solution that we are considering consists in using an encrypted file system to store the data, that are decrypted on the fly – by means of a secret key – by the application that needs them. This solution, however, is only partial, as the data exist in an encrypted form at least into the memory space of the task using them and can be therefore accessed by dumping the above memory, although this is less simple than reading them from the disk (as it would be possible if they were stored in an unencrypted form).

Lesson 4: The system does not manage itself. Although *OurGrid* is a mature middleware, it is also a research product constantly under development that, consequently, has some bugs. While its developing community is quite prompt to react to bug reports, bug fixing requires some time during which it is important to minimize the impact on the user community of ShareGrid. One of the known bugs of ShareGrid is particularly annoying for users, namely the unanticipated disappearance of a peer (with all the computing nodes it manages). A very quick-and-dirty solution to this problem, that can be applied until the bug is fixed, consists in restarting the peer. Doing that manually is an unpracticable option, since it would require the continuous human assistance, so we implemented a simple software monitor that checks the availability of each peer of ShareGrid, and automatically restart those that are not reported as present by the *OurGrid* middleware.

Lesson 5: Sometimes you cannot configure your firewall. Nowadays practically all the institutions protect their networks by means of a firewall that, if not properly configured, does not allow the corresponding peer to communicate with the other ones in ShareGrid. Unfortunately, in our case one of the participating laboratories could not directly configure the corresponding firewall, because of some network access restriction policies. However, one of the other laboratories belonging to the same university did not have the same restrictions, and could directly configure its firewall. This allowed us to solve the problem by setting up a Virtual Private Network connecting the first laboratory to the second one, and vehiculating through it all the traffic coming from and directed to the corresponding peer.

Lesson 6: The world is not Linux-centric. As many research software products, *OurGrid* has a strong Linux orientation, although the bulk of the middleware – being written in Java – runs on any platform supporting this language. However, there are a few components that have been developed having the features and the restrictions of Linux in mind, and do not work well (or not at all) on Windows resources. We felt that this was a major problem that – if not properly addressed – would significantly limit the size that ShareGrid could potentially reach, given the dominant position of Windows in non-academic environments.

The first issue due to the Linux-centric approach adopted by *OurGrid* is that its client runs only on *nix machines, so Windows users cannot submit applications from their machines. In order to overcome this problem, we developed a web portal providing access to ShareGrid by means of a standard web browser. In this way, on the one hand we provide a better support to Windows users, and on the other hand we enlarge the potential user basis for ShareGrid.

Another consequence of the Linux-centric vision mentioned above is that the *idleness detector*, used by the agent of *OurGrid* to decide if a given machine is not being used by the respective owner so that it can be used to run Grid jobs, does not properly work with USB keyboards and mices, although nowadays practically all desktop machines are equipped with USB human input devices (HIDs). This is an heritage coming from the fact that currently the Linux kernel is notoriously unable to detect activity produced by these devices. However, Windows does not suffer from this problem, but the idleness detector is the same for both platforms, so it does not support USB keyboards and mices on Windows too. While this is not a real problem for computational clusters, it can become a significant issue for desktop users, as their machine would be deemed idle even if they are interacting with it via the keyboard and the mouse. We solved this problem by rewriting the idleness detector for Windows, that now is able to properly deal with USB HIDs.

5 Conclusions and Future Work

In this paper we described ShareGrid, a peer-to-peer desktop grid aiming at aggregating and sharing, in a very simple and transparent way, the computing resources contributed by independent research laboratories. ShareGrid targets the research community of the Piedmont region in Northern Italy, and at the moment aggregates more than 200 resources provided by four institutions. The usability of ShareGrid for applications coming from different scientific domains has been demonstrated by the development of various applications, ranging from scene rendering to simulation of molecular, economic, and computer systems. The rapid turnaround time that ShareGrid has been able to guarantee to these applications has enabled the research teams using them to obtain their results in much shorter time than they were used to by using their own resources alone, thus enabling them to explore in the same amount of time scenarios requiring a much larger computing power.

We have been pleasantly surprised by the degree of satisfaction and enthusiasm manifested by the users of ShareGrid, that in some cases have contributed to the project not only by porting their applications on this platform, but also by developing some tools simplifying the usage of ShareGrid and providing them to the rest of the community.

The development of ShareGrid is still work in progress, and many new functionalities are planned to be developed and integrated. Among them, the most prominent ones are the support for the execution of virtual machines encapsulating the applications, instead of the mere executable code as done at the moment, and the integration of an efficient data transfer mechanisms enabling ShareGrid to provide effective support to data-intensive applications as well.

References

[1] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *GRID '04: Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004.

[2] C. Anglano and M. Canonico. The File Mover: High-Performance Data Transfer for the Grid. *Concurrency and Computation: Practice and Experience*, 20(1), January 2008.

[3] Home Page of Blender. http://www.blender.org. Visited on Nov. 22nd, 2007.

[4] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, , and M. Mowbray. Labs of the world, unite!!! *Journal of Grid Computing*, 2006.

[5] W. Cirne and et al. Grid computing for bag of tasks applications. In *Proc. of* 3^{rd} *IFIP Conf. on E-Commerce, E-Business and E-Government*, 2003.

[6] The ClimatePrediction.Net Project. http://www.climateprediction.net. Visited on Sept. 7th, 2007.

[7] The Compute Against Cancer Project. http://www.computeagainstcancer.org. Visited on Sept. 7th, 2007.

[8] The Distributed Folding Project. http://www.distributedfolding.org. Visited on Sept. 7th, 2007.

[9] G. Fedak, C. Germain, V. Neri, and F. Cappello. Xtremweb: A generic global computing system. In *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, page 582, 2001.

[10] The FightAids@Home Project. http://fightaidsathome.scripps.edu. Visited on Sept. 7th, 2007.

[11] The Folding@home Project". http://www.stanford.edu/group/pandegroup/folding. Visited on Sept. 7th, 2007.

[12] The GRID.ORG^{TM} Project. http://www.grid.org. Visited on Sept. 7th, 2007.

[13] The LHC@Home Project. http://lhcathome.cern.ch. Visited on Sept. 7th, 2007.

[14] OurGrid Home Page. http://www.ourgrid.org.

[15] R. Santos, A. Andrade, W. Cirne, F. Brasileiro, and N. Andrade. Relative Autonomous Accounting for Peer-to-Peer Grids. *Concurrency and Computation: Practice and Experience*, September 2007.

[16] The ShareGrid Project Home Page. http://dcs.di.unipmn.it. Visited on Nov. 22nd, 2007.

[17] Torino Piemonte Internet Exchange Home Page. http://www.topix.it.

[18] B. Wei, G. Fedak, and F. Cappello. Towards efficient data distribution on computational desktop grids with BitTorrent. *Future Generation Computer Systems*, 23(8), November 2007.