Bridging the High Performance Computing Gap: the OurGrid Experience

Francisco Brasileiro, Eliane Araújo, William Voorsluys, Milena Oliveira, Flávio Figueiredo
{fubica, eliane, william, milena, flaviov}@lsd.ufcg.edu.br

Universidade Federal de Campina Grande
Departamento de Sistemas e Computação
Laboratório de Sistemas Distribuídos
Av. Aprígio Veloso, s/n, Bloco CO
58.109-970, Campina Grande - PB, Brazil

Abstract

High performance computing is currently not affordable for those users that cannot rely on having a highly qualified computing support team. To cater for these users' needs we have proposed, implemented and deployed OurGrid. OurGrid is a peer-to-peer grid middleware that supports the automatic creation of large computational grids for the execution of embarrassingly parallel applications. It has been used to support the OurGrid Community - a public free-to-join grid that is in production since December 2004. In this paper we show how the OurGrid Community has been used to support the execution of a number of applications. Further we discuss the main benefits brought up by the system and the difficulties that have been faced by the system developers and the users and managers of the OurGrid Community.

Keywords: *grid computing; peer-to-peer; free-to-join; public grid; bag-of-tasks applications.*

1 Introduction

New developments in communication and computing technologies have substantially impacted the way scientific research is conducted. Not only these technologies allow for an unprecedent level of interaction among researchers, but also the use of computers for data analysis, simulations, visualization of results etc play nowadays a fundamental role in the working methodology adopted by many research groups. As a consequence, having access to high performance computing facilities has become a must for many research areas.

Due to this demand, research in computing science has, for some time now, sought ways to expand the reach of high performance computing. One of the first initiatives in this sense provided a way to aggregate unused computing power in a local area network [17]. The next step was to increase the scale on the number of resources aggregated by harvesting the idle computing power in the Internet [4, 2], what has been dubbed *public resource computing* or *voluntary computing*. More recently, *grid computing* has been proposed as a way to build virtual organizations aggregating computing resources that are under different administrative domains [14].

Despite being successful in providing non-trivial amounts of computing power, these mainstream technologies are not affordable to most users with demands for high performance computing. In the case of voluntary computing, it is necessary to set up a large control center that will be responsible for managing the millions of public resources that are contributed to the system. Moreover, a lot of effort needs to be placed in convincing resource owners to install the software that will allow them to contribute their resources to the system. On the other hand, most grids in production (eg. EGEE - www.eu-egee.org, TeraGrid - www.teragrid.org, NAREGI - www.naregi.org, NGS - www.grid-support.ac.uk, APAC - www.apac.edu.au, PRAGMA - www.pragma-grid.net) use some flavor of the Globus toolkit [1]. Installing, configuring, and customizing the Globus middleware is not a trivial task, and today requires a highly skilled support team. Moreover, joining such grids involves a negotiation process that consumes time and may place players with smaller amounts of resources in a disadvantageous position in relation to larger players.

Therefore, if for one side the massive use of computers by researchers fosters, at an ever-faster pace, amazing developments for the society at large, for the other side it con-

Seventh IEEE International Symposium on Cluster Computing and the Grid(CCGrid'07)
0-7695-2833-3/07 $20.00 © 2007 IEEE

tributes to increase even more the gap between the research that can be conducted at the few large research labs that can afford the cost of implementing the above mentioned technologies and that conducted by the majority of small labs that cannot. Yet, the fact that a lab is not large has no relation with the importance of the research it develops. As a matter of fact, in countries with very few large labs, a substantial portion of all research developed is conducted by research groups organized around small and medium-sized labs. In particular, this is the case in Brazil and, probably, in the whole Latin America. For these countries it is of fundamental importance to envision alternative ways for making high performance computing affordable to any user that requires it.

To address this problem we have developed a grid middleware that allows small and medium-sized labs (and naturally large-sized labs as well) to donate their idle computational resources in exchange for accessing other labs' idle resources when needed [11]. The middleware, named OurGrid, can be used to create, in a simple way, a very large peer-to-peer computational grid for the execution of bag-of-tasks (BoT) applications, ie. those parallel applications that can be decoupled in many independent tasks that do not need to communicate with each other. Despite being simple, BoT applications are used in a variety of settings, including data mining, massive searches (such as key breaking), parameter sweeps, simulations, fractal calculations, computational biology, and computer imaging, among others.

