Complex System Simulations with QosCosGrid

Krzystof Kurowski[4,5], Walter de Back[2], Werner Dubitzky[3], Laszlo Gulyás[1,2], George Kampis[2], Mariusz Mamonski[4], Gabor Szemes[1,2], and Martin Swain[3]

[1] AITIA International Inc., Budapest, Hungary
[2] Collegium Budapest – Institute for Advanced Study, Budapest, Hungary
[3] University of Ulster, Coleraine, United Kingdom
[4] Poznan Supercomputing and Networking Center, Poznan, Poland
[5] University of Queensland – Institute for Molecular Bioscience, Australia
krzysztof.kurowski@man.poznan.pl, wdeback@colbud.hu, w.dubitzky@ulster.ac.uk, lgulyas@colbud.hu, gkampis@colbud.hu, mamonski@man.poznan.pl, gszemes@colbud.hu, mt.swain@ulster.ac.uk

Abstract. The aim of the QosCosGrid project is to bring supercomputer-like performance and structure to cross-cluster computations. To support parallel complex systems simulations, QosCosGrid provides six reusable templates that may be instantiated with simulation-specific code to help with developing parallel applications using the ProActive Java library. The templates include static and dynamic graphs, cellular automata and mobile agents. In this work, we show that little performance is lost when a ProActive cellular automata simulation is executed across two distant administrative domains. We describe the middleware developed in the QosCosGrid project, which provides advance reservation and resource co-allocation functionality as well as support for parallel applications based on OpenMPI (for C/C++ and Fortran) or ProActive for Java. In particular, we describe how we modified ProActive Java to enable inter-cluster communication through firewalls. The bulk of the QosCosGrid software is available in open source from the QosCosGrid project website: www.qoscosgrid.org.

Keywords: Grid computing, complex system, parallel applications, ProActive, Java, advance reservation, co-allocation, modeling and simulation.

1 Introduction

Complex systems are defined as systems with many interdependent parts which give rise to non-linear and emergent properties determining the high-level functioning and behavior of such systems [1]. Due to the interdependence of their constituent elements and other characteristics of complex systems, it is difficult to predict system behavior based on the 'sum of their parts' alone. Examples of complex systems include bee hives, bees themselves, human economies and societies, nervous systems, molecular interactions, cells and living things, ecosystems, as well as modern energy or telecommunication infrastructures. Arguably one of the most striking properties of complex systems is that conventional experimental and engineering approaches are inadequate to capture and predict the behavior of such systems. To complement the

G. Allen et al. (Eds.): ICCS 2009, Part I, LNCS 5544, pp. 387–396, 2009.

conventional experimental and engineering approaches, computer-based simulations of complex natural phenomena and complex man-made artifacts are increasingly employed across a wide range of sectors. Complex systems simulations often require considerable compute power.

The present paper describes the perspective and initial results of the QosCosGrid (Quasi-Opportunistic Supercomputing for Complex Systems) project which addresses the computationally intensive simulation of complex systems using parallel methods and grid technology. We consider the QosCosGrid approach from the perspective of the computational scientist, i.e. the user, as well as from the technology side.

Complex system simulations often use supercomputers because of the high data volume and computing requirements of the individual computations, but also because the high communication overhead between the computation tasks on individual elements. However, supercomputers are expensive to acquire and maintain and, therefore, many users do not have access to such technology. Recently, local clusters, such as Beowulf clusters and other multi-core and multi-machine systems, have become the technology of choice for many complex systems modelers. However, with the advent of flexible modeling tools, complex systems simulations have become more and more comprehensive and complex. As a result, local clusters are increasingly inadequate to satisfy the required computing and communication needs. QosCosGrid aims to address this gap by facilitating supercomputer-like performance and structure through cross-cluster computations.

