Austrian Grid

Report on the State of the Art Survey

Document Identifier:	**AG-D4-1-2008_2.pdf**
Status:	**Public**
Workpackage:	**4**
Partners:	**Research Institute for Symbolic Computation (RISC)**
Lead Partner:	**RISC**
WP Leaders:	**Wolfgang Schreiner (RISC)**

Delivery Slip

	Name	Partner	Date	Signature
From				
Verified by				
Approved by				

Document Log

Version	Date	Summery of changes	Author
1	27.03.2008	Initial Version	K. Bosa, W. Schreiner
2	30.07.2008	Extended Version	K. Bosa, W. Schreiner

Report on the State of the Art Survey

Karoly Bosa
Wolfgang Schreiner

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University Linz
{Karoly.Bosa, Wolfgang.Schreiner}@risc.uni-linz.ac.at

July 30, 2008

Report on the State of the Art Survey

Károly Bósa Wolfgang Schreiner

July 30, 2008

Abstract

We proposed to participate in the Austrian Grid Phase 2 within the frame of the activity "Grid Research". We intend to deal with development of a distributed programming tool for grid computing which shall empower applications to perform scheduling decisions on their own, utilizing the information about the grid environment in order to adapt the algorithmic structure to the particular situation. Our goal is to design and implement an API that can be used for developing grid-distributed parallel programs without leaving the level of the language in which the core application is written (C/Fortran/Java). This API will be able to eliminate lots of the algorithmic challenges of nowadays grid programming. In this paper, we summarize our experiences concerning the existing programming tools on the area of the multi-cluster and grid environments.

1 Introduction

We proposed to participate in the Austrian Grid Phase 2 [1] within the frame of the activity "Grid Research". We intend to deal with development of a distributed programming API for grid computing. This work shall in particular assist applications whose algorithmic structures do not lend themselves to a decomposition into big sequential components whose only interactions occur at the begin and the end of the execution of a component and that can be scheduled by a meta-level grid workflow language that implements communication between components by file-based mechanisms. Rather the API shall empower applications to perform scheduling decisions on their own, utilizing the information provided by the API about the grid environment at hand in order to adapt the algorithmic structure to the particular situation.

However, no application can execute efficiently in the grid that is not aware of the fact that it does not run in a homogeneous cluster environment with high-bandwidth connectivity between all pairs of nodes but in an environment with heterogeneous nodes and bandwidths that dramatically vary between (at least) three different levels: the processors within a grid node, the grid nodes within the same network, and grid nodes in different networks linked by wide-area connections. Correspondingly, the API shall not hide this fact from the application but reflect the information provided by the grid management and execution environment to the programming language level such that the application can utilize this information and adapt its behavior to it, e.g., by mapping closely interacting activities to nodes within a network and minimizing communication between activities executing on nodes in different networks.

The proposed API shall however hide low-level execution details from the application by providing an abstract execution model that in particular allows to initiate activities and communicate between them independent of their physical location. The execution engine has to map these abstract model features to the appropriate underlying mechanisms: to initiate an activity on a local machine or on a machine within the same administrative authority, simply a process may be started, to initiate an activity on a remote node may mean to contact a corresponding service on that machine, provide the appropriate credentials, and ask the service to start the activity. Likewise, communication with another processor or with another machine in the local network may be performed on the basis of MPI, while communication across the grid may mean the exchange of SOAP messages with web services, or (in the case of large messages) even the shipping of files using GridFTP.

The proposed API thus not at all compete with the "grid workflow" principle or the "service-oriented" approach of grid computing, but simply adds another layer of abstraction to it. The API reflects into the application program the information provided by the available services and utilized by an workflow execution environment, and provides a high-level interface to the underlying mechanisms, making them more amenable to (some classes of) application programs.

In this paper, we summarize our experiences concerning the existing programming tools on the field of the multi-cluster and grid computing. The State of the Art is presented in Section 2. Section 2.4 deals with the topology aware structures of the MPICH-G2 system, which we would like to utilize in our work. Finally, the goals and tasks of the project are outlined in Section 3.