OurGrid leverages from the fact that most computers are not fully used on a continuous basis. Even when actively using computers as research tools, researchers alternate between application execution (when they demand computational power) and result analysis (when their computational resources go mostly idle). In fact, in most small and medium-sized labs, researchers run relatively small applications, for which they seek fast turn-around times, so as to speed up the run/analyze iterative cycle. Interestingly, such behavior has also been observed among users of Enterprise Desktop Grids [16].

OurGrid was designed to be fast, simple, scalable, and secure. It uses replication schedulers [20, 22] to achieve very good application turn-around time even in the absence of information about the application and the resources. Assembling a grid based on OurGrid is simple and scalable because it requires no negotiation. OurGrid allows peers (for instance, research labs) to join the grid at their will, making their idle computational resources available to other peers at the same time that they gain access to the idle cycles of the resources that have been added to the grid by other peers. OurGrid uses a reciprocation mechanism to prevent free-riding [6, 5, 7] and a sandboxing mechanism to provide security [10] against malicious applications.

OurGrid has been used to implement the OurGrid Community, which is in production since December 2004. Any lab can join the community by simply downloading the code from http://www.ourgrid.org/ and following the straightforward installation procedure. Naturally, OurGrid is open source. The number of labs and institutions forming the community varies over time, as it will be discussed later, but the OurGrid Community is arguably one of the most important grid initiatives in Brazil (see http://status.ourgrid.org/ for a fresh snapshot of the running system).

In this paper we share our experience in developing the OurGrid middleware and deploying the OurGrid Community. The rest of the paper is organized as follows. Section 2 briefly presents OurGrid's architecture and gives some details on its functioning. We proceed showing how the OurGrid Community has been used so far. There are four possible ways to write parallel applications to run on OurGrid; Section 3 describes real applications that have been developed following each of these four approaches. Then, in Section 4, we discuss the main benefits that users have had in using OurGrid to run their applications, as well as the main difficulties that both managers and users of the OurGrid Community have faced in the past two years during which the system has been in production. Finally, Section 5 closes the paper with our concluding remarks and a brief discussion on directions for further work in the context of the OurGrid project.

2 The OurGrid Middleware

A grid supported by the OurGrid middleware has three main components, namely the SWAN security mechanism, the OurGrid Resource Manager Peer (or simply the OurGrid Peer) and the MyGrid Broker. Figure 1 depicts the OurGrid architecture.

SWAN is an agent that executes on the grid resources (referred as a *grid machines*) and is responsible for implementing a secure environment for the execution of the tasks of both local and remote applications. The OurGrid Peer is the component that manages, at each administrative domain (or site), the set of grid machines that are made available to the grid by the corresponding site. In general, one peer is installed per site. A peer joins the grid by notifying a peer discovery service about its existence and it is immediately informed about the presence of other peers on the grid. The MyGrid Broker is responsible for providing an interface for users to submit their applications. It is also responsible for performing the scheduling of the applications on the grid machines and manage the execution of the scheduled applications. A user wishing to run an application must use the MyGrid broker to connect to a known peer (usually her own site's peer, named its *local* peer), becoming a local user of that peer.

