Experiments in running a scientific MPI application on Grid'5000

Stéphane Genaud[1], Marc Grunberg[2] and Catherine Mongenet[3]

[1,3]LSIIT-ICPS, UMR 7005 CNRS-ULP
Ple API, Boulevard Sébastien Brant,
67412 Illkirch, France
{genaud,mongenet}@icps.u-strasbg.fr

[2]IPGS, UMR 7516 CNRS-ULP
5 rue R. Descartes
67084 Strasbourg
grunberg@eost.u-strasbg.fr

Abstract

Over the last couple of years, several dedicated grid platforms have been set up to test applications and middleware for grids. Among these is Grid'5000, a reconfigurable platform gathering resources at nine remote geographical sites in France. This paper presents one of the eight experiments that have tested software scalability at the scale of a thousand processors (i.e. 500-1000) on this grid testbed. The experiment aims at analyzing the behavior of a geophysical application (a seismic ray tracing in a 3D mesh of the Earth [14]). The application is computationnally intensive but requires an all-to-all communication phase during which processors exchange their results, which has shown to be a real bottleneck on many hardware platforms. We analyze various runs and show that this application scales well up to about 500 processors on such a grid.

1 Introduction

Grid computing [10] aims at taking advantage of the many disparate computers interconnected through networks such as the Internet. The idea is to use these machines as a virtual computer architecture and offer distributed resources (processors, memory, disk storage, or even remote instruments) to solve large-scale applications. Grid computing is therefore becoming a very attractive alternative to parallel machines for many scientific applications.

However, the behavior of applications on grids is difficult to predict because of the heterogeneity of resources. In order to better assess an application's performance on grids, large dedicated grids have been built to serve as scientific instruments, such as DAS-3[1] in the Netherlands, or Grid'5000 [8] in France. The experiments conducted in this paper have been realized on the Grid'5000 platform. It is currently composed of 9 french campus sites gathering about 2600 CPUs (and growing towards 5000) interconnected with the national education and research network Renater.

In this work we analyze the behavior on Grid'5000 of a scientific code, namely a geophysical application which performs a seismic ray tracing in a 3D mesh of the Earth. The application is computationally intensive as millions of seismic rays (extracted from seismicity recorded since 1965) have to be traced. This application exhibits two main phases: an embarrassingly parallel phase in which all rays can be independently computed, followed by an all-to-all communication phase during which processes exchange their results.

We show that this type of application scales well up to about 500 processors on such a grid. We put forward the impact of the network and we show that the network performance has increased by an order of magnitude in the light of experiments conducted 3 years ago.

The paper is structured as follows. Section 2 browses the contexts in which message-passing applications may be deployed today and discusses current issues related to deployment on grids as well as perspectives introduced by experiments on dedicated grids. Section 3 presents the seismology application used in the experiments. The Grid'5000 platform is described in section 4 while section 5 analyzes the various benchmarks conducted on this platform. Finally, concluding remarks and future works are presented in section 6.

2 Parallel message-passing applications on Grids

Many scientific codes are parallel programs that follow the message passing paradigm and most of them use an implementation of the MPI [2] standard. When considering which platform would be best suited to exploit a scientific

1-4244-0910-1/07/$20.00 ©2007 IEEE.

code using MPI, one should consider their respective benefits and constraints.

Dedicated parallel computers or clusters This is the preferred hardware for running MPI programs for two main reasons. First, MPI does not include built-in fault tolerance features and with almost all MPI implementations the failure of one process during the execution leads to the crash of the whole application. Hence, a dedicated reliable execution platform is highly desirable. Moreover, such platforms are generally characterized by the homogeneity of the resources (e.g. processors), and by the quality of the network interconnecting the processors and its I/O performances. These parameters deeply impact the performance of MPI programs since they may involve numerous communications between any pair of processors, and frequent global synchronizations between processors. Of course, the drawback of these systems is the economic cost which results in numerous users sharing the equipment. As a consequence, the processors are often a scarce resource: a user may wait for a long time, for example, if he requests hundreds of processors for a single program. For instance, when we were regularly running the application described in this paper, we never got more than 256 out of the 768 processors of an SGI Origin 3800 in a french national computing center.