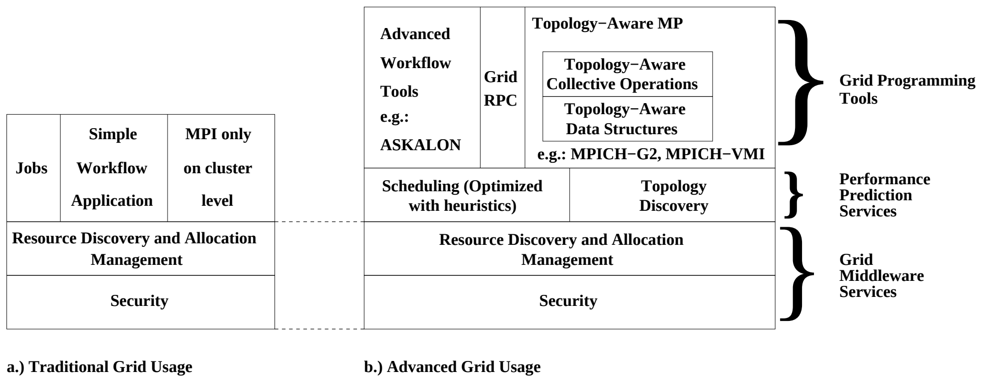
Jobs
Simple Workflow Application
MPI only on cluster level
Resource Discovery and Allocation Management
Security
a.) Traditional Grid Usage
Advanced Workflow Tools e.g.: ASKALON
Grid RPC
Topology−Aware MP
Topology−Aware Collective Operations
Topology−Aware Data Structures
e.g.: MPICH−G2, MPICH−VMI
Scheduling (Optimized with heuristics)
Topology Discovery
Resource Discovery and Allocation Management
Security
b.) Advanced Grid Usage
Grid Programming Tools
Performance Prediction Services
Grid Middleware Services

Figure 1: Traditional and Advanced Programming of the Grid

2 State of the Art

To execute efficiently nowadays advanced grid applications, the fundamental grid middleware services (e.g.: security, resource allocation, etc.) are not sufficient anymore, but some additional grid service layers are required to introduce like *performance prediction* (see Figure 1). The term performance prediction in the context of grid denotes a group of grid services which provide assessment for the performance of various grid resources in advance for a limited period of time. There are many ways how applications can make a profit on this information, e.g.: applying some heuristic models in the scheduling mechanism or optimizing the communication among the grid nodes.

A typical performance prediction service is the *Network Weather Service (NWS)* [34] software system, which became a de facto standard in the grid community as it is used by major Grid middlewares like Globus, to gather qualitative information about the current state of a platform (both network and CPUs) and to predict its short-term performance.

For instance, the *NetSolve* [6] software system which is an implementation of the *Grid Remote Procedure Call (GridRPC)* [25] API of the OGF applies NWS for performance prediction, too.

2.1 Performance Prediction – Scheduling

An example for the applying of performance prediction is the ASKALON project [13, 29, 28] that develops integrated environments under the WSRF framework of Globus Toolkit 4 to support the development and execution cycle of scientific workflows on dynamic grid environments. ASKALON employs its own prediction model based on historical data collected through a well-defined experimental design and training phase. The output of this prediction service is used in scheduling mechanisms enhanced with global optimization heuristics (ASKALON applies some full-graph heuristics such as genetic or *Heterogeneous Earliest Finish Time (HEFT)* [35] algorithms) to find good mappings onto the grid that minimize the execution time.

2.2 Performance Prediction — Topology Discovery in Grid Computing

Unlike classical parallel machines, grids present heterogenous and sometimes even non-dedicated capacities.

Indeed, the efficient use of the Grid resources can only be achieved from a parallel program through the use of accurate network information. Qualitative information such as the network topology is crucial to achieve tasks such as running network-aware applications [22], efficiently placing servers [11], or predicting and optimizing collective communications performance [19].

However, the description of the structure and characteristics of the network interconnecting the different Grid resources is usually not available to users. This is mainly due to security (fear of Deny Of Service attacks) and privacy reasons (hiding bottlenecks). Hence there is a need for tools which automatically construct models of network platforms.