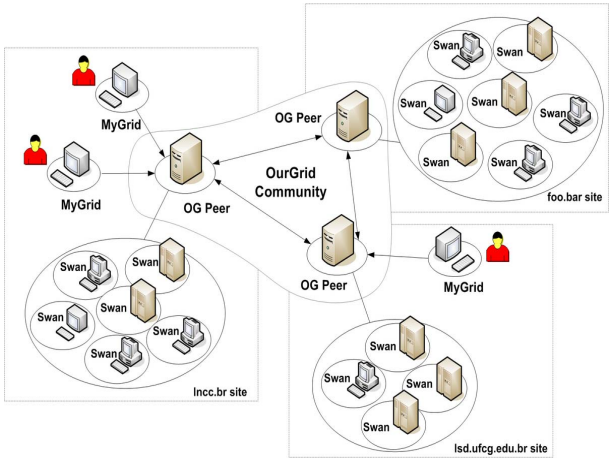

Figure 1. OurGrid's architecture

As mentioned before, parallel applications run on OurGrid are structured as a set of independent tasks (a bag-of-tasks). There are several ways to interact with the MyGrid Broker to submit applications, which will be further discussed in Section 3. No matter how the user interacts with the broker, when an application is submitted for execution the broker contacts its local peer asking for the number of grid machines that it requires to run the application (this request may carry specific attributes for the machines, such as a particular operating system, or a minimal amount of memory, etc). When the peer receives such a request it tries to find enough machines that are suitable to execute the application's tasks. If the peer cannot satisfy the request with its own local machines, it tries to obtain machines from other community peers, by forwarding the request to peers that may have suitable machines. It then waits these peers to deliver remote machines. Whenever new machines (local or remote) are made available the local peer delivers these machines to the requesting broker. As soon as one or more machines are delivered to the broker, it schedules tasks and starts the management of the their execution.

OurGrid has been built to be fast, simple, scalable and secure. It is fast as it allows users to improve the turn-around time of applications. This capability is provided by features of the MyGrid broker, such as scheduling with task replication and file transfer optimizations. The system's simplicity is mainly due to its capacity of hiding the grid heterogeneity behind MyGrid. To achieve scalability a peer-to-peer approach has been employed and the Network of Favors, a technology that promotes cooperation among peers, has been designed and implemented. On OurGrid, security is delivered to users by means of the sandboxing mechanism implemented by SWAN. It isolates a potentially malicious application inside a virtual machine, thus protecting the machine and the network from attacks. A more detailed discussion on the OurGrid approach to each one of these issues can be found in [11].

3 Applications

Since OurGrid appeared as a production grid, many users have benefited from its technology. Applications executing over the OurGrid Community concerns different areas and vary from computer science simulations to medical images processing. These applications share common requirements such as high processing demand and the ability of being parallelized as a BoT application.

OurGrid applications can be categorized by the way they interact with the middleware. There are four different approaches: script-based applications, embedded applications, framework-based applications, and portal-based applications. In this section, we are going to explain each of these interaction modes and exemplify them with concrete applications.

3.1 Script-based Applications

This is the simplest way to produce an application that interacts with the OurGrid middleware. This approach can be used by applications whose tasks can be defined as the execution of a program that reads input data from a file and outputs its result to another file. In this case the application is written as a text file that contains entries for all tasks that comprise the application, each entry being described as follows: i) a stage-in command used to transfer input data and executable code to the grid machine; ii) the command that will be executed at the grid machine; and, iii) a stage-out command that transfers the output of the execution to the user's machine. In most cases it is possible to use a script to automate the generation of the application description file, since there may be hundreds or thousands of tasks per application.

The main representant of this kind of applications are home-brewed simulators that are used to perform parameter sweep. In fact, many of our own research work was made possible thanks to the OurGrid Community [20, 22, 5, 7, 18].

3.2 Embedded Applications

In this kind of application, the application incorporates calls to the OurGrid middleware API. The use of grid resources is transparent to the application users. All the grid complexity is hidden from them. This approach is normally used when the application is more complex than those exemplified before. When using the OurGrid API directly,

users are able to access sophisticated features that are not available through the simpler script-based interaction mode.

One of the main applications currently using the OurGrid community following this approach is the SmartPumping system [9]. This application was developed in the context of a cooperation between the Universidade Federal de Campina Grande and PETROBRAS. Its goal is to provide support for the operation of complex and large-scale pipelines. A simulation-optimization approach is the strategy adopted for the network state prediction and control. The simulation module is based on the steady state hydraulics of the fluid flow. The system is modeled using genetic algorithms and the best action to be performed, according to the system overall evolution simulation, is suggested to the application user. After a period of testing at UFCG, the SmartPumping is now being deployed by PETROBRAS engineers in the target oil pipeline.

Another example of application developed using this interaction mode is CBIR-Grid - a content-based image retrieval (CBIR) application to support medical imaging diagnosis. Its development was part of GridVida a project funded by FINEP/MCT (a R&D funding agency of the Brazilian Ministry of Science and Technology). CBIR-Grid applies computational grid technology to speed up CBIR procedures, preserving security of data in clinics and hospitals. The application combines several techniques (texture and registration algorithms) to retrieve images with a greater-than-90% precision. Experiments show that running this application without OurGrid support would take hours, conversely, using the grid, it takes just a few minutes, making it usable in the clinical routine.

