CMST SI(1) 47-56 (2010) DOI:10.12921/cmst.2010.SI.01.47-56

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **Special Issue 2010,** 47-56

Parallel Large Scale Simulations in the PL-Grid Environment

Krzysztof Kurowski, Tomasz Piontek, Piotr Kopta, Mariusz Mamoński, Bartosz Bosak

Poznań Supercomputing and Networking Center, Poznań, Poland
e-mail: {kurowski/piontek/pkopta/mamonski/bbosak}@man.poznan.pl

(Received: 25 August 2010; revised: 29 October 2010; published online: 23 November 2010)

Abstract: The growing demand for computational power causes that Grids are becoming mission-critical components in research and industry, offering sophisticated solutions in leveraging large-scale computing and storage resources. The nature a Grid in which resources are usually shared among multiple organizations offering resources under their control based on the "best effort" approach with no guarantee concerning the quality-of-service may be inadequate to support large-scale simulations. Requirements of such simulations often exceed capabilities of a single computing center causing the need to simultaneously allocate and synchronize resources belonging to many administrative domains whose functionality is missing in leading grid middlewares preventing researchers from executing large-scale simulations in grids. The paper presents tools and services that were designed to build multilayered infrastructure capable of dealing with computationally intensive large-scale simulations in the grid environment. The developed and deployed middleware enables computing clusters in different administrative domains to be virtually welded into a single powerful compute resource that can be treated as a quasi-opportunistic supercomputer. We describe the middleware developed in the QosCosGrid project and being enhanced under the PL-Grid national grid initiative, which provides advance reservation and resource co-allocation functionality as well as support for parallel large-scale applications based on OpenMPI (for C/C++ and Fortran) or ProActive for Java.

Key words: grid computing, MPI, co-allocation, advance reservations

I. INTRODUCTION

Grid computing systems could be viewed as large-scale computing systems with considerable levels of hardware resources, but without the features that make supercomputers so powerful. In particular, grids usually lack sophisticated support for highly parallel applications with significant inter-process communication requirements. Grid computing environments are based on the heterogeneous, widely dispersed and time-variant resources which typically lack central control. Connected via local and wide area networks, grids typically rely on an opportunistic marshalling of resources into a coordinated action to meet the needs of large-scale computing applications. Grids are often offered as panacea for all kinds of computing applications, including those that require supercomputing-like environments. However, this vision of grids as virtual supercomputers is unattainable without overcoming the performance and reliability issues plaguing current grids.

The demanding nature of scientific simulations and problems modelling requires an environment which is able to simultaneously manage many kinds of resources, such as computing resources, storage and network to guarantee the level of the Quality of Service (QoS) required by advanced simulations. Addressing the need of scientists to run large scale simulations, a system able to bring supercomputer-like performance and structure to cross-cluster computations was designed and developed in the QosCosGrid FP 6 project [1] and then adopted and extended in the PL-Grid [2] national grid initiative. Tools and services, further referred to as QosCosGrid middleware, were designed to build multilayered infrastructure being capable of dealing with computationally intensive large scale simulations. The developed and deployed middleware enables computing clusters in different administrative domains to be virtually welded into a single powerful computing resource that can be treated as a quasi-opportunistic supercomputer, whose computational power exceeds the power offered by a single administrative domain (data center).

The remaining part of this paper is organized as follows. In Section II, the project motivations backed up by the PL-Grid survey results are presented. Section III gives a general view of the QosCosGrid architecture and describes its mains components. Section IV presents example deployments of the QosCosGrid infrastructure while Section V concludes this paper.

II. MOTIVATIONS

The PL-Grid Project is funded under the framework of the Innovative Economy Operational Programme. The main purpose of this Project is to provide the Polish scientific community with an IT platform based on Grid computer clusters, enabling e-science research in various fields. One of the main assumptions of the project is a close collaboration with users to identify their needs and requirements and to make the project user-driven. In order to gather the aforementioned preferences, requirements and needs in the PL-Grid project, it was decided to carry out, as an iterative process, a specially prepared survey (the "PL-Grid Survey"[1]). The questionnaire is available online and also distributed in paper versions during various events whose target audience are potential PL-Grid users. Examples of such events are conferences dedicated to Grids and HPC technologies, HPC users meetings or workshops dedicated to scientific simulations software. In this paper we would like to shortly present the current results of the "PL-Grid Survey", but staying focused on only two vital questions being asked in the questionnaire: "The experiments carried by you consist of..." and "What do you expect from cooperation with HPC center?". The results with the possible answers are presented in Tables 1 and 2. Multiple choices were permitted.

Table 1. Answers to the question "The experiments carried by you consist of..."