In grid computing, a typical assumption is that the availability of computing resources depends on their local usage patterns. This opportunistic mode of grid computing is highly problematic for a complex system simulation. Complex system simulations are characterized by a high degree of dependency among the parallel computations that comprise the simulation. To address this problem, QosCosGrid has developed advance reservation and topology-aware methods.

2 The User Perspective

2.1 Problems of Complex System Modeling

Complex system simulations are now used across a wide range of scientific fields. However, the challenges faced by computational scientists and their demands for high performance computing differ substantially. Typically, there are many different approaches to model and simulate a concrete complex system. Each approach comes with its own requirements in terms of methods and computing technology. In an attempt to address a wide range of complex systems, we developed a classification of the computing requirements for different complex systems simulation scenarios. Key to this classification scheme is the required communication topology that reflects the element-to-element interaction characteristics of the underlying complex system.

2.2 Partitioning Complex System Simulations

The main problem in the parallelization of complex system simulations is their high demand for inter-process communication. However, parallel computing applications are most efficient when there is no or little demand for inter-node communication. In order to minimize execution time of a distributed complex system simulation, the computation needs to be partitioned such that communication between interacting components is minimized. In addition, the computational load of each partition, determining the time between communication events, needs to be balanced.

Instead of creating individual parallel solutions for every single complex system simulation, we developed a categorization of the required communication topologies (Fig. 1). The categorization scheme consists of six categories referred to as *communication templates*: ***Template T0*** is the simplest communication template. *T0* has no communication between the components, so the partitioning is only constrained by load-balancing considerations. Perfectly (or "embarrassingly") parallel applications, such as parameter sweeping, falls into the *T0* category. ***Template T5*** represents the other end of the spectrum of communication templates. Essentially, *T5* covers those complex systems for which it is difficult or impossible to determine a meaningful communication template or partitioning of the computations. ***Template T1*** describes the communication topology of a non-spatial complex system, with fixed and a priori known element interaction pattern. ***Template T2*** applies to complex systems whose element interaction patterns are changing over time. Hence the communication topology needs to be defined by some graph transition function that provides enough information for the system's efficient distribution. ***Template T3*** covers classical cellular automata simulations. In this case, the definition of a meaningful partitioning is straightforward. ***Template T4*** describes an agent-based simulations where mobile agents are embedded in a (spatial) coordinate system and have only local interactions.

Various partitioning algorithms are available for each of these communication templates. The categorization is aimed at guiding users to understand the structure of a given complex system simulation and to inform the choice of parallelization techniques. Hence, the scheme provides a useful guide for parallelizing of complex systems simulations.

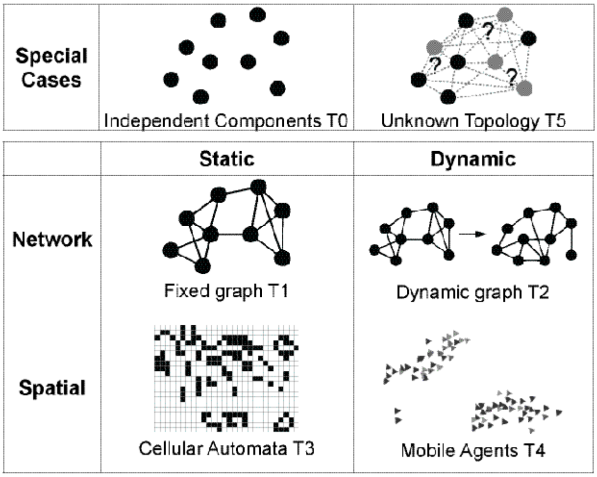

Fig. 1. Communication templates supported in the QosCosGrid system

2.3 Developing and Deploying QosCosGrid Applications

For end users, the main benefit of QosCosGrid lies in the transparency of grid-enabling complex system applications. The QosCosGrid middleware, described below, transparently handles all intricate grid aspects when it deploys complex system applications across clusters. From the perspective of the application developer, therefore, there is no difference between developing code to be deployed on a supercomputer or on a grid. Moreover, legacy parallel code can be executed on QosCosGrid without any reimplementation.