3.3 Framework-based Applications

A framework is a high-level, domain-specific software structure built in order to be reused in the development of derived applications (that inherits its design and code). To build her or his application, the user just needs to extend and/or customize the existing code.

One of the applications that currently use OurGrid is a framework to speed up the software testing process using a grid. GridUnit is an extension of the widely adopted JUnit testing framework, able to automatically distribute the execution of software tests on a computational grid with minimal user intervention [13]. The motivation for this application development is the increasing complexity of the systems, mainly distributed ones, and the need to reduce the test-time (that is paramount in agile methodologies that recommend the cycle test/develop/refactor). The solution does not require source code modifications, so the users' test-suites, written to JUnit, can easily execute on the grid middleware. Experiments reported have shown a speed up of up to 70x, reducing the test phase of a synthetic application from 24 hours to less than 30 minutes running on the grid.

3.4 Portal-based Applications

Grid portals have emerged with the vision of helping scientific users to execute their applications in a grid environment. Fundamentally, a grid portal is a web-based application (its interface is accessed through a web browser) that submits the execution of an application to the grid. This architecture hides the grid complexity from the user but she is aware that there is an underlying grid infrastructure.

The SegHidro Portal (available at http://portalseghidro.dca.ufcg.edu.br/) was developed in the context of the SegHidro project, also funded by FINEP/MCT. It supports the execution of coupled environmental computational models to enable a better management of hydrological resources. These environmental models are used to obtain weather forecasts or climate predictions, to simulate reservoir management and basins hydrological behaviors. Some of these models are very computing demanding. The simulation of a variety of environmental scenarios (a set of models that are composed following a simple workflow approach) demands large amounts of computing power. This problem motivated the development of the SegHidro portal and its integration with the OurGrid middleware. Users access the portal to compose their workflows, define simulations scenarios and submit it to execution. The SegHidro portal engine creates the jobs to be submitted to the grid, monitors its execution and collects the resulting files. An important requirement of this portal is its ability to execute not only the applications that already exists in the portal, but also others upload by the users. Users are free to include new applications to run on the grid environment [8].

Another portal that is implemented on top of the OurGrid Community is the EpanetGrid Portal. Algorithms for the functional analysis of water supply systems, particularly the water distribution networks, usually demand high processing capacity, due to the generation of a great number of design and/or operational scenarios (e.g., in reliability analysis), or due to time limit restrictions for the computer processing (e.g., in analysis and optimization of real-time system operation) [15]. Simulation of multiple scenarios on hydraulic models, such as EPANET, demands great computational processing power. In order to reduce this processing time, it was performed the integration of EPANET with the OurGrid Community through the EpanetGrid Portal. The main advantage of this approach is the possibility of using a great number of remote computational resources at a low cost, since the users of this kind of application (governmental agencies, water supply public companies, cooperatives) usually do not have access to them.

4 Benefits and Hurdles

4.1 Benefits

By observing the evolution of the OurGrid Community during a certain period of time and analyzing the grid usage, it is possible to understand which benefits were obtained by OurGrid users. We have analyzed the system's logs for the time period spanning from March to November, 2005. On the beginning of March the OurGrid Community had about 5 active peers and about 60 machines. The community had been in production since December 2004. From April to October, the number of peers has steadily grown to about 30 peers, comprising about 250 active machines. The major increases on the number of peers took place shortly after a new and more stable version of OurGrid has been released, specifically, after the release of version 3.2, in July, 2005.

The benefits obtained by OurGrid users in terms of available computing resources are apparent when we measure the percentage of the available machines that were donated by the peers. If a peer is consuming a donated machine, it means that its local resources are not enough to satisfy this peer's users. From March to May, about 40% of the grid machines were being donated. On the highest utilization levels, about 80% of the machines were donated. Note that the percentage of total used machines is typically higher, since it also includes the local resources owned by the peers.

The impact that the use of the OurGrid Community has on the turn-around time of applications has been reported by many users. For instance, results reported in [19] show that the execution of sagittal knee and axial head image comparisons using CBIR-Grid had a reduction in processing time of as much as 117 and 95 minutes, respectively, when compared to the processing time obtained when the same comparison is executed in a single machine. Another application that reported impressive speed ups is GerPavGrid. This application is a distributed system used in the city of Porto Alegre, located in the south of Brazil, to maintain and plan the investments in the city road system. The processing time to generate one of the most complex reports (Porto Alegre road system) when using the OurGrid Community was approximately 80 times faster than its sequential version [21].