Answer	%
Running single program on one node	49.03
Running single program on many nodes	43.87
Running single program simultaneously with multiple input data (Parameter Sweep)	73.55
Running workflow applications	25.16

[1] http://www.plgrid.pl/ankieta

Table 2. Answers to the question "What do you expect from cooperation with HPC center ?"

Answer	%
Shortening the computation time	83.23
Possibility of solving larger problem instances	65.16
Security of data and computation	40.00
Support with developing custom applications	16.77
Support with parallelization of existing application (either self-developed or open-source)	25.16
Creation of more convenient interfaces to existing applications (e.g. portals)	20.00
Frequent training events	35.48

The summarized answers to the first question show that almost half (43.87%) of the surveyed users run parallel applications. The answers to the second question state that the two most desired benefits of using HPC infrastructure are "shortening the computation time" and "possibility of solving larger problem instances". As contemporary trends in HPC hardware are to increase the number of CPU cores rather than increasing the speed of single computation units, this may lead to a conclusion that these objectives can be achieved only by parallelization of existing single-threaded applications.

The presented survey results seem to confirm our predictions that formed the basis of the QosCosGrid project: The parallel applications are becoming more common, as in many cases they are the only way to achieve better performance or solve larger problem instances. Thus, the QosCosGrid systems focused on exposing different parallel environments in an easy and consistent way seem to follow the current needs of the users of HPC centers. In addition, the high ratio of parameter sweep applications (73.55% of queried users have stated that they run this kind of applications) has found its reflection in built-in support for parameter studies in the QosCosGrid system.

III. MAIN COMPONENTS OF THE QOSCOSGRID e-INFRASTRUCTURE

III.1. Architecture Overview

The QosCosGrid middleware consists of two logical levels: grid level and administrative domain (AD) level. Grid-level services control, schedule and generally supervise the execution of tasks, which are spread between independent administrative domains. The administrative

domain (AD) represents a single organization (e.g. an HPC center or research lab) participating in a virtual organization (VO) and sharing resources. Each organization contributes its resources for the benefit of the entire VO, while controlling its own administrative domain and own resource allocation/sharing policies. The organizations agree to connect their resource pools exposed by AD-level services to a trusted "grid level" middleware which tries to achieve optimal resource utilization and ensure the requested level of Quality-of-Service. The key component of every AD is the SMOA Computing service, which gives remote access to queuing systems resources and features including the advance reservations, parallel execution environments – OpenMPI and ProActive with coordinators responsible for synchronization of cross-clusters executions and Data Transfer services for managing data. AD-level services, in turn, are connected to the Grid-level ones. Two of the most important services on this level are the Grid Resource Management System [GRMS], which is a grid meta-scheduling framework controlling executions of tasks, and the Grid Authorization Service [GAS] that offers dynamic, fine-grained access control and enforcement for shared computing services and resources. From the perspective of the architecture, the GAS can also be treated as a trusted single logical point for defining security policies.

The general QosCosGrid architecture is depicted in Fig. 1, and all components except GAS are described in detail in the next sections.

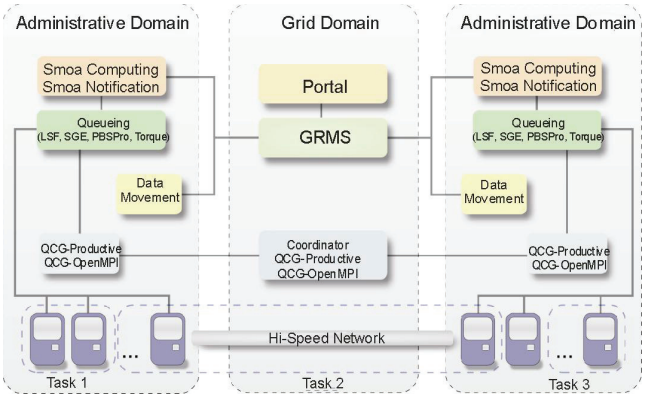

Fig. 1. QosCosGrid architecture

III. 2. Parallel Environments

On the basis of the QosCosGrid system lay two parallel programming and execution environments: OpenMPI (targeted at C and FORTRAN applications developers) and ProActive (targeted at Java applications developers, which is currently an exceptional feature among the other middlewares). Below we briefly describe both of them providing additional information which was altered within the QosCosGrid project with respect to their original versions. The adaptations described in the two next sections were needed in order to make cross-cluster execution feasible in firewalled and NAT (Network Address Translation) environments. It is worth noting that despite the fact that cross-cluster executions in QosCosGrid system are only possible with the two enhanced parallel environments, the single-cluster runs are still feasible with any implementation of the MPI standard.