Network discovery tools have received a lot of attention in recent years. However, most of them are not suitable for usage on the grid. Because of it is necessary to rely on tools that only use *application-level measurements*, i.e., measurements that can be done by any application running on a computing grid without any specific privilege. This should comprises the common end-to-end measurements, like bandwidth and latency, but also interference measurements (i.e., whether an interaction between a pair of machines has non-negligible impact on the interaction of another pair of machines, namely they may use the same physical link). For security reasons, these measurements are usually performed using the most basic tools, namely *ping* for latency and *scp* of bandwidths.

Although, *topology discovery* is not part of the performance prediction conventionally, but as was mentioned network topology discovery tools for the grid must rely on only application-level measurements that are typically provided by performance prediction services as NWS. (NWS is able to report end-to-end bandwidth, latency and connection time, which are typical application-level measurements.)

However, the NWS project focuses on quantitative information and does not provide any kind of topological information. This provided information is necessary but not sufficient. With more than two participating sites, simultaneous transmissions may collide with each other on shared links of the wide-area network. Therefore, the predictions of NWS is often too optimistic, because it lacks topology information and thus cannot know which links are shared. This problem is especially important for applications that use collective communication (e.g.: MPI), where many sites communicate with each other simultaneously.

TopoMon [10] is a ping-based tool which extends NWS with additional sensors for the routes between the sites of a grid environment. It provides active topology discovery (in run-time) such that its sensors are implemented as wrapper processes around the locally available *traceroute* [32] programs on the grid sites.

Another tool that attempts to discover automatically the network topology is the *Application-Level Network Mapper (ALNeM)* [21]. It is able to apply the most common application-level network model builder algorithms (e.g.: *spanning tree* related algorithms with minimal latencies, etc.) respectively and its goal is to assess the performance of concurrent transfers (for example to improve collective communications) and not to discover the physical machines interconnection scheme (for administration purposes).

[12] defines a metholodology to compare fairly the various the topology discovery (or network model building) algorithm built in the ALNeM software system and to evaluate their quality. This work shows, that none of the main existing techniques are enable to predict accurately the execution time of simple parallel application running on the actual grid environments.

2.3 Programming Models for the Grid

A grid environment is inherently parallel, distributed, heterogeneous and dynamic, both in terms of involved resources and their performance [15]. Furthermore, grid applications want to use resources and services dynamically and flexibly across that dynamic architecture. While it may be possible to build grid applications using existing programming tools, they are not particularly well-suited for developing and managing flexible compositions or deal with heterogeneous hierarchies of machines, data and network with heterogeneous performance [14].

[20] investigates what properties and capabilities grid programming tools should possess to support not only efficient grid codes, but their effective development. Obviously there is not any tool that addresses all requirements in all situations. The most frequent programming principles applied on the grid are:

Scientific workflows: Scientific workflows emerged as one of the most attractive paradigm for programming grid infrastructures. Roughly, a workflow on the grid is a repeatable pattern of activities, which is enable systematic allocation and organization of grid resources. Workflows are usually represented with directed graphs, where the nodes represent discrete computational components, and the edges represent paths along which data can flow between components. The ASKALON project [13, 29, 28] develops integrated environments under the WSRF framework of Globus Toolkit 4 to support the development and execution cycle of scientific workflows on dynamic grid environments. In ASKALON, the workflows are expressed at a high-level of abstraction using a graphical tool based on the UML modeling standard and advanced constructs such as parallel and sequential loops. This system also contains a scheduling service that uses global optimization heuristics to find good mappings onto the grid that minimize the execution time.

high-performance computing: Grid is also employed for large-scale, high-performance computing. Obtaining high-performance requires a balance of computation and communication among all involved resources. Currently this is done by manually managing computations, communications and data locality using message-passing (e.g.: *MPI*) or remote method invocation (e.g.: *GridRPC*), since they require the programmer to be aware of the arrangement of arguments and their transfers from particular sources to particular destinations.

While MPI addresses some of the challenges in grid computing, it has not addressed them all. Some issues (e.g.,algorithm design, communication patterns) can only be addressed by the MPI application developer. Local- and wide-area networks inject significantly higher latencies and lower bandwidths, and therefore MPI applications that expect to run efficiently in grid environments must be written with respect to this disparity in communication paths.