To further enhance the ease-of-use of deploying applications on the grid, a comprehensive Web interface has been developed. The QosCosGrid portal website[1] facilitates monitoring of the grid infrastructure, including bandwidth and latency, monitoring of status, usage and (advance) reservations, grid file transfer to upload applications and download results.

3 The Technical Perspective

The overall QosCosGrid architecture and the key middleware are presented in Fig. 1. Based on various middleware services, QosCosGrid components can be grouped into two levels: The *Grid* level domain (consisting of multiple organizations) and the *Administrative* level (a single organization, typically providing and sharing one or a small number of computing clusters). At one of the lowest layers there is the Local

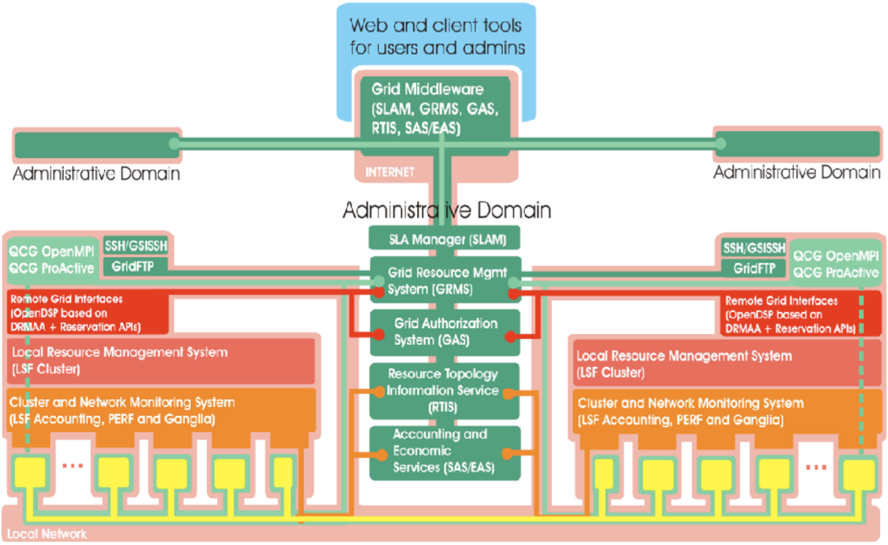

Fig. 2. The QosCosGrid multi-tier architecture connecting different middleware services as well as parallel application development and deployment tools

[1] http://node2.qoscosgrid.man.poznan.pl/gridsphere/gridsphere/guest/testbed/r/

Resource Management System supporting job submission, control and advance reservation (AR) mechanisms. Currently, QosCosGrid uses Platform Computing's Load Sharing Facility (LSF). We have also successfully tested the QosCosGrid middleware with the Sun Grid Engine and there are possible extensions for PBSPro or Maui. Above this layer, there is the OpenDSP service that communicates with LSF using the well-adopted standard DRMAA interface that communicates with the local queuing system. OpenDSP exposes its functionality remotely with the OGSA-BES HPC Profile compliant WS interface, and, to our knowledge, is the most efficient remote multi-user access to underlying queuing systems. On top of the OpenDSP service, there is the GRMS (Grid Resource Management Service), a main meta-scheduler service, which acts both at the Grid and Administrative level. From the end user perspective, the GRMS provides its own job description language (called Job Profile) that allows the topology requirements of various jobs, including sequential, parallel, massively parallel (with communication topology requirements) as well as workflow jobs to be defined and controlled on behalf of end users.

Below we briefly describe two main programming and execution environments, ProActive and OpenMPI, which have been successfully integrated with OpenDSP and GRMS to allow end users to perform multi-cluster job submission and control. These well-known environments have been modified according to the complex system requirements we collected. They can be distributed as powerful tools for large-scale and long-term computation analysis involving many computing clusters located in various administrative domains. The QosCosGrid middleware offers also many management features for local IT administrators responsible for sharing computing resources among many end users and their applications.