III.2.1. QCG OpenMPI

The MPI (Message Passing Interface) is de facto a standard in the domain of parallel scientific applications which demands computational resources that are beyond what a single machine can provide. It delivers end users both the programming interface consisting of simple communication primitives and the environment for spawning and monitoring MPI processes. A variety of implementations of the MPI standard are available (both as commercial and open source). In QosCosGrid, it was decided to use OpenMPI [3] implementation of the MPI 2.0 standard as input for further enhancements. The inter-cluster communication techniques that deal with firewalls and Network Address Translation were of key importance. In addition, the mechanism for spawning new processes in OpenMPI needed to be integrated with the QosCosGrid-developed middleware. The extended version of the OpenMPI framework was named QCG-OMPI (where QCG stands for QosCosGrid).

III.2.2. QCG ProActive

The existence of Java based applications in the project's use cases portfolio implied a search for a framework which could provide similar functionality for Java to what MPI offer to the C and FORTRAN written ones. Instead of exploiting existing Java bridges to MPI implementations, a decision was taken to use the ProActive Parallel suite [4]. The library 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 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 (described in the following sections) 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.

III.2.3. OMPI and Proactive Coordinators

In the QosCosGrid environment, additional services were required in order to support the spawning of parallel application processes on co-allocated resources. This was caused by the fact that standard deployment methodologies used in OpenMPI and ProActive relayed on either RSH/SSH or specific batch systems functionality, both of which are limited for single cluster runs (e.g. the SSH based deployment methods are problematic if at least one cluster has worker nodes that have private IP addresses). Those services are called coordinators and are implemented as Web Services.

III.2.4. Cross-cluster communication

Taking into account different existing cluster configurations, we generally distinguish:

- a computing cluster with public IP addresses – both the front-end and the worker nodes have public IP addresses. Typically, a firewall is used to restrict the access to internal nodes.
- a computing cluster with private IP addresses – only the front-end machine is accessible from the Internet, all the worker nodes have private IP addresses. Typically, NAT is used to provide out-bound connectivity.

Those two different cluster configuration types influence inter-cluster communication techniques supported in QosCosGrid called port range and proxy respectively.

III.2.4.1. *Port range*

The Port Range technique is a simple approach that makes the particular parallel environment firewall friendly. Most of the existing parallel environments use random ports by default to listen for incoming TCP/IP traffic. This makes cross-domain application execution almost impossible as most system administrators often forbid to open all inbound ports to the Internet due to security reasons. By forcing the parallel environments to use only predefined, unprivileged range of ports, it is much easier for administrators to configure the firewall in a way to allow incoming MPI and ProActive traffic without exposing critical system services to the Internet.

The Port Range technique is visualized in Fig. 1 below. Each of the site administrators has to choose a range of ports to be used (e.g. [5000-5100] for the parallel communication and configure the firewall appropriately. Please note that the port range technique solves the problem of the cross-cluster connectivity for computing clusters where all worker nodes have public addresses.

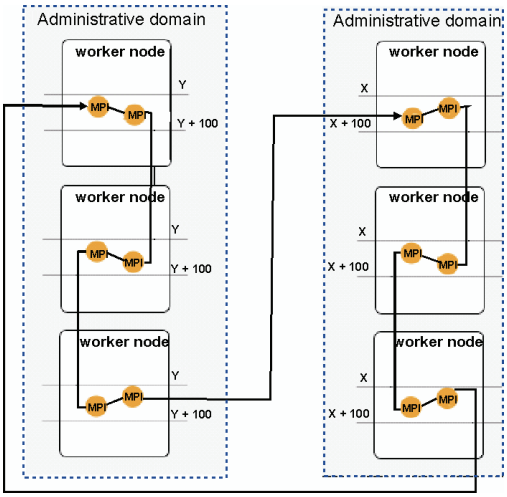

Fig. 2. Operation of the "Port Range" technique

III.2.4.2. *Proxy*

In the second category of the clusters (where worker nodes have IP private addresses), the Port Range technique is not sufficient as all the worker nodes are not addressable from the outside networks. Therefore, in addition to the Port Range technique (which helps to separate traffic), the SOCKS proxy service has to be deployed on front-end machines to route the incoming traffic to the MPI/ProActive processes running at the local worker nodes.

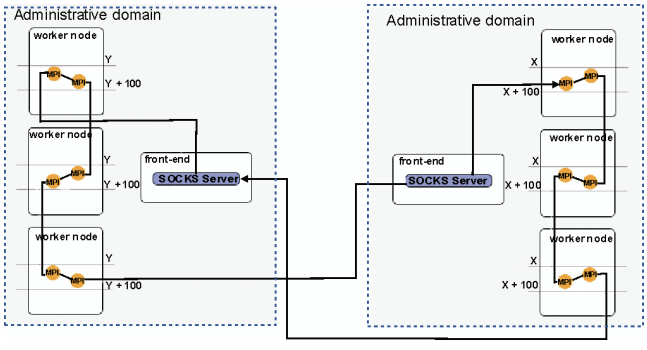

Fig. 3. Operation of the "Proxy" technique

The Proxy operation (used in conjunction with the Port Range technique) is visualized in Fig. 3.