There are several projects for the realization of MPI libraries for heterogeneous multi-cluster environments: MPICH-G2, Stampi, MetaMPI, PACX-MPI, etc. The most complete and remarkable libraries are:

PACX-MPI [9] is a complete MPI-1 standard implementation extended some routines of the MPI-2 standard. It implements two-level communication approach for multi-cluster systems. It also supports parallel computers with private networks by daemon processes executing on the front-end nodes of cluster for intermachine communication.

MPICH-G2 [3, 18] is a grid-enabled implementation of the MPI-1 standard which based on the MPICH [5] library and which uses grid services provided by the Globus Toolkit pre-Web Service (pre-Web Service) architecture for user authentication, resource allocation, I/O management, process control and monitoring. MPICH-G2 implements topology-aware collective operations that minimize the communication via the slowest channels.

MPICH-G2 describes a topology with a four levels array where each level represents a communication channel: TCP over WAN (level 0), TCP over LAN (level 1), TCP over machine networks (clusters, level 2) and vendor MPI library over high performance network (level 3). MPICH-G2 assigns to every process at each level a non-negative integer named color; processes with same colors can communicate over the corresponding channel.

MPICH-VMI [27] is a grid-enabled MPI implementation. It is also based on MPICH [5] and utilizes the *Virtual Machine Interface (VMI)* [26], which is a middleware communication layer that addresses the issues of availability, usability, and management within heterogeneous wide-area grids. The most important difference between the support of topology aware MPI collectives in MPICH-VMI and in MPICH-G2 that

- MPICH-G2 requires the user to manually provide the physical topology of the network using Resource Specification Language (RSL), while
- MPICH-VMI constructs a limited (2 levels) network topology at runtime using the Grid Cluster Resource Manager (GCRM) which is an external service on the TeraGrid.

2.3.1 Extension Of MPICH-G2

MPICH-GQ [30] The GARA Quality of Service system is integrated with MPICH-G2 in order to reduce communication latency and to optimize the flows of messages.

MPICH-G2/SCore [24] The MPICH-SCore low level high performance communication library for cluster computing is integrated into the MPICH-G2 library in order to optimize the intra-communication within the parallel machines. MPICH-SCore exchanges the vendor MPI in MPICH-G2. The integrated library is called MPICH-G2/SCore.

MPI Globus Forwarder (MGF) [16, 17] extends the topology description provided by the MPICH-G2 with information about existing private networks. MPICH-G2 usage becomes complicated for application developers in presence of clusters where only the front-end node is provided with a public IP address. As a matter of fact MPICH-G2 does not provide any routing mechanism among networks like PACX-MPI; therefore MPI processes started on computing nodes belonging to different private networks are unable to contact each another. This prevents the transparent porting of MPI application to grids where clusters with private networks ares used. MGF uses communication daemons as PACX-MPI. These processes are called forwarders in MGF and executed on the front-end nodes.

MPICH-GX [4] extends MPICH-G2 library with private IP support (similarly to MGF) and checkpointing.

A new version of MPICH-G2 is under development. The name of this version is mpiG, and it uses Globus Web service interfaces rather than pre-Web service interfaces.

2.3.2 Grid Remote Procedure Call (GridRPC)

GridRPC [25] is a remote procedure call API for grid Computing based on the Client-Server model. The concept was proposed for providing a standardized parallel programming interface for the grid. GridRPC specification is the first OGF document that achieved *"Grid Recommendation"* status.

There are necessary features of GridRPC systems such as dynamic resource discovery, dynamic load balancing, fault tolerance, security (multi-site authentication, delegation of authentication, adapting to multiple security policies, etc.), easy-to use client/server management, firewall and private address considerations, remote large file and I/O support etc. All these features are essential for executing the GridRPC systems efficiently on the Grid.

However, some systems based on the GridRPC concept have already existed nowadays, but the interoperability among them is still an essential open issue, since each such a implementation employs its own protocol. Web Services, where XML-based standards such as SOAP and WSDL are expected, could yield a solution for this problem. However, it is not clear and evident, whether:

1. XML-based schemas have sufficient expressive power for GridRPC and

2. performance of the communication could be made sufficient.