Grids Ideally, grids are a means to overcome the abovementioned limitations since they are generally depicted as virtual supercomputers composed of a potentially unlimited number of computers (mostly individual PCs) offering their resources to others. In reality, the exploitation of parallel applications on grids is, in our opinion, still a challenge for regular users due to two main obstacles:

- **Performances**: The first obstacle lies in the inherent heterogeneity of grids, which implies that applications specifically designed for supercomputers may see their performances drop when run on grids. Even if schedulers could eventually select homogeneous processors, the presence of long distance communications incur unavoidable latencies that have to be taken into account. A re-design of the original application using a loosely synchronized algorithm (e.g. [4]) is sometimes the only solution that yields efficient results.

- **Middleware**: Second, the lack of operating system abstraction that existing middleware currently provides to applications induces cumbersome manual configurations, making MPI application deployments prone to failures. Consider for instance mpich-G2 [15], one of the most popular available environments for message-passing. Though the library improves the performances of collective operations in heterogeneous environments, it does not get much support from its underlying middleware (the Globus tool kit): there is no resource scheduler(users have to explicitly say which computers will take part in the computation via a RSL description) and no fault-tolerance facility is provided. Some other projects (e.g. [9], [6]) have addressed fault-tolerance issues but not the problem of automatic resource selection.

Contrary to applications involving independent tasks, where early large-scale experiments on the grid have achieved runs with thousands of processors (see for example the Condor-G installed grid used to solve the NUG30 problem in year 2000 using a total of 2500 CPU's [16]), we know of very few experiments involving message-passing applications deployed at large-scale. Furthermore, the reported large-scale grid experiments generally involve several super-computers rather than numerous individual computers. For instance Allen et al [3] reports the behavior of an application in astrophysics using 1500 processors on four super-computers at SDSC San-Diego and NSCA Urbana-Champaign. Some other similar experiments have been reported with PACX-MPI [12] (for instance experiments over a European and an intercontinental testbed [11]) but they are mostly "proof-of-concept" demonstrations, set up once, and not permanent infrastructures. Studies that aim to evaluate the underlying grid software or hardware depending on the sites and on the number of processors used, report more technical details affecting performance (e.g. [5]) but are generally less spectacular because the number of hosts involved is smaller.

However, given the ever increasing performance of "off-the-shelf" hardware and networks, and provided grid middleware becomes more sophisticated, one can expect grids to become competitive infrastructures in the near future. In the meantime, dedicated grids such as Grid'5000, which use high performance equipments predetermine how MPI applications may behave on grids at large scale.

3 The seismic ray-tracing application

The geophysical application used in this paper consists in building a seismic tomography model for the Earth, in which the seismic wave velocities in the Earth interior are determined according to the geological nature of the different parts of the Earth. The application and its parallelization have been described in [14] and we quickly recall here its main characteristics.

Application Description In order to build such a tomography model, we use the seismic event's information as it is recorded in international databases. These seismic events are captured by the many stations located all around the world. After such an event, the seismogram data are analyzed in order to localize the earthquake hypocenter. Each earthquake is recorded in the databases by its location, the waves arrival times at the different stations and the characteristics - also called *signature* - of the wave front propagation. Hence, each time the front reaches a geological interface (such as the one between the mantle and the core) is can be either transmitted or reflected and its propagation mode may change from compression to shear (or vice-versa). The ray signature records these changes.

A seismic wave is modeled by a set of rays, where each ray represents the wave front propagation from the hypocenter to one station. The application consists in tracing these seismic rays in a regular mesh of the Earth, according to their signature. The ray tracing algorithm is an iterative process that builds the ray path as a set of discretized points in the 3D space defining the Earth interior. The number of points can go from several hundred to several thousands depending on the length and nature of the ray. Notice that since international databases contain several millions of rays, the ray tracing algorithm consists in computing millions of discretization points. A parallel method is therefore necessary to tackle such huge quantities of data. In the experiments presented in this paper we trace 1.17 million rays.

The computed ray information is stored into the 3D mesh, in each of the cells it intersects. This mesh is decomposed into layers from the surface to the center of the Earth, each layer is then decomposed into regular angular sectors (in latitude and longitude) issued at the center of the Earth. Each elementary volume thus obtained defines a cell that can be approximated by a hexahedron. Each cell of the 3D mesh will contain all information related to its intersecting rays, that is at least the number of rays, and for each ray the length of the ray in the cell, the input and output impact points, the input and output incidence angles.