III.3. Topology Awareness

Cross-cluster communication links are usually an order of magnitude slower (in terms of available bandwidth and observed latency) than connections within a cluster. However, many parallel applications, especially those applying the functional decomposition paradigm, have a communication topology that can be described as a set of separate processes groups (so called "communication islands"). The essential feature of such topology is that the communication within groups is much more intensive than the communication between them (as presented in Figure 4 below) so the quality of links joining processes within groups have primary influence on the application performance. In such cases it is possible to avoid degradation of application performance by slower inter-cluster communication links, by allocating processes according to the application topology. The QosCosGrid system addresses this use case by allowing users to describe the requested topology and network parameters which are further taken into account in the scheduling process. In order to fully benefit from such a scheduling system, developers should also make their applications "topology aware".

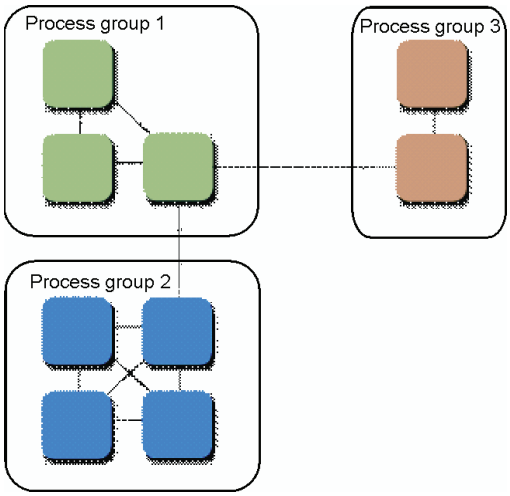

Fig. 4. An example of an application topology consisting of three "communication islands"

Any application wishing to adapt to the runtime topology must receive feedback information from the system that started it. In the QosCosGrid middleware, the topology information is conveyed to the application in two ways, depending on the parallel environment used:

- in QCG-OpenMPI via environment variable which can be used later to build new MPI communicators,
- in QCG-ProActive processes running on one cluster are always grouped in one virtual node.

III.4. SMOA Computing

SMOA Computing (the successor of the OpenDSP[5] project) is an open architecture implementation of the SOAP Web Service for multi-user access and policy-based job control routines by various Distributed Resource Management systems. It uses the Distributed Resource Management Application API (DRMAA)[6] to communicate with the underlying DRM systems. SMOA Computing has been designed and implemented in the way to support different plugins and modules for external communication. Consequently, it can be used and integrated with various authentication, authorization and accounting infrastructures and other external services.

The SMOA Computing service is compliant with the OGF HPC Basic Profile [7] specification (a document which serves as a profile over the JSDL and OGSA® Basic Execution Service OGF standards). In addition, it offers a remote interface for Advance Reservations management, and support for basic file transfer mechanisms.

The service was successfully tested with the following Distributed Resources Management systems:

- Sun Grid Engine (SGE),
- Platform LSF,
- Torque/PBSPro,
- PBS Pro,
- Condor,
- Apple XGrid.

The Advance Reservations capabilities were exposed for SGE, LSF and Maui (a scheduler that is typically used in conjunction with Torque) systems.

III.5. SMOA Notification

SMOA Notification is an open source implementation of the family of WS-Notification[8] standards (version 1.3). It supports the topic-based publish/subscribe pattern for the asynchronous message exchange among Web Services and other involved entities. The main architecture of the notification system is based on a highly efficient, extended version of the NotificationBroker managing all items participating in notification events. Today, SMOA Notification offers sophisticated notification capabilities, e.g. notification message filtering, and it was successfully integrated with different communication protocols and various Web Services security mechanisms. The modular architecture of SMOA Notification also provides a great

opportunity for developers to build new extensions and plugins to meet other specific requirements. Within the QosCosGrid project, the SMOA Notification Provider service is used for brokering notification messages about the job state changes. All instances of the SMOA Computing services act as the information producers while the GRMS service as the consumer.

III.6. Data Movement Layer

As many other projects' middlewares, the QosCosGrid one uses the GridFTP which is part of the Globus Toolkit. GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. It is actually an unofficial standard for all data transfers in Grid environments and extends the standard FTP protocol with facilities such as third-party transfer, parallel and striped data transfer, self-tuning capabilities, X509 proxy certificate based security, support for reliable and restartable data transfers. The development of GridFTP is coordinated by the GridFTP Working Group under the hood of the Open Grid Forum community. The GridFTP is used by GRMS to create the sandbox on resource to ensure isolation of applications runs and to stage in and out data files.