[31] evaluates the XML (SOAP, WSDL) based communication and states that Web technologies are promising for GridRPC systems. Nevertheless, this work also presents that encoding of various features of GridRPC is not feasible due to limitations of the existing WSDL standard. Therefore, the existing XML-based web protocols must be extended in order to apply them for GridRPC.

Major implementations are Ninf-G [7, 33], NetSolve [6], OnmiRPC [8] and DIET [2]:

Ninf/Ninf-G is a Japanese project developing programming middleware which enables users to access various resources such as hardware, software and scientific data on the grid with an easy-to-use interface. Ninf-G is an open source software which supports development and execution of Grid-enabled applications using GridRPC on distributed computing resources.

The Ninf-G software system is a reference implementation of GridRPC system using some services of Globus Toolkit 2 (pre-Web Service architecture), e.g.: GSI, GRAM, MDS (for resource discovery), GASS and Globus I/O (for effective, secure, platform independent communication). The new version called Ninf-G4 is already compatible the some WSRF services of Globus 4.

Furthermore, Ninf-G comprises some interesting and useful features (e.g. stateful server-side objects) which have not been part of the GridRPC API recommendation yet.

NetSolve/GridSolve is one of the middleware that implements GridRPC and is developed by Jack Dongarra at University of Tennessee. NetSolve consists of three elements, client, agent and computational server. In NetSolve system, the agent is the most important element because of its role. The roles of the agent are resource discovery and allocation, maintaining of load balance and fault tolerance, and so on. This enables a client to perform the request of GridRPC by knowing only the place of the agent. NetSolve uses the Network Weather Service and Heart Beat Monitor of the Globus Project.

OmniRPC is thread-safe GridRPC system, which inherited from Ninf Project and which contains support for clusters with private IP networks. It uses OpenMP (Open Multi-Processing) for programming multi-threaded GridRPC clients and remote executables (even in a single processor by POSIX threads). However its API still does not contain tools for communicating between the threads.

2.4 Topology-Aware Structures in MPICH-G2

A large number of vendor MPI implementations reside the *Abstract Device Interface (ADI)* [23] layer so do most of the topology-aware MPICH implementation, e.g.: MPICH-VMI and MPICH-G2 (globus2 device). ADI is a portable communication layer in MPICH. The ADI calls such as broadcast or barrier are implemented in terms of point to point calls in the underlying *channel interface* layer.

The common features of the existing topology-aware programming tools are the following:

- they either attempt to discover a limited (2 levels) network topology or expect a description of a (max. 4 levels) topology as an input,
- they forward and make available the given topology information on the level of their programming API and
- they optimize the collective communication operations (e.g.: broadcast) with the help of the topology information such that they minimize the usage of the slow communication channels (only in the case of collective operations).

But they are still not able to adapt the point-to-point communication structure of a parallel programs to network topologies such that they achieve a nearly optimal execution time on the grid.

2.4.1 Accessing Topology Information in MPICH-G2

If the network topology is known by the user (or discovered some software tool) another problem is how to implement parallel programs which exploit this knowledge. In MPICH-G2, the user has the accurate information about the topology and predefines this information as input for the software system via Globus RSL scripts.

The topology-aware data structures of MPICH-G2 described in this section works directly only from C/C++. MPICH-G2 communicates over TCP or a vendor-supplied MPI (vMPI). Some of MPI's collective operations are implemented in MPICH-G2 by making a distinction between WAN-TCP, LAN-TCP, and intra-machine TCP. MPI applications could make use of this TCP stratification by creating communicators (e.g., MPI_Comm_split) that cluster processes based on this topology information. MPICH-G2 exports information on system topology to grid applications programs through attributes (associated with every communicator) [18]:

- *MPICHX_TOPOLOGY_DEPTHS* is a vector (length = communicator size) of integers in which the i^{th} element is the *topology depth/network level* of the i^{th}-ranked process in the communicator.

- *MPICHX_TOPOLOGY_COLORS* is a vector (length = communicator size) of integer pointers in which the i^{th} element is, in turn, a pointer to a vector of integers (length = MPICHX_TOPOLOGY_DEPTHS[i]) and MPICHX_TOPOLOGY_COLORS[i][j] is the color of the i^{th}-ranked process at level j (note that those processes that cannot communicate over vMPI have a topology-depth=3, and therefore, do not have a color defined at MPICHX_TOPOLOGY_COLORS[i][3]).