OpenMPI. The MPI (Message Passing Interface) is a leading standard in the domain of parallel scientific applications. It provides end users with both the programming interface consisting of simple communication primitives and the environment for spawning and monitoring MPI processes. A large number of implementations of the MPI standard is available (both as commercial and open source). In QosCosGrid, it was decided that MPI serves as the input for a new OpenMPI[2] distribution and we added enhancements to this implementation. Of key importance were the advance inter-cluster communication techniques that deal with firewalls and Network Address Translation. In addition, the mechanism for spawning new processes in OpenMPI needed to be integrated with QosCosGrid-developed middleware. The extended version of the OpenMPI framework was named QCG-OMPI [2] (where QCG stands for QosCosGrid).

ProActive Java. Even though enhanced OpenMPI is the primary execution environment in QosCosGrid, the existence of legacy Java applications based on the Repast Agent Simulation Toolkit [3] led to the decision to provide support for a new Java-based parallel programming environment called ProActive [4]. The ProActive library, by default, uses the standard Java RMI framework as a portable communication layer. With a reduced set of simple primitives, ProActive (version 3.9 as used in QosCosGrid) provides a comprehensive toolkit that simplifies the programming of applications distributed on local area networks, clusters, Internet grids and peer-to-peer

[2] http://www.open-mpi.org

intranets for Java-based applications. However, when we designed QosCosGrid, the standard ProActive framework did not provide any support for multi-user environments, advance reservation and cross-cluster co-allocation. To satisfy the requirements of complex system simulation applications and users, we developed extensions to the ProActive library (called QCG-ProActive) with the following goals: (1) To preserve standard ProActive library properties (i.e., allow legacy ProActive applications to be seamlessly ported to QosCosGrid). (2) To provide end users with a consistent GRMS Job Profile schema as a single document used to describe application parameters required for execution as well as resource requirements (in particular network topology and estimated execution time). (3) To prevent end users from the necessity to have direct (i.e., over SSH) access to remote clusters and machines.

Cross-site QCG-ProActive Deployment Model. ProActive deployment involves the starting of the main application and the subsequent spawning on other machines (starting ProActive Runtimes). In the context of service-level agreements and accountability, this is problematic, because it is difficult to guarantee the atomicity of such a two-phase deployment process. Application and ProActive Runtimes cannot be started at the same time because ProActive Runtimes needs to know some callback information that is used to contact the main application. Therefore, we proposed to create an external service in QosCosGrid called ProActive Node Coordinator (PNC) that helps to exchange initial arguments between the master application and ProActive Runtimes. Consequently, a local queuing system (OpenDSP with LSF) does not have to start ProActive Runtimes directly. Instead, we need to provide an appropriate wrapper script – the QosCosGrid ProActive Wrapper, which connects the PNC service and synchronizes initial data required to start ProActive Runtimes properly. Additionally, in order to support a cross-cluster ProActive deployment, before any job submission request to LSF via OpenDSP, there is an appropriate advance resource reservation call. The main difference, from the end user perspective, is that instead of a typical PAD file (ProActive Deployment Descriptor) the GRMS Job Profile is used as a language to describe ProActive application requirements and then the Job Profile is automatically converted to the corresponding PAD.

Inter-cluster QCG-ProActive Communication Mechanism. The basic ProActive Library transport layer is based on Java RMI. The RMI communication usually consumes one TCP/IP port per remote object instance and ports are randomly selected. Moreover, it is almost impossible to configure a firewall to forward the RMI traffic, which was an important issue for incorporating ProActive into QosCosGrid. To deal with this problem, the ProActive application can be configured to use the RMISSH communication protocol that simply creates on demand SSH tunnel for each outgoing RMI connection. Unfortunately, this solution does not work with sites behind Network Address Translation and it requires password-less authentication to be configured for each machine (and for every user) in the whole grid. Thus, in QosCosGrid project we proposed to use SOCKS server as the basic way to tunnel RMI traffic between clusters behind Network Address Translations and firewalls.