Application Parallelization Due to the large amounts of memory required, the application has been parallelized. The parallel approach we have used consists in replicating the mesh structure on different processors. A master process then decomposes the set of rays to be traced in equal-size blocks (the default block size has been set to $N/10p$i, where N is the number of rays and p the number of processes) and distributes a block to each slave process which then proceeds to the tracing of these rays in its private copy of the mesh, storing the corresponding information in the impacted cells. When a process has traced its current block of rays, it calls the master process for another block. Once all the rays have been traced, the copies of the 3D mesh held by the different processes must be merged together in order to construct the final mesh in which each cell sums up all the information related to all the rays intersecting with its volume.

This merging phase is realized as follows. The 3D mesh is decomposed into disjoint geographic subsets, called submeshes. Each process holds one submesh and has to merge all the information related to the cells of its submesh. This requires an all-to-all communication step since each process has to first send the data it has computed to the appropriate processes (according to the mesh decomposition) and then to receive data computed by others and related to its submesh. This step is obviously expensive as each process has to exchange data with all the other ones. In the experiments presented in this paper, the total amount of data exchanged (called in-transit traffic) is in the order of tens of gigabytes.

The ray tracing algorithm can be decomposed into three main steps : (1) ray tracing and mesh update by each process with blocks of rays successively fetched from the master process, (2) all-to all communications to exchange submesh information between the processes, (3) merging of cell information of the submesh associated with each process.

Since each ray can be traced independently from any other ray, the first step is highly parallel and can be implemented efficiently on a grid. As millions of rays have to be traced, this step can benefit from the many processors available on Grid'5000. Moreover we will show in section 5 that, despite the cost of the all-to-all communication step, it can be done efficiently thanks to the quality of the interconnection network between the different sites of the Grid.

4 The Grid'5000 testbed

Grid'5000 architecture The Grid'5000 testbed is a federation of dedicated computers hosted across 9 campus sites in France, and organized in a VPN (Virtual Private Network) over Renater, the national education and research network. Each site has currently about 100 to 700 processors arranged in one to several clusters at each site. The total number of processors is currently around 2500 and will be funded to grow up to 5000 processors. The testbed is partly heterogeneous concerning the processors since 75% are AMD Opteron (2, 2.2 or 2.4 GHz), and Itanium2, Xeon and G5 makeup the remainder.

Enhancements to Renater's equipments have been brought in with a new version (Renater-4) completed in November 2005. Left part of Figure 1 shows the leased lines (light-colored lines) between sites which all have been upgraded to 2.5 Gbps (Gigabit per second) since Renater-3. The novelty of Renater-4 lies in the introduction of DWDM (Dense Wave Division Multiplexing) equipments. This technology based on optical networks, is a promising candidate for Next-Generation Internet. Thanks to optical cross-connects interconnected by fiber links, all-optical point-to-point connections, often referred to as a *lightpaths* (which uses a single given wavelength usually called a *lambda*), can be established between site network interfaces.

In Renater-4, the DWDM equipments are dedicated to (currently three) specific projects among which is Grid'5000. The black links on Figure 1 (left) represent the "dark fibers" segments that are progressively exploited to set up DWDM links, and the map on the right of Figure 1 shows the current lightpaths. When connected to this infrastructure, a Grid'5000 site is able to see other sites in the same VLAN and benefits from a connection with a throughput of 10 Gbps.

At the time of writing, three sites can benefit from a 10 Gbps VLAN, namely Nancy, Rennes and Nice (Sophia-Antipolis campus), and three others are connected at 1 Gbps (Grenoble, Lille, Toulouse). The remaining sites Orsay (near Paris) and Bordeaux are still using the initial interconnection system based on Ethernet over MPLS (Multi-Protocol Label Switching), which offers in practice 1 Gbps VLANs.

Not that the new network infrastructure hardly improves latencies: the distance between sites (1500 and 2000 kms of fiber length for the more distant sites) yields an incompressible delay due to the speed of light in fiber. Table 1 shows typical latencies observed at the time of experiments. Latencies inside a given site are unsignificant in comparison (inner cluster latency). Rather, we expect an improved throughput and almost no congestion because the VPN only carries data from Grid'5000 users on its WDM links. On the contrary, the leased lines used in Renater-3 mix Grid'5000 proper traffic with an heavy cross traffic.