4.2 Hurdles

The development of a multi-threaded distributed system is, commonly, a complex activity. Grids add new challenges due to their wide-dispersion, loose coupling, and presence of multiples administrative domains. Moreover, developing a complex software with production quality within the academic environment makes the task even more challenging.

Since 2003, when we start developing OurGrid, a number of people took part in the development team and has now left our lab as they follow their natural path from undergraduate students to professionals in the industry or in the academy, possibly after getting a graduate degree. This means that the development team has always been heterogeneous and dynamic. Besides, some of ourGrid's functionalities have been developed by independent teams. All this causes great concerns with respect to code quality, and makes the use of automatic test techniques a must. However, providing high-quality automatic test for a grid solution is a not an easy task [3]. We noticed, during OurGrid development, that the state-of-practice techniques used for automatic test of distributed and multi-threaded software is not always realistic, since they rely on the use of timeouts and can fail both due to a insufficient time to wait before an assertion is performed as well as due to a bug. As a result of this limitation, OurGrid versions have been launched carrying bugs, that although difficult to detect at testing time, would frequently appear when the system was deployed in a real setting. Some solutions to support more deterministic tests, using Aspect-Oriented Programming (AOP) or Java 5 new features, have been developed by the project [3]. The effectiveness of these solutions are still to be assessed.

One of OurGrid's remarkable characteristic is allowing the creation of free-to-join communities. There is no human negotiation to define who is going to access the grid or how long the access will take. Users or computational resources do not need identification or authentication to join the community, and they can remain anonymous in the system for an undefined period of time. This clearly raises security issues. There is a need not only to protect resources from malicious applications as well as protecting applications from malicious resources. The latter issue is left for the application developer to handle, while SWAN is OurGrid's solution for the former issue. Unfortunately, SWAN's implementation is based on the Xen virtual machine technology [12], which makes the use of SWAN limited to some Linux distributions and adds some difficulty in setting up a secure OurGrid site. Our experience has shown that in an open grid like the ones supported by the OurGrid middleware, defining a one-size-fits-all security solution is not the most advisable approach. We are currently developing a number of alternative security solutions, leaving to the site administrators the task to define the most suitable trade-off between security level and installation complexity.

5 Concluding Remarks

OurGrid is an open free-to-join collaborative grid that caters for BoT applications. OurGrid is in production since December 2004 and has been used by many real applications. Joining is automatic; no paperwork or approvals of any sort are required. The vision is that OurGrid provides a massive worldwide computing platform on which research

labs can trade their spare compute power for the benefit of all. Naturally, OurGrid keeps evolving to be simpler, faster, and more robust. We highlight as critical aspects where improvement is needed: i) developing effective techniques to perform automatic tests; and, ii) increase the portfolio of security solutions.

Acknowledgments. Many thanks to the OurGrid team. This work would have not happen without everybody's commitment to do their best. Thanks also to the OurGrid users and contributors. You are the ones who make this project alive. Francisco Brasileiro would like to thank the financial support from CNPq/Brazil (grant 300.646/96). This work was partially developed in collaboration with HP Brazil R&D.

References

[1] Globus web site. http://www.globus.org.

[2] SETI@home web site. http://setiathome.ssl.berkeley.edu/.

[3] W. C. A. Dantas and K. Saikoski. Using AOP to Bring a Project Back in Shape: The OurGrid Case. *Journal of the Brazilian Computer Society* , 11(3):735–746, April 2006.

[4] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: an experiment in public-resource computing. *Comm. ACM*, 45(11):56–61, Nov. 2002.

[5] N. Andrade, F. V. Brasileiro, W. Cirne, and M. Mowbray. Discouraging Free Riding in a Peer-to-Peer CPU-sharing Grid. In *Proc. 13th IEEE Symp. High Performance Distributed Computing, Honolulu, Hawaii*, pages 129–137, June 2004.

[6] N. Andrade, W. Cirne, F. V. Brasileiro, and P. Roisenberg. OurGrid: An approach to easily assemble grids with equitable resource sharing. In *Proc. 9th Workshop on Job Scheduling Strategies for Parallel Processing*, volume 2862 of *Lecture Notes in Computer Science*, pages 68–86. Springer-Verlag Heidelberg, June 2003.