4 Results

The aim of QosCosGrid is to provide supercomputing-like structure and performance to cross-cluster computations for complex systems simulations. In order to assess the performance of the infrastructure provided by QosCosGrid, performance tests were conducted on a simple test bed. We ran performance tests using a cellular automaton simulation application as a benchmark application. A cellular automaton is a lattice of finite state machines, in which the state of a cell is determined by the states the neighboring cells in the previous time step [5]. Cellular automata are a widely used methodology to study spatial phenomena in physics and biology.

The regular lattice interaction topology of cellular automata makes them a prime candidate for parallelization and deployment on a supercomputer. A cellular automaton allows for simple partitioning by mapping each part to a different core or processing element. After each iteration, state information on the border cells between adjacent partitions is exchanged. As the size of a cellular automaton simulation increases, the computational cost grows quadratically, while the communication costs grow only linearly.

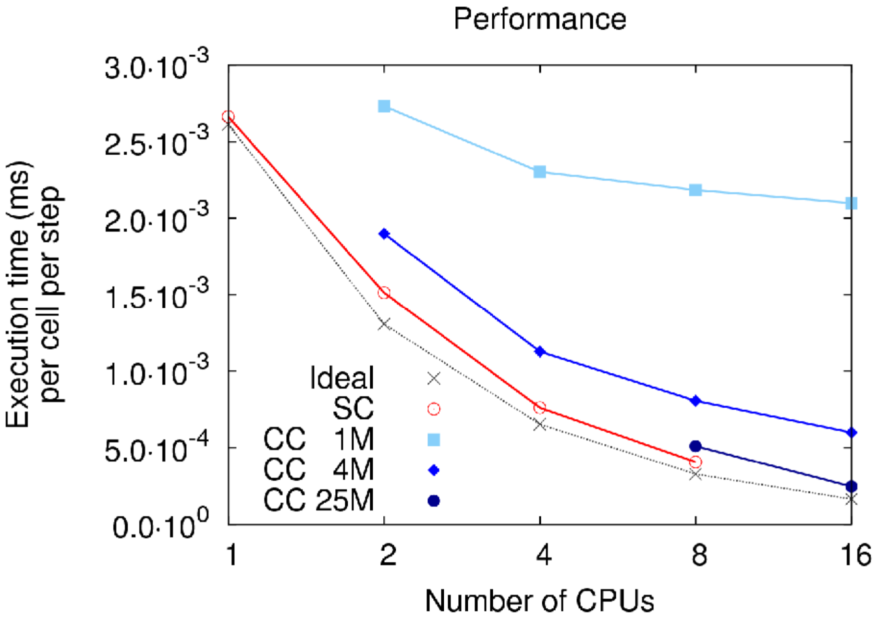

Fig. 3. Performance of a cellular automaton (CA) of increasing size (1000 x 1000, 2000 x 2000, 5000 x 5000). The line with cross-symbols shows ideal performance (execution time of non-parallel CA divided by number of cores). The line labeled SC shows single cluster runs, the CC 4M line shows cross-cluster run performance in the QosCosGrid test bed. Legend: SC = single cluster; CC = cross-cluster; 1 M = 1 million cells = 1000 x 1000; 4 M = 4 million cells = 2000 x 2000; 25 M = 25 million cells = 5000 x 5000.

Cellular automata are frequently deployed on supercomputing facilities. In this study we used cellular automata as a benchmark system to assess the performance of the QosCosGrid technology. The cellular automaton used in our study was implemented in Repast J 3.0 and parallelization was performed using QCG-ProActive. We tested the performance in a heterogeneous cross-cluster environment, and conducted experiments on two small clusters (8 cores each) located in Paris (France) and Poznan (Poland) and connected via the Internet (no dedicated network). We compared the execution time of a parallel implementation of a cellular automaton executed on a single cluster, with the execution time in a cross-cluster run.

