2010 International Conference on Complex, Intelligent and Software Intensive Systems

Grid and P2P Middleware for Scientific Computing Systems

Fatos Xhafa
Department of Computer Science and Information Systems
Birkbeck, University of London
London WC1N 3QS UK
EMail: fatos@dcs.bbk.ac.uk

Sabri Pllana
Department of Scientific Computing
Faculty of Computer Science
University of Vienna
Nordbergstrasse 15/C308
1090 Vienna, Austria
EMail: pllana@par.univie.ac.at

Leonard Barolli
Department of Information and Communication Engineering
Fukuoka Institute of Technology
3-30-1 Wajiro-higashi, Higashi-ku
Fukuoka 811-0295, Japan
EMail: barolli@fit.ac.jp

***Abstract*—Grid and P2P systems have achieved a notable success in the domain of scientific and engineering applications, which commonly demand considerable amounts of computational resources. However, Grid and P2P systems remain still difficult to be used by the domain scientists and engineers due to the inherent complexity of the corresponding middleware and the lack of adequate documentation. In this paper we survey recent developments of Grid and P2P middleware in the context of scientific computing systems. The differences on the approaches taken for Grid and P2P middleware as well as the common points of both paradigms are highlighted. In addition, we discuss the corresponding programming models, languages, and applications.**

***Keywords*-Grid, P2P Computing, Parallel Processing, Middleware, Communications Libraries, e-Science Applications.**

INTRODUCTION

Computational Grids and *P2P systems* have emerged as new distributed computing paradigms for the development of large-scale distributed applications. With the fast developments in Internet technologies and with the continuous increase in the connected computational resources, Grid and P2P appeared as the disruptive technologies that can greatly affect not only scientific and academic activity but also business and enterprise productivity. The rationale is that such technologies are *inexpensive* (often built up and maintained in a contributory way), technologically *manageable* and easy to maintain and extend, thus reducing the needs for replacement of existing systems or acquisition of new infrastructures.

The demand for more computing power has been a constant trend in many fields of science, engineering and business. Now more than ever, the need for more and more processing power is raising in the resolution of problems from life sciences, financial services, drug discovery, weather forecasting, massive data processing for e-Commerce and e-Government, etc. Grid and P2P computing are based on the premise to deliver greater computing power at less cost enabling thus the solution of complex problems from many fields of science, engineering and business.

The term Grid computing was introduced in 1990s [9] to express the *"computing power Grid"* in analogy to an electric power Grid. Computational Grids were motivated by the need to develop computational frameworks to support large-scale applications that benefit from the large computing potential offered by such distributed infrastructures.

P2P systems became quite popular for file sharing among Internet users through Napster, Gnutella, FreeNet, BitTorrent and other similar systems. Clearly, the motivation and the community of users and developers behind P2P systems have been different from that of Grid computing. Foster et al. stressed the difference between Grids and P2P systems by stating that *"the sharing that we are concerned with is not primarily file exchange but rather direct access to computers, software, data, and other resources"*

Commonly Grid or P2P middleware provides fundamental services that are at a low-level of abstraction, making thus difficult their use in the process of application development. Moreover, such middleware is not complete with regard to the demands of different application domains, requiring thus some ad-hoc development. Because of the inherent complexity of the Grid and P2P middleware and the lack of adequate documentation, Grid and P2P systems remain still difficult to be used by the domain scientists and engineers.

In this paper we describe and evaluate recent developments in Grid and P2P middleware with respect to scientific and engineering applications. We highlight the differences on the approaches taken for Grid and P2P middleware as well as the common points of both paradigms. In addition, we discuss the corresponding programming models and languages. We also highlight a collection of applications that have been successfully Grid-enabled or P2P-enabled in the context of e-Science.

The remainder of this paper is organized as follows. In Section I we give an overview of the Grid paradigm for parallel processing, and survey the corresponding middleware as well as programming models and languages. The P2P counterpart is presented in Section II. The use of Grid and P2P paradigms for e-Science applications is discussed in Section III. The common characteristics and lessons learned from both paradigms are presented in Section IV. We conclude the paper in Section V.