III.7. Grid Resource Management System

The Grid Resource Management System (GRMS) is an open source meta-scheduling system which allows developers to build and deploy resource management systems for large scale distributed computing infrastructures. The GRMS, based on dynamic resource selection, mapping and advanced scheduling methodology, combined with feedback control architecture, deals with a dynamic Grid environment and resource management challenges, e.g. load-balancing among clusters, remote job control or file staging support. The main goal of the GRMS is to manage the whole process of remote job submission to various batch queuing systems, clusters or resources. It has been designed as an independent core component for resource management processes which can take advantage of various low-level Core and Grid Services and existing technologies, such as SMOA Computing and Notifications or GridFTP, as well as various Grid middleware services, e.g. Grid Authorization Service, Data Management Service and more. All these services working together to provide a consistent, adaptive and robust Grid middleware layer which fits dynamically to many different distributing computing infrastructures enabling large scale simulations and to ensure the requested the Quality of Services.

One of the main assumptions for GRMS is to perform remote jobs control and management in the way that it satisfies Users (Job Owners) and their applications' requirements as well as constraints and policies imposed by other stakeholders, i.e. resource owners and Grid or Virtual Organization administrators. Simultaneously, Resource Administrators (Resource Owners) have full control over resources on which all jobs and operations will be performed by appropriate GRMS setup and installation. Note that GRMS together with Core Services reduces operational and integration costs for Administrators by enabling Grid deployment across previously incompatible cluster and resources.

The heart of GRMS is the Meta Scheduling framework which is responsible for scheduling tasks in the environment controlled by it. GRMS has been successfully integrated with the Scheduling Framework which have been designed, implemented and used in the Grid Scheduling SIMulator [9], what allows the GRMS administrator to change the scheduling policies in a very easy and flexible way, using different scheduling plug-ins. The scheduling framework allows developers to easily implement specific algorithms hiding low-level technical details, so that it helps to focus on the scheduling problem and algorithm itself. Because GSSIM and GRMS share the same scheduling interfaces, it is possible to test new scheduling plug-ins in a simulated environment before they are used in production.

One of the most interesting features of GRMS is its ability to deal with jobs defined as a set of tasks with precedence relationships (workflows). The Workflow model used in GRMS is based on direct acyclic graphs (DAG). In this approach a user specifies task's precedence constraints in the form of task's states relationships. With just one call a user can submit the whole computational experiment that consists of many independent application executions. Two ways of expressing the dependencies between tasks are possible. The first one is a direct way, based on the parent-child relationship among tasks. In this case the execution of a child depends on the status change of its parents. The second way of expressing dependencies is associated with the data flow between the tasks. Here a user can specify that an output of one task becomes the input for the other one. What is the beauty here is that a user does not have to specify the exact file locations. However, in such a case it is user's responsibility to define file dependencies correctly. GRMS will reject execution of the job where data dependencies contradict the parent-child relationship. As it was already mentioned, the basic way of introducing dependencies between tasks is by defining the parent-child dependencies. A very interesting and novel