Mode of operation Grid'5000 has the fantastic capability of *deploying an environment* on almost any of its nodes. It means a user can reserve any node during a time-slot, reformat one disk partition and install a full system of his choice and finally reboot the node with that system.This mode of operation avoids uncertainties related to different OS or software layers encountered in most experimental environments. In our case, we deployed a system image based on a linux 2.6.13 kernel, and LAM [7] ver. 7.1.1 as the MPI implementation. To minimize the hardware influence, we chose to deploy our image on as many homogeneous nodes as possible. On all selected sites, we use bi-Opteron nodes with 2GB RAM, and only CPU frequencies vary (Nancy, Rennes, Nice 2.0 GHz, Toulouse 2.2 GHz, and Orsay 2.4 GHz).

5 Benchmarks on Grid'5000

5.1 Objectives and Metrics

The objective is to understand how the behavior of an MPI application such as the one described in section 3, is influenced by: (a) the number of processors used (from 32 to several hundreds), and (b) the number of geographical sites involved (from one to five, distant from 400 to 1500 kms).

We expected the latencies incurred by long-distance communications to be the main source of load-imbalance and hence speedup limitations. The metrics we chose in order to describe imbalance was the mean time spent by processors in the two main phases of the application, together with the standard deviation to measure dispersion around the mean.

- In the computation phase, a high dispersion may indicate either heterogeneity in the computation power of CPUs (this does not apply with our homogeneous configurations), or latencies in work requests to the master process, leading to idle time for some processors. The average number of rays computed by a processor and its correlated standard deviation are also good indicators of workload imbalance.
- In the all-to-all communication phase, a large dispersion would denote highly variable durations in send and receive operations of submeshes.

Finally, in order to better understand a potential saturation of the network during the all-to-all communication phase, we measure the total amount of data in-transit, i.e. the sum of individual transfers for all processors.

5.2 Results

Table 2 summarizes the experimental results. The first and second columns respectively define the number of sites and the total number of processors involved. Column 3 indicates the distribution of processors at each site. Columns 4, 5 and 6, respectively, report the times for the ray computation phase, the all-to-all communication phase and the total time. The number of computed rays is given in column 7 and the total amount of data in-transit is in column 8.

These results are a real surprise. Figure 2 shows the measured times on the left hand side, and their respective speedups on the right hand side. Note that the strict definition of speedup (ratio of the best sequential time to the time of parallel program with p processors) does not make sense in our case since the sequential execution on the dataset used in these experiments required us to activate intermediate results files writing to bypass memory limitations. The smallest configuration known to achieve this dataset computation without out-of-core mechanism invocation is a quadri-processor Xeon 3.2 Ghz, 8 GB RAM on which it took 9 hours. The speedup is thus evaluated from the times obtained with the 32 processor configuration.

We observe a quasi-linear speedup of the total execution time for various configurations, with up to 458 processors. Such a good speedup would obviously decrease when using

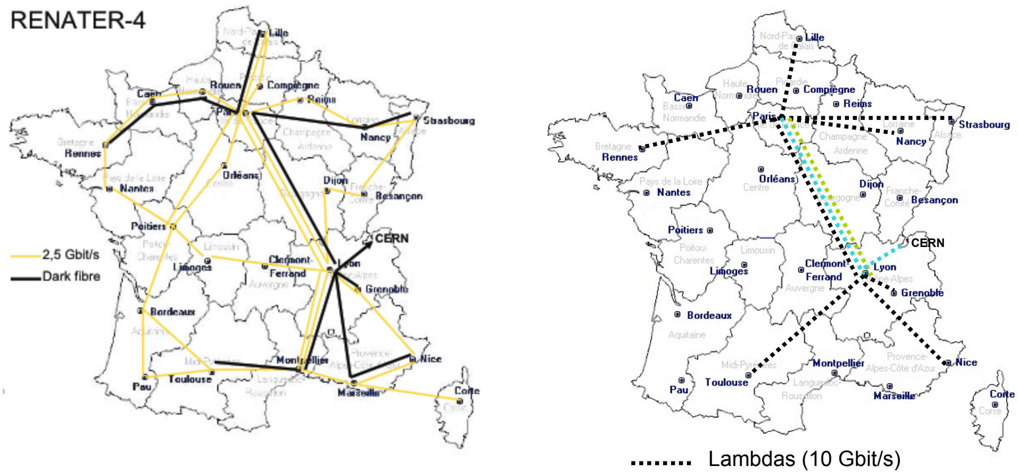

Figure 1. The Renater-4 network underlying Grid'5000.