978-0-7695-3967-6/10 $26.00 © 2010 IEEE
DOI 10.1109/CISIS.2010.109

I. Grid Paradigm

The Grid paradigm may be used in various contexts, such as the *high throughput computing* or *high performance computing*.

High throughput computing. The aim is to complete the largest number of applications per unit of time. It should be noted that in such case, we have a coarse-grained parallelism while the fine-grained is not necessarily exploited. System-related QoS requirements are important in this case.

High performance computing. The aim is to reduce the execution time of applications. To achieve this goal, the exploitation of fine-grained parallelism is necessary. In this case is important the user's requirement on fast completion of a certain application.

A common set of requirements on Grid application-level tools includes: (1) Grid tools should be built on top of the Grid software infrastructure, (2) should isolate users from the dynamics of the Grid infrastructure, (3) should reduce application development cost and complexity, and (4) should be generic and easy to use facilitating thus the application development also to domain scientists or engineers non-familiar with the low-level details of Grid technology. Much of the current research efforts are devoted to bridging the gap between existing Grid middleware and application-level needs.

A. Grid Middleware and Communication Libraries

A major concern in Grid environments is dealing with the high degree of heterogeneity of resources that can range from laptops and PCs to supercomputers. The unified virtual view of Grid systems and the efficient and transparent access to computational resources is achieved by Grid middleware. Grid middleware aims at integrating heterogeneous resources, abstract low-level characteristics, efficient assignment of resources, job/application allocation and execution, monitoring, data access and transfer, and secure access to resources. Grid middleware can be seen as distributed software that enables the communication between applications and underlying computational platforms and as such it should be as complete and generic as possible in terms of operations/services offered to mach any needs of applications to be build on top of it. Due to such vital functionalities, Grid middleware is considered as the backbone of Grid computing systems.

Considerable research efforts are currently devoted to the design and implementation of Grid middleware. Since Grid systems are considered as a natural development step of parallel and distributed systems, existing middleware and communication libraries used in parallel and distributed systems such as Condor, MPI and PVM were extended to support Grid-enabled applications. Certainly, the existing libraries showed several limitations when faced to Grid infrastructures, therefore new middleware and communication libraries are proposed by Grid computing community to cope with new characteristics of such systems and thus match the needs of the development of large scale Grid applications. Below we briefly consider some of them, that are currently widely used by the Grid computing community.

MPI-based middleware: Message Passing Interface (MPI) is established as the main paradigm in the development of parallel scientific applications. Message passing paradigm is more appropriate for tightly-coupled parallel applications rather than loosely-coupled Grid model, however, new versions of MPI appeared attempting to improve the performance of the MPI collective communications for fault tolerant MPI implementations. MPICH-G2 [13] is a Grid-enabled implementation of the MPI, which allows to couple multiple heterogeneous machines to run MPI applications. PACX-MPI (PArallel Computer eXtension to MPI) [4] enables an MPI application to run on a *meta computer* consisting of several, possibly heterogeneous, machines, each of which may itself be massively parallel.

Java RMI-based middleware: Due to its platform independent nature, Java is expected to be of growing importance for the development of Grid middleware. MPJ (Java MPI) [3] is the MPI-like Message Passing for Java, attempting to establish a standard Java parallel programming APIs after Message Passing Interface (MPI). Other efforts use Java's Remote Method Invocation model. RepMI [14] is a compiler-based approach for object replication in Java and is based on RMI. It implements a MagPIe-like broadcast operation for Grid environments.

NetSolve/GridSolve: NetSolve/GridSolve [2], [6] is among the first Grid middleware used for high performance computing that enabled the solution of complex scientific problems using geographically distributed resources. NetSolve intents to use the best computational resources on a network. Fault tolerance and load-balancing are also features of the NetSolve system to achieve high performance. It should be noted however that NetSolve is based on client-server RPC-like model and has thus the limitations of a centralized system in contrast to full-featured Grid systems which are cheaper than a server farm like approaches. It specializes for coarse-grain task parallelism. Another limitation of NetSolve is the lack of universal interface; in fact, its interface protocol can only be applied in the clients of NetSolve system.