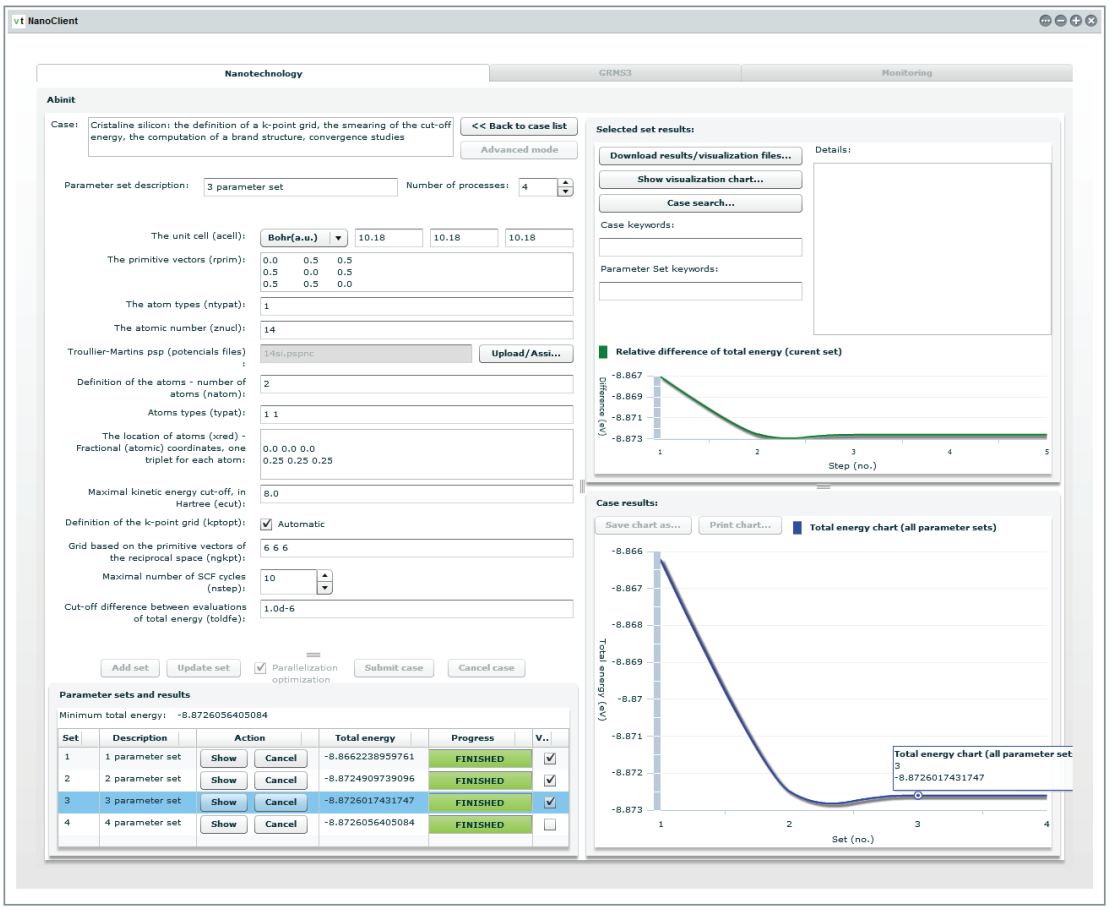

Fig. 5. Gate for NanoSciences with page for Abinit application

feature of GRMS, which distinguishes it from the other systems, is that the GRMS workflow description is more general and flexible than a typical DAG representation and allows in precedence constraints use of not only finished tasks but also of other states like RUNNING, FAILED, SUBMITED. This feature is very useful in many scenarios. For instance, we can imagine that a user would like to an execute an application as soon as the other one starts running, e.g. for client server communication. The other example could be the flow of computation that depends on the failure of execution of one of the tasks (failover mechanisms).

To address users' needs, GRMS supports parameter sweep and allows to start in one call many instances of the same task with a different set of arguments. For every task in the collection, the value of one or more of the task parameters may be changed in some preordained fashion creating a so called “parameters space”. The task of the parameter sweep type can be of course part of a bigger workflow experiment, and all parent-child dependencies are automatically converted by the system to take into consideration the whole collection of generated tasks. This is a ery useful feature and it gives the user an easy way to search the parameters space for the concrete set of parameters that meet the defined criterion. What differs GRMS from other middleware dealing with parameter sweep tasks is the support for multi dimensional parameters spaces in which many parameters can be changed to build the aforementioned space of parameters.

GRMS supports multi-cluster topology-aware parallel applications that can be executed on single cluster but also, if it does not stand in contradiction to application requirements, to be dispersed between resources belonging to many administrative domains. In fact, administrative domains are logically and physically separated services providing access to underlying computing resources managed by different local queuing systems. In order to simultaneously create, synchronize and manage co-allocation of computing resources located in different administrative domains advance reservation mechanisms are used. Large scale parallel applications requirements together with complex parallel communication topologies

can be easily expressed in a formal way using the XML-based job definition language called Job Profile. Consequently, applications developers and end-users are able not only to run their experiments in parallel onto many clusters, but also to perform various benchmark-based experiments as alternative topologies are taken into account during meta-scheduling processes in GRMS. Defined topologies may contain definitions of group of MPI or ProActive processes with resource requirements, using resource and network attributes, for the internal and external group-to-group communication. Therefore, various application-specific topologies such as master-slave, all-to-all or ring are supported in the Job Profile language.

III.8. Scientific Gateways

Addressing user requirements to make access to computing resources intuitive and as easy as possible, simultaneously to the QosCosGrid services a portal called QosCosGrid Gateway was developed on the basis of the VineToolkit library[10]. Vine is a modular, extensible Java library that offers developers an easy-to-use, high-level Application Programmer Interface (API) for Grid-enabling applications. Vine can be deployed for use in desktop, Java Web Start, Java Servlet 2.3 and Java Portlet 1.0 environments with ease. Vine supports a large set of middleware and third-party services, so one can focus on his/her applications and not lose focus on the Grid. The Gateway consists of a general part showing and monitoring computing resources characteristics to provide feedback for end-users, developers and administrators in different administrative domains. To gather all presented metrics various monitoring tests are invoked periodically and their results are presented as graphical maps or diagrams in the portal. For example, the results of bi-directional tests of cross-domain QCG OMPI and QCG ProActive applications measuring bandwidth and latency between clusters located in different administrative domains can be visualised. One of the most useful features of the portal are Gantt charts showing local and cross-domain job executions as well as advance reservation and co-allocation of computing resources in time.