The results (Fig. 3) describe performance data for three different cellular automaton sizes. These results confirm that parallel execution in a single cluster is highly efficient for all sizes. For relatively small cellular automaton problems (i.e., 1000 x 1000), the cross-cluster performs very poorly, as the execution time using 16 cores is similar to the performance of the non-parallel cellular automaton on a single core. As the size of the cellular automaton increases, however, the performance of cross-cluster execution approaches the performance in a single cluster. For the largest cellular automaton in the performance test (5000 x 5000), the performance of the cross-cluster execution becomes almost indistinguishable from the single cluster performance.

These preliminary results show that parallel complex systems simulations with a relatively high computation/communication ratio can be meaningfully deployed on the QosCosGrid grid infrastructure.

5 Discussion

QosCosGrid is an end-to-end solution that is already being used by scientists for compute intensive parallel applications in the field of complex systems modelling. The advantage of QosCosGrid is that it requires no dedicated network connections or specific configurations. It is a multi-user environment that efficiently controls access to computing resources in different administrative domains. It is specifically designed for non-trivial parallel computations using Open-MPI with C/C++ and Fortran, or ProActive for Java-based legacy applications. Additional features of interest to scientists include its user-friendly Web-based user interface and its reusable application schemas or template categories based on ProActive.

QosCosGrid is aiming to provide a high quality of service (for users) equivalent to that of a dedicated supercomputing facility. To achieve this, QosCosGrid provides services such as co-allocation and advance resource reservation [6], which are very difficult to provide in dynamic grid environments. Recent efforts in these areas include that by Elmroth and Tordsson for the NorduGrid/ARC middleware [7] and the work by Kyriazis and colleagues [8] who concentrate on orchestrating grid resources in order to support application workflows. More established solutions are provided by Condor-G [9], and Nimrod-G [10]. After a careful review of existing open solutions and standards, QosCosGrid has based its services on top of exiting third-party software. In addition, the QosCosGrid Gateway is a key feature for high-quality of service as it provides essential support for both users and administrators.

Mateos *et al.* [11] provide a useful survey that compares a number of different approaches to grid-enabling applications. They discuss a number of other Java tools used in grids, apart from ProActive, such as Satin [12], which is based on Ibis [13], and "grid-aspecting" [14], which is based on AspectJ. The difference between these Javas and ProActive is in the degree of granularity and the amount of modification required to the application source code. Both ProActive and Satin require more extensive modifications to the source than grid-aspecting, but an advantage of these modifications is that applications can make more sophisticated and efficient use of grid resources.

There are other projects which have aims that are similar to those of the QosCosGrid project, these include EGEE [15], HPC4U [16] and DEISA [17]. However, these projects do not provide all the functionality of QosCosGrid. EGEE does not support all of QosCosGrid's quality-of-service components such as advance reservation and check-pointing. HPC4U concentrates on parallel executions performed within clusters as opposed to QosCosGrid's cross-cluster execution environment. DEISA, the Distributed European Infrastructure for Supercomputing, is a grid of supercomputers suitable for petascale applications whereas QosCosGrid is a more "humble" system designed to enable distributed clusters to be combined into a resource with compute power similar to a single supercomputer.

6 Conclusions

This paper describes how QosCosGrid enables clusters in different administrative domains to be welded (virtually) into a single powerful compute resource which we call a *quasi-opportunistic supercomputer*. We outlined the middleware that we have developed to achieve this and although QosCosGrid provides extensive support via OpenMPI for parallel C/C++ and Fortran applications, here we have mainly focussed on the modifications we made to ProActive Java for supporting inter-cluster communications and dealing with firewalls and Network Address Translations. Our results, which are based on an ecological simulation using parallelized cellular automata, demonstrate the feasibility of running non-trivial parallel simulations across administrative domains located in different European countries. For large simulations there is only a minor reduction in performance when running an inter-cluster simulation compared to a single cluster simulation. QosCosGrid is a largely open-source project which is due to be completed in mid-2009.