After the processes have started on the grid, MPICH-G2 uses information specified in the *Resource Specification Language (RSL)* script to create *multilevel clustering* of the processes based on the underlying network topology. Processes that communicate using only TCP are assigned topology depths of 3 (to distinguish between wide area, local area and intramachine TCP messaging), and processes that can also communicate using vMPI have a topology depth of 4. Using these topology depths MPICH-G2 groups processes at a particular level through the assignment of *colors*. Two processes are assigned the same color at a particular level if they can communicate with each other at that network level.

Using these ordered lists of multi-method communication we can ask the question *"Can process A communicate with process B at multi-protocol level i?"*. For example, any two processes can communicate with each other at WAN-TCP. Processes can communicate at LAN-TCP if and only if they are in the same LAN cluster, they can communicate at intramachine TCP if and only if they are in the same RSL subjob, and they can communicate at vMPI if and only if they are in the same RSL subjob and that subjob specifies (jobtype=mpi).

3 Goals and Plans

Our goal is to design and implement, on the basis of the grid execution engines and management mechanisms of Globus (as developed in the Austrian Grid Phase 1 by the groups Fahringer and Benkner) and of the existing MPICH-G2 framework an API that can be used for developing grid-distributed parallel programs without leaving the level of the language in which the core application is written (C/Fortran/Java).

This API should be able to eliminate lots of the algorithmic challenges of grid programming, such that:

- First of all, API will contain built-in scenarios for different distributed programming situations. These libraries will be based on the topology aware information provided by MPICH-G2 and will take care of the arrangement of the processes/jobs among the allocated resources in order to minimize the communication latencies and to achieve an optimal performance for some kind of grid applications on a geographically distributed and heterogeneous grid architecture.
- Then, we plan to extend the previously mentioned scenarios with resource allocation strategies to determine the optimal pool of the available resources for each an application on a dynamic grid environment.
- Finally, we intend to add some services to our libraries which support SOAP-based communication on the highest topology level (via WAN) on the one hand and which also allows to use GridFTP service (for supporting big data transfer) among the nodes on the other hand.

Acknowledgement

The work described in this paper is partially supported by the Austrian Grid Project [1], funded by the Austrian BMBWK (Federal Ministry for Education, Science and Culture) under contract GZ 4003/2-VI/4c/2004.

References

[1] Austrian Grid Project Home Page. http://www.austriangrid.at.

[2] DIET Project Home Page. http://graal.ens-lyon.fr/DIET/.

[3] MPICH-G2 Project Home Page. http://www.hpclab.niu.edu/mpi/.

[4] MPICH-GX Project Home Page. http://www.moredream.org /mpich.htm.

[5] MPICH Project Home Page. http://www-unix.mcs.anl.gov/mpi /mpich1/.

[6] NetSolve/GridSolve Project Home Page. http://icl.cs.utk.edu/netsolve.

[7] Ninf/Ninf-G Project Home Page. http://ninf.apgrid.org/.

[8] OmniRPC Project Home Page. http://www.omni.hpcc.jp/OmniRPC/.

[9] PACX-MPI Project Home Page. http://www.hlrs.de/organization/amt /projects/pacx-mpi/.

[10] M. Burger and ... Topomon: A monitoring tool for grid network topology.

[11] Pushpinder Kaur Chouhan, Holly Dail, Eddy Caron, and Frédéric Vivien. Automatic middleware deployment planning on clusters. *Int. J. High Perform. Comput. Appl.*, 20(4):517–530, 2006.

[12] Lionel Eyraud-Dubois, Arnaud Legrand, Martin Quinson, and Frdric Vivien. A first step towards automatically building network representations. In *Proceedings of Euro-Par 2007*, volume 4641 of *LNCS*, pages 160–169, 2007.

[13] Thomas Fahringer, Alexandru Jugravu, Sabri Pllana, Radu Prodan, Clovis Seragiotto Junior, and Hong-Linh Truong. ASKALON: A Tool Set for Cluster and Grid Computing. *Concurrency and Computation: Practice and Experience*, 17(2-4), 2005. http://dps.uibk.ac.at/askalon/.