Globus and Globus-based middleware: Globus Toolkit [12] is the most widely used middleware for Grid applications. It allows discovering resources, running applications, dealing with heterogeneity and security issues. It is an integrated toolkit of Grid services offering resource allocation and process management, communication services, authentication and security services, system monitoring, remote data access, etc. Due to its wide use, Globus has been integrated by other projects such as Condor-G, MPICH-G2 and NetSolve.

gLite.: gLite[1] is another middleware for Grid computing, which provides a framework for building Grid applications through an integrated set of components that support resource sharing. gLite is developed as part of the Enabling Grids for E-sciencE (EGEE) project. The gLite Grid services follow a Service Oriented Architecture, aiming to facilitate compliance with Grid standards, such as Open Grid Service Architecture (OGSA). One distinguished feature of gLite is that users can implement just services according to their particular needs without having to use the whole system.

B. Programming Languages and Models for Grid Systems

Grid programming languages and models should be able to cope with heterogeneous and dynamic nature of the Grid computing infrastructures.

Programming Languages and Runtime Environments: Due to space limitations we briefly describe recent developments in languages and runtime environments for Grid computing. The reader is referred to Lee et al. [17] and Laforenza [15], for surveys on Grid programming models. One common observation is that up-to-date programming tools and languages are insufficient to support the effective development of Grid applications. In particular, the lack of a simple, *standard* programming interface that hides complexities of Grid systems makes difficult and tedious the programming task in Grid systems.

GEL (Grid Execution Language) [8] is a scripting language for programming parallel applications for a Grid environment. The language is aimed to facilitate coping with high-latency communications and heterogeneity in Grid environments. GEL's semantics provides structures for while loops, conditionals and explicitly parallel execution. *Abacus* [27] is a service-oriented programming language for Grid applications. Abacus offers a service abstraction at language level and hides the user from low-level details such as service deployment are supported by the compiler and the runtime system. Abacus has a similar syntax with Java. *Lightweight Java taskspaces framework* [22] is suited for applications that require inter-task communication. The framework is characterized by decentralization, direct communication between tasks through tuple space distributed over the worker hosts.

Programming Models: Programming models employed for Grid applications are essentially the known ones from parallel and distributed computing such as Shared Memory Model, Threads/OpenMP, Message Passing Model, Object and Service Oriented Models as well as their hybridizations. Yet, due to the intrinsic characteristics of Grid infrastructures, the known models need adaption in order to efficiently exploit the Grid systems.

Classical parallel models such as Master-Worker, Task Farming, Parametric Computation and Divide-and-Conquer Models are commonly used for developing Grid applications. On the other hand, adaption of workflow models has been proposed for Grid applications having dependencies among their tasks in order to exploit coarse-grain, dataflow-like parallelism. Finally, specific models are also proposed to address needs for specific classes of applications. For instance, the IBM's Compute Grid programming models includes two programming models, namely, transactional batch and compute-intensive models within the WebSphere Software. In this case, a compute Grid application is a J2EE application that conforms to the aforementioned Grid programming models.

[1] http://glite.web.cern.ch/glite/

II. P2P Paradigm

While Grid computing originated in the scientific community and was as at the very beginning conceived as a way to gain more processing power for applications, scientists have only recently started to use P2P technologies for large-scale scientific applications. P2P technologies can contribute in several ways to large-scale distributed systems and applications, however, the most important one is collaborative and contributory computing. Harnessing the power of networked PCs, through CPU scavenging and volunteer computing [1], P2P systems are showing their great potential in solving complex problems such as protein folding. It should be noted however that in a P2P system, the performance is usually *"best effort"*-like as compared to high performance Grid. Despite of recent advances, there are still many issues that prevent P2P from being widely used as paradigm for parallel processing.

Scalability. Unlike current Grid systems which tend to be moderate to medium size, P2P systems can actually be very large, virtually joining millions of peers.