The second part of the portal is a set of domain specific web applications developed to support typical experiment cases by automating the creation and submission of simulations, as well as gathering, analyzing and visualizing simulation results. All these steps that previously had to be made manually by the user, can now be made using the QosCosGrid portal. This speeds up obtaining results and hides the complexity of the underlying infrastructure so that the user can now focus on the scientific domain aspects of experiments.

As an interesting example of a domain specific application is the portal called “Gate for NanoSciences” (Fig. 5) developed in cooperation with a team of researchers from the Faculty of Technical Physics, Poznań University of Technology. The portal serves as a graphical user interface alternative for the command-line Abinit [11] application. The portal supports several computation scenarios, including searching for the minimal total energy of cristaline structures with the definition of a k-point grids; the computation of a band structure (a SCF density computation, then a non-SCF band structure calculation); convergence studies with visualization of partial and final results of simulation. In the advanced mode provided for more sophisticated

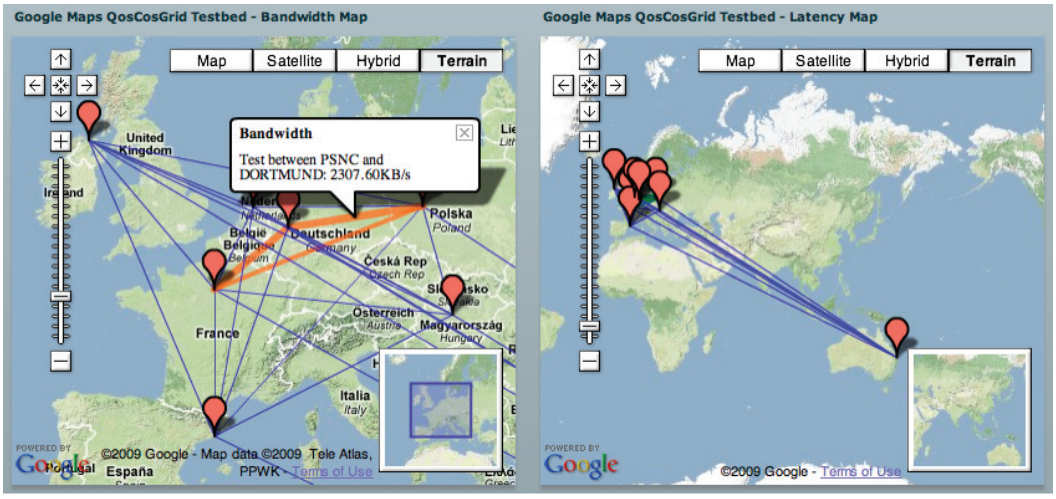

Fig. 6. The geographic map of QosCosGrid testbed together with a productive HPC environment (Dortmund, Germany) successfully connected

users, there is a possibility to submit all Abinit application command-line scenarios, not losing any of the portal-specific enhancements or user-friendly features, like automatic multi series Local Density (DOS) chart generation or persistent results-usecase binding together with possible search and view of similar context cases solved and published in the web by other scientists.

IV. QosCosGrid DEPLOYMENT IN EUROPE

At the end of the QosCosGrid EU project (June 2009) the developed system was deployed on three sites in Europe: Poznań Supercomputing and Networking Center (Poland), INRIA (France) and Dortmund University of Technology (Germany), as it is shown in Figure 6, being a capture of the QosCosGrid Gateway portal. The managed resources had the computing capacity of about 120 CPU cores.

The most recent deployment was related with the "Nano2010" workshop [nano2010] organized by the Faculty of Technical Physics of Poznań University of Technology, where PSNC provided HPC support. While this workshop in short term gave us a feedback on what end-users expects from HPC centers, the long term cooperation with the nanotechnology team from Poznań University of Technology may result in the creation of many useful applications, like the aforementioned Abinit Scientific Gateway.

V. CONCLUSIONS

In this paper, we have presented a comprehensive software stack called QosCosGrid middleware supporting the execution of large-scale simulations on computational grids.

The described QosCosGrid middleware offers an effective multi-user access to job management and co-scheduling features comparing to other existing grid middleware services. It is the first complete grid system with the capability to harness the available grid resources and provide a computationally equivalent to a supercomputer service, allowing to run large scale applications on geographically distributed clusters despite the firewalls existing between them. The transparent integration of the QosCosGrid middleware with the most popular parallel execution environments like OpenMPI and ProActive gives end-users an opportunity to migrate their calculations from single clusters to a grid environment and to take advantage of the power of Grid without any code modification.

The QosCosGrid middleware was successfully deployed in some productive HPC environments (e.g. INRIA, UPF, Dortmund University, PSNC, etc.), extending functionalities of infrastructure services based on UNICORE or gLite. Currently, the QosCosGrid middleware is being enhanced and tested under the national Polish Grid Infrastructure (PL-Grid) project and is planned to be deployed in production environments alongside with gLite and UNICORE middleware stacks.