[14] I. Foster and C. Kesselmann. *Computational Grids*, chapter 2, pages 15–51. In [15], 1999.

[15] I. Foster and C. Kesselmann, editors. *Grid: Blueprint for a New Computing Infrastructure*. Morgan-Kaufman, 1999.

[16] F. Gregoretti, G. Laccetti, A. Murli, G. Oliva, and U. Scafuri. MGF: A Grid-Enabled MPI Library with a Delegation Mechanism to Improve Collective Operations. In *Proceedings of the 12th European PVM/MPI Users' Group Meeting (Euro PVM/MPI 2005)*. Lecture Notes in Computer Science, Springer, 2005.

[17] F. Gregoretti, G. Laccetti, A. Murli, G. Oliva, and U. Scafuri. MGF: A Grid-Enabled MPI Library. *Journal of Future Generation Computer Systems*, 24(2):158–165, February 2008.

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

[19] Thilo Kielmann, Rutger F. H. Hofman, Henri E. Bal, Aske Plaat, and Raoul A. F. Bhoedjang. MagPIe: MPI's collective communication operations for clustered wide area systems. *ACM SIGPLAN Notices*, 34(8):131–140, 1999.

[20] 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, August 2001.

[21] Arnaud Legrand, Frdric Mazoit, and Martin Quinson. An application-level network mapper. Technical Report 2002-09, Ecole Normale Superieure de Lyon, February 2002.

[22] Arnaud Legrand, Hélène Renard, Yves Robert, and Frédéric Vivien. Mapping and load-balancing iterative computations. *IEEE Trans. Parallel Distrib. Syst.*, 15(6):546–558, 2004.

[23] E. Lusk and W. Gropp. The second-generation adi for the mpich implementation of mpi.

[24] Motohiko Matsuda, Tomohiro Kudoh, and Yutaka Ishikawa. Evaluation of MPI Implementations on Grid-connected Clusters Using an Emulated WAN Environment. In *3rd International Symposium on Cluster Computing and the Grid (CCGRID)*, page 10, 2003.

[25] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, and Casanova. A GridRPC Model and API for End-User Applications. GridRPC Working Group of Global Grid Forum, June 2007.

[26] S. Pakin and A. Pant. Vmi 2.0: A dynamically reconfigurable messaging layer for availablility, usability and management. 2002.

[27] Avneesh Pant and Hassan Jafri. Communicating efficiently on cluster based grids with mpich-vmi.

[28] Radu Prodan, Thomas Fahringer, Farrukh Nadeem, and Marek Wieczorek. Real-world workflow support in the askalon grid environment. In *CoreGRID Workshop on Grid Middleware*, Dresden, Germany, June 2007. Springer Verlag.

[29] Jun Qin, Marek Wieczorek, Kassian Plankensteiner, and Thomas Fahringer. Towards a Light-weight Workflow Engine in the ASKALON Grid Environment. In *Proceedings of the CoreGRID Symposium*, Rennes, France, August 2007. Springer-Verlag.

[30] A. Roy, I. Foster, W. Gropp, N. Karonis, V. Sander, and B. Toonen. MPICH-GQ: Quality-of-Service for Message Passing Programs. In *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*, 2000. Conference on High Performance Networking and Computing 2000, Dallas, Texas, United States.

[31] S. Shirasuna, H. Nakada, S. Matsuoka, and S. Sekiguchi. Evaluating Web Services Based Implementations of GridRPC. In *11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11 '02)*, page 237, 2002.

[32] R. Siamwalla and S. Keshav. Discovering internet topology. In *IEEE INFOCOM*, 1999.

[33] Yusuke Tanimura, Tsutomu Ikegami, Hidemoto Nakada, Yoshio Tanaka, and Satoshi Sekiguchi. Implementation of Fault-Tolerant GridRPC Applications. *Journal of Grid Computing*, 4(2):145–157, June 2006.

[34] Rich Wolski, Neil T. Spring, and Jim Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, 1999.

[35] Henan Zhao and Rizos Sakellariou. An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm. In *Proceedings of Euro-Par Conference 2003*, page 189194, 2003.