Fault-tolerance. Grid systems can be considered as more problematic as regards fault tolerance due to their centralized nature, while P2P systems are in principle more robust. However, it is not yet clear nor implementable in practice how to build robust P2P applications for parallel applications. Decentralization and dynamism are not fully addressed in current P2P systems.

Efficiency. The efficiency of the P2P systems depends much on the contributed bandwidth and other peer resources (storage, processing power, ...). Incentive mechanisms and implementing give-to-get algorithms so that any peer is both a contributor and a beneficiary of the shared resources would increase the systems' efficiency of parallel applications.

Security. P2P systems have all security issues of volunteer computing. Peers might wish to hide personal identifying information, so maintaining anonymity is each time more important to contributing peers. Detecting malicious peers, such as those who can falsify and return incorrect results or distributing malicious executable programs to peers, etc. are important problems in P2P systems.

A. P2P Middleware and Programming Models

As in the case of Grid computing, middleware and communication libraries are vital for the development of P2P-enabled applications. Compared to Grid systems, there have been developed fewer middleware and communication libraries for P2P systems. They are centred around three architectures: client/server, pure P2P (fully decentralized) and hybrid (super-peer) architectures.

JXTA Library: JXTA [18] is a library of generalized P2P protocols that enable the communication between any connected devices in the network. The set of protocols can be used as a basis for the implementation and deployment of P2P networks ubiquitously and independently of computing platforms. Inter-operability among different networks is also claimed (using peerIds, peers can traverse different networks).

A JXTA-based P2P network consists of a set of interconnected peers, which can be self-organized in groups and offer common services to the rest of the network. Services are announced using XML documents, the so called, advertisements. Advertisement is an important piece in a JXTA network; through advertisements peers can join other peers, be aware and use available services in the network. Among others, they include peer advertisement, peer-group advertisement, pipe advertisement, rendezvous advertisement and peer information advertisement. Messages among peers are sent through pipes, asynchronous communication channels. Messages are XML documents encapsulating routing information, credential information (binary data can be also encoded and included).

Peers use several protocols for communication and accomplishing their functionalities. Essentially there are six protocols; it should be noted that a peer needs not to implement all these protocols but only those required for its specification. These protocols are: *peer discovery protocol*, which serve for announcing resources (peers, groups, pipes, services, ...) and discover other peers' resources; *peer information protocol*, which serves for getting state information (time, state, traffic, ...); *peer resolver protocol*, which serves for sending generic requests to one or more peers, usually for information exchange; *pipe binding protocol*, which serves for establishing virtual communication channels (pipes); *endpoint routing protocol*, which serves for finding routing to other peers through relay peers; and, *rendezvous protocol*, which enables peers for subscribing to service propagation.

As can bee seen from the above description, JXTA is a basic library and can be still considered as low level to be used in a straightforward way by programmers for developing P2P applications. Its features such as stable API, NAT and firewall traversal, decentralized architecture and scalability make JXTA appropriate for developing middleware building on top of JXTA.

JXTA-based middleware: Several middleware based on JXTA have been reported in the literature. These include JXTA-Overlay, P3, JXTPIA, Jalapeno, JNGI, JXTA-Grid, OurGrid, Triana and Xeerkat. We briefly describe them next.

JXTA-Overlay[2] [28] project is an effort to use JXTA technology for building an overlay on top of JXTA offering a set of basic primitives that are most commonly needed in P2P applications. It comprises primitives for peer discovery, peer resources discovery, resource allocation, task submission and execution, file/data sharing, discovery and transmission, instant communication, peer group functionalities (rooms etc.) and monitoring of peers, groups, tasks etc. This set of basic functionalities is intended to be as complete as possible regarding the needs of P2P applications, which can be built on top of the overlay. One of the characteristics of the primitives offered by the JXTA-Overlay is their genericity and independence from the applications that will be using them, yet allowing to keep the intrinsic decentralized nature of Grid/P2P systems. Regarding the architecture of the JXTA-Overlay, it is based on a broker-client model, implemented as a broker layer and a client layer, respectively.