Acknowledgments. The work described in this paper is supported by EC grant QosCosGrid IST FP6 STREP 033883.

References

1. Special edition: Complex Systems. Science 284, 5411, 1–212 (1999)
2. Coti, C., Herault, T., Peyronnet, S., Rezmerita, A., Cappello, F.: Grid Services for MPI Cluster Computing and the Grid. In: 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2008), pp. 417–424 (2008)
3. North, M.J., Collier, N.T., Vos, J.R.: Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit. ACM Transactions on Modeling and Computer Simulation 16(1), 1–25 (2006)

4. Caromel, D., Delbe, C., di Costanzo, A., Leyton, M.: ProActive: an Integrated Platform for Programming and Running Applications on Grids and P2P systems. Computational Methods in Science and Technology 12(1), 69–77 (2006)
5. Farmer, J.D., Toffoli, T., Wolfram, S. (eds.): Cellular Automata. Elsevier, Amsterdam (1984)
6. Kravtsov, V., Schuster, A., Carmeli, D., Kurowski, K., Dubitzky, W.: Grid-enabling complex system applications with QosCosGrid: An architectural perspective. In: Proceedings of The International Conference on Grid Computing and Applications (GCA 2008), Las-Vegas, USA (2008)
7. Elmroth, E., Tordsson, J.: Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions. Fut. Gen. Comput. Sys. 24(6), 585–593 (2008)
8. Kyriazis, D., Tserpes, K., Menychtas, A., Litke, A., Varvarigou, T.: An innovative workflow mapping mechanism for Grids in the frame of Quality of Service. Fut. Gen. Comput. Sys. 24(6), 498–511 (2008)
9. Frey, J., Tannenbaum, T., Mirnon, L., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing 3, 237–246 (2002)
10. Abramson, D., Buyya, R., Giddy, J.: A computational economy for grid computing and its implementation in the Nimrod-G resource broker. Fut. Gen. Comput. Sys. 18(8), 1061–1074 (2002)
11. Mateos, C., Zunino, A., Campo, M.: A survey on approaches to gridification. Softw., Pract. Exper. 38(5), 523–556 (2008)
12. Wrzesinska, G., Maassen, J., Verstoep, K., Bal, H.E.: Satin++: Divide-and-Share on the Grid. In: Second IEEE International Conference on e-Science and Grid Computing (e-Science 2006) (2006)
13. van Nieuwpoort, R.V., Maassen, J., Wrzesinska, G., Hofman, R., Jacobs, C., Kielmann, T., Bal, H.E., Henri, E.: Bal: Ibis: a Flexible and Efficient Java based Grid Programming Environment. Concurrency and Computation: Practice and Experience 17(7-8), 1079–1107 (2005)
14. Maia, P.H., Mendonça, N.C., Furtado, V., Cirne, W., Saikoski, K.: A process for separation of crosscutting grid concerns. In: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC 2006, Dijon, France, April 23 - 27, 2006, pp. 1569–1574. ACM, New York (2006)
15. Gagliardiand, F., Jones, B., Grey, F., Bgin, M.-E., Heikkurinen, M.: Building an infrastructure for scientific grid computing: Status and goals of the EGEE project. In: Philosophical Transactions A of the Royal Society: Mathematical, Physical and Engineering Sciences, vol. 363(833), pp. 1729–1742 (2005)
16. Hovestadt, M.: Operation of an SLA-aware grid fabric. IEEE Trans. Neural Networks. 2(6), 550–557 (2006)
17. Lederer, H., Hatzky, R., Tisma, R., Bottino, A., Jenko, F.: Hyperscaling of plasma turbulence simulations in DEISA. In: Proceedings of the 5th IEEE Workshop on Challenges of Large Applications in Distributed Environments, Monterey, California, pp. 19–26 (2007)