	nancy	orsay	rennes	toulouse	nice	lyon
nancy	–	5.06/0.03	7.64/0.03	9.74/0.03	11.92/0.02	6.49/0.02
orsay	3.28/0.03	–	3.54/0.03	5.44/0.03	9.07/0.02	2.49/0.02
rennes	7.80/0.02	5.39/0.02	–	6.36/0.01	9.14/0.01	8.17/0.01
toulouse	8.13/0.02	5.34/0.02	4.40/0.01	–	2.85/0.01	2.53/0.01
nice	12.23/0.02	10.95/	9.11/0.01	4.77/0.01	–	5.78/0.01
lyon	6.51/0.02	5.25/0.02	7.93/0.02	4.21/0.01	5.48/0.02	–

Table 1. Typical latencies between Renater-4 sites (delay/jitter in ms).

significantly more processors as the ratio of communication time to ray computation time would increase.

Slave responsiveness A first necessary condition met by the application throughout the tests to reach a linear speedup is load-balance. The figures clearly show that the standard deviation related to durations of computations is small: 3% of mean duration in all configurations, except for 458 processors where it reaches 9%. This is correlated with the average number of rays computed that is nearly the same on all processors. This means that the messages sent by slaves to request work to the master do not suffer idle time, or, in other words, the slave responsiveness is perfect. In a previous study [13] we showed that distant processors on Renater-2 were in a state of starvation, i.e. were not receiving work as quickly as they could compute, even in small configurations with 16 processors. This clearly shows that the network improvement between Renater-2 and Renater-4 impacts drastically this type of application.

Network performance Apparently, the performances do not suffer from long distance communications, even when several distant sites are involved with many processors. When the number of processors increases, the number of messages in the all-to-all communication phase increases quadratically, but the total amount of computed data in the submeshes stays constant as it corresponds to the same set of traced rays. However, each cell has a constant header and as more cells have to be transmitted when the number of processors increases, it results in an increase of the in-transit data, as can be seen in table 2. In addition to the increasing overall volume to communicate, more numerous but smaller messages have to be transmitted. Consequently, more latency overhead is paid on the whole communication phase. For instance, in the test with 192 processors, a processor located in Rennes sends 64 messages of 457 KB on average towards Nancy, 64 other messages towards Toulouse, and 63 to neighbour processors.

In the tests, the all-to-all communication time decreases linearly up to 128 processors, whatever the number of sites (1, 2 or 3). From 128 processors, the communication time nearly stops decreasing to reach a floor of about 30 seconds. Although we expected these 30 seconds to be an inflexion point and to see an increase in the communication time due to the multiplication of messages as well as the

sites	procs	site(procs)	ray comp.	all-to-all	total	nb rays	in transit
1	32	nice*(32)	2651.62/86.79	344.85/61.94	3164.19	35351.60/1265.52	7.29 GB
	62	nancy*(62)	1316.91/44.94	93.13/11.92	1496.71	17965.60/992.46	9.02 GB
	62	nice*(62)	1349.09/45.48	98.78/12.65	1536.36	17965.60/1099.80	8.88 GB
	138	nice*(138)	647.12/21.72	37.39/3.32	729.54	8629.14/547.12	15.47 GB
2	64	nancy*(62) toulouse(2)	1271.91/43.17	90.89/11.45	1445.46	17395.30/972.66	9.26 GB
	128	nancy*(62) toulouse(66)	610.91/20.40	34.68/3.08	688.28	8629.14/573.01	15.29 GB
3	128	rennes(42) nancy*(44) toulouse(42)	620.27/21.21	33.56/2.98	699.57	8629.14/648.83	15.47 GB
	192	rennes(64) nancy*(64) toulouse(64)	412.16/13.70	30.84/2.23	474.65	5737.70/410.90	16.77 GB
5	458	rennes(152) nancy*(32) orsay(184) nice(58) toulouse(32)	177.07/5.69	31.43/1.47	227.53	2398.03/221.91	20.82 GB

Table 2. Experiments results. Columns 4-7 report average/standard deviation values per process. Columns 4-6 are times in seconds. The * symbol indicates the input dataset location.