[7] N. Andrade, M. Mowbray, W. Cirne, and F. V. Brasileiro. When Can an Autonomous Reputation Scheme Discourage Free-riding in a Peer-to-Peer System? In *Proc. 4th Workshop on Global and Peer-to-Peer Computing (GP2PC 04), Chicago, USA*, Apr. 2004.

[8] E. C. Araujo, W. Cirne, G. Mendes, N. Oliveira, E. P. Souza, C. Galvao, and E. S. Martins. The seghidro experience: Using the grid to empower a hydro-meteorological scientific network. In *Proc. of the 1st IEEE International Conference on e-Science and Grid Computing e-Science 2005, Melbourne, Australia* , Melbourne, Australia, 2005.

[9] E. Brasileiro, C. Galvão, and F. Brasileiro. Otimização do Controle de Redes de Escoamento de Petróleo. *Petro Química*, XXIII(273):153–157, 2005.

[10] E. Cavalcanti, L. Assis, M. Gaudencio, W. Cirne, F. Brasileiro, and R. Novaes. Sandboxing for a free-to-join grid with support for secure site-wide storage area. In *Proc. of the First International Workshop on First International Workshop on Virtualization Technology in Distributed Computing*, Tampa, USA, November 2006.

[11] W. Cirne, F. Brasileiro, N. Andrade, L. Costa, A. Andrade, R. Novaes, and M. Mowbray. Labs of the world, unite!!! *Journal of Grid Computing*. To appear. Available as technical report at http://www.ourgrid.org/twiki-public/pub/OG/OurPublications.

[12] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the art of virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, Oct. 2003.

[13] A. N. Duarte, W. Cirne, F. Brasileiro, and P. Machado. Gridunit: Software testing on the grid. In *Proc. 28th ACM/IEEE International Conference on Software Engineering ICSE 2006, Shanghai, China* , Shanghai, China, 2006.

[14] I. Foster and C. Kesselman. *The Grid: Blueprint for a New computing Infrastructure*. Morgan Kaufmann, 1998.

[15] C. Galvao, W. Cirne, F. Brasileiro, E. Brasileiro, and E. C. Araujo. Computação em grade aplicada à análise do abastecimento de água. In *Proc. Seminario Hispano-Brasileno sobre Sistemas de Abastecimento Urbano de Agua IV SEREA, Joao Pessoa, Brasil*, Joao Pessoa, Brasil, 2004.

[16] D. Kondo, A. Chien, and H. Casanova. Resource management for short-lived applications on enterprise desktop grids. In *Proceedings of Supercomputing 2004*, page 17, Pittsburgh, PA, USA, Nov 2004.

[17] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *Proc. 8th Int'l Conf. Distributed Computing Systems*, 1988.

[18] R. V. Lopes, W. Cirne, F. V. Brasileiro, and E. Colaço. Can dynamic provisioning and rejuvenation systems coexist in peace? In *DSOM*, pages 245–256, 2005.

[19] M. C. Oliveira, J. F. M. Vieira, P. Marques, and W. Cirne. Aplicacao de grids computacionais para otimizar a recuperacao de imagens medicas baseada em conteudo. In *Proc. Congresso Brasileiro de Engenharia Biomédica XX CBEB, Sao Pedro, Brasil* , São Pedro, Brasil, 2006.

[20] D. Paranhos, W. Cirne, and F. V. Brasileiro. Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational Grids. In *Euro-Par 2003: Int'l Conf. Parallel and Distributed Computing*, volume 2790 of *Lecture Notes in Computer Science*, pages 169–180. Springer, Jan. 2003.

[21] C. A. F. D. Rose, T. C. Ferreto, M. B. Farias, V. G. Dias, W. Cirne, M. P. M. Oliveira, K. Saikoski, and M. L. Danieleski. Gerpavgrid: using the grid to maintain the city road system. In *SBAC-PAD '06: Proceedings of the 18th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'06)*, pages 73–80, Washington, DC, USA, 2006. IEEE Computer Society.

[22] E. L. Santos-Neto, W. Cirne, F. V. Brasileiro, and A. Lima. Exploiting Replication and Data Reuse to Efficiently Schedule Data-intensive Applications on Grids. In *Proc. 10th Workshop on Job Scheduling Strategies for Parallel Processing* , pages 123–135, New York, USA, June 2004.