Several economic-like models have been implemented for selecting peers in job executions. JXTA-Overlay has been tested through different types of applications. We distinguish here two of them. The first is a distributed application that process large log data files in regularly sequenced data format. The second one is to create a group-ware environment for students of a virtual university. Both applications showed the feasibility of the overlay approach.

P3 [21] (which stands for *Personal Power Plant*) is another middleware using JXTA library aimed at supporting common requirements of P2P software as well as large scale distributed computing. P3 consists of a job management subsystem, a job monitor, and parallel programming libraries. P3 creates a JXTA-supported peer group called a job group for each job that is a submitted by a parallel application. The experimental evaluation of this library as reported in [21] showed a certain amount of overhead on the middleware in terms of communication performance even though a cluster on Gigabit Ethernet LAN was used as infrastructure.

Jalapeno [25] is intended for supporting task execution in a desktop Grid. It consists of manager, worker and task submitter hosts. Jalapeno is appropriate as a framework for solving embarrassingly parallel applications in which the problem is split into smaller independent sub-problems. *JNGI* [26] is another JXTA-based framework for job executions using computational peers in the P2P network. Jobs are split and distributed among several peers. It is claimed that by providing redundancy within peer groups, it is ensured that failures do not affect job completion. *Xeerkat*[3] is a JXTA based framework that uses a worker/hiring analogy to establish computational Grids. It uses an agent computing model where an agent runs a number of available services.

[2] https://jxta-overlay.dev.java.net/

[3] Xeerkat http://code.google.com/p/xeerkat/

XtremWeb: XtremWeb [5] is a Java-based software for the development and deployment of light-weight Grids, namely Grids based on desktop PCs. It is mainly addressed for developing Grid and P2P-enabled scientific applications, especially embarrassingly parallel applications (parametric model). The computing resources are provided in a volunteer-basis. XtremWeb architecture is made up of Client, Coordinator and Workers and uses a pull model in which Workers initiate communications to request jobs from Server. Worker's results are sent to the Coordinator (result collector). In a P2P scenario, XtremWeb can be used to build centralized P2P Systems in which a Worker behaves as a Client. Jobs submitted by Client are registered on the Server and scheduled on Workers. Communications between different parts of XtremWeb include remote procedure call (RPC) messages and data transfers.

P2P-MPI: P2P-MPI [11] is a middleware for developing embarrassingly parallel applications by grouping computing resources of desktop Grids. Typically, a user can request P2P-MPI to transparently find a given number of processors for running a Java application, using the P2P-MPI message passing library, which in turn conforms to MPJ.

III. Grid and P2P-enabled e-Science Applications

Grid computing has shown its usefulness for a family of applications arising in science and engineering. Simulation-driven or experiment-driven applications such as parametric modeling, parameter sweep applications, monte-carlo simulations are examples of successful e-Science applications. Presently, e-Science applications span domains such as chemistry and physics, medical and life sciences, engineering and design, mathematics, economics, business and finance, environmental science, earth sciences and astronomy.

In [20] the authors provide a classification of different approaches for e-Science Grid infrastructures in solving complex problems. The multi-disciplinary nature of e-Science applications as well as the cross-domain collaboration of researchers in the context of virtual laboratories/organizations, make Grid technologies a promising approach for solving complex problems. On the other hand, P2P paradigm has also started to be considered in combinatorial optimization (e.g. for branch&bound algorithms [23], [24] and evolutionary algorithms [16]).

IV. Integrated Approaches to Grid and P2P Systems

As can be seen, Grid and P2P systems have followed different trajectories motivated by different needs and users' interest and supported by rather different developing and research communities. Fortunately, both paradigms are evolving towards sharing more common characteristics. Grid and P2P systems have common intrinsic features of large distributed systems and both of them are concerned with harnessing and sharing of computationally resources.