Further works may be related to tightening the integration of QosCosGrid middleware with PL-Grid e-Infrastructure, e.g. exploiting QosCosGrid co-allocation capabilities in the ARU system – a PL-Grid users and computational grants management application. Moreover, a new Scientific Gateway for the other top scientific applications could be provided.

Acknowledgements

This work has been funded by the EC STREP project QosCosGrid (contract number 033883) and the PL-Grid project: contract number: POIG.02.03.00-00-007/08-00, website: www.plgrid.pl. The PL-Grid project is co-funded by the European Regional Development Fund as part of the Innovative Economy program.

References

[1] Quasi Opportunistic Supercomputing for Complex Systems in Grid Environments. http://www.qoscosgrid.eu/
[2] Polish Infrastructure for Information Science Support in the European Research Space PL-Grid. http://www.plgrid.pl/en
[3] E. Gabriel, G.E. Fagg, G. Bosilca, T. Angskun, J.J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, T.S. Woodal, *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation,* Proceedings, 11th European PVM/MPI Users' Group Meeting.
[4] D. Caromel, C. Delbe, A. di Costanzo, M. Leyton, *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] Open DRMAA Service Provider. http://sourceforge.net/projects/opendsp/
[6] OGF DRMAA Working Group, http://www.drmaa.org/
[7] GFD 114-HPC Basic Profile, Version 1.0. http://www.ogf.org/documents/GFD.114.pdf
[8] OASIS Web Services Notification (WSN) Technical Committee. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn

[9] K. Kurowski, J. Nabrzyski, A. Oleksiak, J. Węglarz, *Grid scheduling simulations with GSSIM*. In Proceedings of the International Conference on Parallel and Distributed Systems, IEEE, 2:1–8, 2007. http://www.computer.org/portal/web/csdl/doi/10.1109/ICPADS.2007.4447835

[10] Vine Toolkit, A better way to use the Grid. http://vinetoolkit.org/

[11] X. Gonzea, B. Amadond, P.M. Anglade et al., *ABINIT: First-principles approach to material and nanosystem properties*. Computer Phys. Commun. 180, 2582-2615 (2009).

KRZYSZTOF KUROWSKI holds the PhD degree in Computer Science and he is leading now Applications Department at Poznań Supercomputing and Networking Center, Poland. He was involved in many EU-funded R&D projects in the areas of Information Technology and Grids over the last few years, including GridLab, inteliGrid, HPC-Europa, or QosCosGrid. He was a research visitor at University of Queensland, University of Wisconsin, University of Southern California, and CCT Louisiana University. His research activities are focused on the modeling of advanced applications, scheduling and resource management in HPC and networked environments. Results of his research efforts have been successfully presented at many international conferences and workshops.

TOMASZ PIONTEK received his M.Sc. in computer science in 1998 from Poznań University of Technology (Parallel and Distributed Computing). 1998-2002 he worked at Poznań University of Technology as a member of the programmers group worked on mobile network protocol analyzers software for Siemens A.G. and Tektronix, Inc. Since 2002 he has been working in Poznań Supercomputing And Networking Center in Application Department and was involved in many EU-funded and national R&D projects in the area of Grids: GridLab, ACGT, QosCosGrid, PL-Grid, MAPPER. His research interests include distributed computing, large-scale simulations and resource management. Since 2010 he leads the Large-scale Simulations Laboratory at PSNC.

PIOTR KOPTA received his M.Sc. degree in Computer Science from the Technical University of Częstochowa in 2002. Currently he is an systems analyst at the Poznań Supercomputing and Networking Center. His research interests concern high performance computing in particulary new computational architectures.

MARIUSZ MAMOŃSKI received his diploma in Computer Science at the Poznań University of Technology (Laboratory of Computing Systems) in 2008. He started working at the Application Department of the Poznań Supercomputing and Networking Center in the 2005. Since then he contributed to several research EU projects, in particulary: GridLab,InteliGrid, BREIN and QosCosGrid, while currently being involved in national and european e-infrastracute projects: PL-Grid and MAPPER. His research primally focus on web services, queueing systems and parallel execution and programing environments. He is an active member of the Open Grid Forum Distributed Resource Management Application API (OGF DRMAA) working group.

BARTOSZ BOSAK received his M.Sc. degree in computer science from Poznań University of Technology in POLAND (Laboratory of IT Systems in Management). Since 2007 he has been working at the Application Department of Poznań Supercomputing and Networking Center as a system analyst and developer. His research interests concern Grids, communication in distributed environments and service integration in SOA. He was participant of the BREIN project (FP6). Currently he works in polish initiative PL-Grid.