Grid and P2P systems can benefit from each other characteristics. Grid systems can benefit from P2P techniques of resource discovery, decentralization techniques, replication techniques, and scalability, which are extensively addressed in the P2P literature. In this sense, P2P systems can provide more flexibility to Grid infrastructures. For instance, resource and service discovery approach in Grid systems is centric or hierarchic-based, while P2P systems use advanced presence mechanisms of peer nodes. On the other side, P2P systems can benefit from resource allocation, scheduling, load balancing techniques, security techniques, etc., developed for Grid systems.

Currently there are no P2P-Grid middleware that would support best features of both systems, however, there are attempts to develop integrated approaches for Grid and P2P systems for parallel and distributed computing. OGSA model is one such example, in which Grid, P2P and Web Service concepts are merged. Also, efforts are being done at infrastructure level, such as mixing clusters and desktop machines for P2P-based computational Grids (e.g. [7], [10]) or the PlanetLab platform [19] as a testbed for Grid-P2P applications.

V. Conclusions

Grid computing originated in the scientific community in response to the need for harnessing more processing power for the development of large-scale scientific and engineering applications. P2P paradigm only recently has started to be used for large scale distributed computing systems. Harnessing the power of networked PCs, through CPU scavenging and volunteer computing is enabling P2P systems to leverage high rates of throughput.

In this paper we have described and evaluated recent developments in Grid and P2P middleware with respect to scientific and engineering applications. The analysis of the existing Grid middleware, programming tools and languages shows that they do not fully support the effective programming and development of Grid applications. In particular, the lack of simple, *standard* programming interfaces that hide complexities of Grid systems makes difficult and tedious the programming task in Grid systems. Nonetheless, Grid computing has shown to be more successful for parallel and distributing computing than P2P systems so far. Despite of the recent advances, there are still many issues such as scalability, standardization, efficiency and security that prevent P2P from being fully exploited and widely used as paradigm for efficient parallel processing.

Finally, there is a clear need for integrated frameworks for Grid and P2P, namely a middleware that would support best features of both systems. Models such as OGSA, which merges Grid, P2P and Web Service concepts, are showing their usefulness in this regard.

ACKNOWLEDGEMENTS

Fatos Xhafas research work completed at Birkbeck, University of London, on Leave from Technical University of Catalonia (Barcelona, Spain). His research is supported by a grant from the General Secretariat of Universities of the Ministry of Education, Spain.

REFERENCES

[1] D.P. Anderson and G. Fedak. The Computational and Storage Potential of Volunteer Computing. In *Proceedings of the Sixth IEEE international Symposium on Cluster Computing and the Grid*, IEEE Computer Society, pp. 73-80, 2006.

[2] D. Arnold, S. Agrawal, S. Blackford, J. Dongarra, M. Miller, K. Seymour, K. Sagi, Z. Shi and S. Vadhiyar. Users' Guide to NetSolve V1.4.1, University of Tennessee, *Technical Report*, ICL-UT-02-05, 2002.

[3] M. Baker, B. Carpenter and A. Shafi. MPJ Express: Towards Thread Safe Java HPC. The *IEEE International Conference on Cluster Computing*, pp. 1-10, 2006.

[4] M.A. Brune, G.E. Fagg and M. Resch. Message-Passing Environments for Meta-computing. *Future Generation Computer Systems* **15** (5-6) (1999), pp. 699-712.

[5] F. Cappello, S. Djilali, G. Fedak, T. Herault, F. Magniette, V. Neri and O. Lodygensky. Computing on Large Scale Distributed Systems: XtremWeb Architecture, Programming Models, Security, Tests and Convergence with Grid. *Future Generation Comp. Syst.* **21**(3) (2005), pp. 417-437.

[6] H. Casanova and J. Dongarra. Netsolve: Network enabled solvers. *IEEE Computational Science and Engineering* **5**(3) (1998), pp. 57-67.

[7] A. Chazapis and A. Zissimos. A P2P Replica Management Service for High-Throughput Grids. In *Proceedings of the 2005 international Conference on Parallel Processing*. IEEE CS (2005), pp. 443-451.

[8] Ch. Ching Lian, F. Tang, P. Issac and A. Krishnan. GEL: Grid execution language. *Journal of Parallel and Distributed Computing* **65**(7), pp. 857-869, 2005.

[9] I. Foster and C. Kesselman. *The Grid - Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[10] G. Fox, Sh. Pallickara and X. Rao. Towards Enabling Peer to Peer Grids. *Journal of Concurrency and Computation: Practice & Experience* **17**(7-8) (2008), pp. 1109-1131.

[11] S. Genaud and Ch. Rattanapoka: P2P-MPI: A P2P Framework for Robust Execution of Message Passing Parallel Programs on Grids. *Journal of Grid Computing* **5**(1) (2007), pp. 27-42.

[12] Globus: http://www.globus.org/

[13] N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing* **63**(5) (2003), pp. 551-563.

[14] J. Maassen, Th. Kielmann and H.E. Bal. Parallel Application Experience with Replicated Method Invocation. *Concurrency and Computation: Practice and Experience* **13**(8-9) (2001), pp. 681-712.

[15] D. Laforenza. Grid programming: some indications where we are headed. *Parallel Computing* **28**(12) (2002), pp. 1733-1752.

[16] J.L. Laredo, A.E. Eiben, M. Steen, P.A. Castillo, A.M. Mora, and J.J. Merelo. P2P Evolutionary Algorithms: A Suitable Approach for Tackling Large Instances in Hard Optimization Problems. In *Proceedings of the 14th Euro-Par Conference on Parallel Processing*. LNCS **5168** (2008), 622-631.

[17] C. Lee, S. Matsuoka, D. Talia, A. Sussman, N. Karonis, G. Allen and J. Saltz. A Grid Programming Primer. *Global Grid Forum, Advanced Programming Models Working Group*, 2001.

[18] S. Oaks, B. Traversat, and L. Gong. *JXTA in a Nutshell*. O'Reilly, 2003.

[19] Planet Lab. http://planet-lab.org/.

[20] M. Riedel, A. Streit, F. Wolf, Th. Lippert and D. Kranzlmller. Classification of Different Approaches for e-Science Applications in Next Generation Computing Infrastructures. In *IEEE 4th International Conference on e-Science* (2008), pp. 198-205.

[21] K. Shudo, Y. Tanaka and S. Sekiguchi. P3: P2P-based middleware enabling transfer and aggregation of computational resources. *Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05)* (2005), pp. 259-266.

[22] H. De Sterck, R.S. Markel, T. Phol and U. Rde. A lightweight Java taskspaces framework for scientific computing on computational grids. In *Proceedings of the 2003 ACM Symposium on Applied Computing* (2003), pp. 1024-1030.

[23] S. Tagashira, M. Mito and S. Fujita. Towards generic solver of combinatorial optimization problems with autonomous agents in P2P networks. *Proc. ISHPC 2005* IEICE - Trans. Inf. Syst. E89-D, **6** (2005), pp. 1940-1947.

[24] E-G. Talbi, A. Bendjoudi and N. Melab. A parallel peer to peer branch and bound algorithm for computational Grids. *Proc. of the Intl. Workshop on Peer to Peer, Parallel, Grid and Internet Computing (3PGIC-2007)* (2007), pp. 271-278.

[25] N. Therning and L. Bengtsson Jalapeno: decentralized grid computing using P2P technology. In *Proceedings of the 2nd Conference on Computing Frontiers* (2005), pp. 59-65.

[26] J. Verbeke, N. Nadgir, G. Ruetsch, and I. Sharapov. Framework for P2P distributed computing in a heterogeneous, decentralized environment. LNCS **2536** (2002), pp. 1-12.

[27] X. Wang, L. Xiao, W. Li, H. Yu and Zh. Xu. Abacus: a service-oriented programming language for grid applications. *IEEE International Conference on Services Computing* (2005), pp. 225-232.

[28] F. Xhafa, L. Barolli, Th. Daradoumis, R. Fernandez and S. Caballé, Jxta-Overlay: An interface for efficient peer selection in P2P JXTA-based systems. *Computer Standards & Interfaces*, Available online March 2008