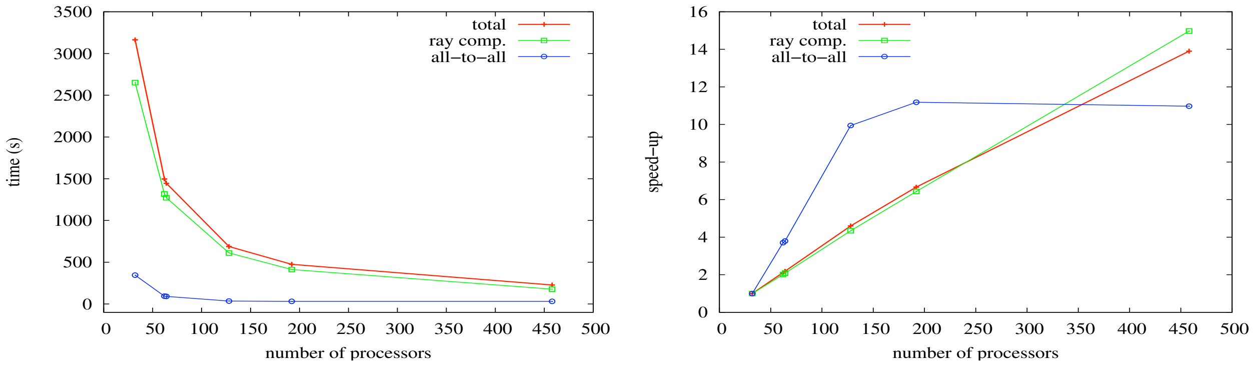

Figure 2. Execution times and speedups for various configurations.

increasing data volume exchanged, it appears to stay nearly stable at 458 processors and 5 sites. Of course, this is the limiting factor for the speedup of this parallel application but given the mass of computations to perform, the communication overhead does not preclude the benefits that can be drawn from using hundreds of processors scattered over distant sites.

6 Conclusion

In this paper, we have reported the behavior of a scientific code in the field of geophysics. The application belongs to a class of parallel applications made up of an embarrassingly computation phase followed by an all-to-all communication phase. We believe that exploitation of parallel applications on grids depends on two main factors: a) the middleware capabilities of transparently handling the execution, namely supporting fault-tolerance, discovering resources, scheduling processes, etc, and b) the hardware system's capabilities and performances, in particular network latencies. In our opinion, no software is mature enough to fulfill the former requirement (middleware factor) and this issue is not in the scope of this paper. We have focused here on the latter requirement by conducting experiments on Grid'5000. Though this platform may not be considered as a true representative of many current grid environments because it is composed of state-of-the-art equipments, it prefigures next generation performances.

We show the application performances on large-scale configurations on this testbed. The conclusion is that this kind of application is today perfectly suited to the modern equipments Grid'5000 offers. The new network Renater-4 equipments (in particular WDM equipments) have latencies next to the physical limit induced by long distances, and enough bandwidth to transfer the gigabytes of results computed in this application, with an interesting ratio of computations to communications up to 458 processors taken on 5 geographical sites. In practice, geophysicists would be interested in running this application with more than 10 millions rays, as opposed to the 1 million used in this experiment. In this case, it would be interesting to check if the application would keep a good speedup when using significantly more processors.

Similar performances evaluations should now address the class of message-passing applications with regular global synchronizations between computing phases. Many scientific codes such as numerical simulations follow this scheme. Until now, they have not shown to perform well on wide area grids because of their sensitivity to communication latencies.

References

[1] http://www.cs.vu.nl/das/.

[2] MPI: A message passing interface standard, version 1.1. Technical report, University of Tennessee, Knoxville, TN, USA, June 1995.

[3] G. Allen, T. Dramlitsch, I. Foster, N. T. Karonis, M. Ripeanu, E. Seidel, and B. Toonen. Supporting efficient execution in heterogeneous distributed computing environment with cactus and globus. In *Proceedings of SuperComputing 2001*, page 52. ACM/IEEE, November 2001.

[4] J. Bahi, S. Contassot-Vivier, and R. Couturier. Evaluation of the asynchronous iterative algorithms in the context of distant heterogeneous clusters. *Parallel Computing*, 31(5):439–461, 2005.

[5] M. Bornemann, R. V. van Nieuwpoort, and T. Kielmann. Mpj/ibis: a flexible and efficient message passing platform for java. In *Euro PVM/MPI 2005*, volume 3666 of *LNCS*, Sept. 2005.

[6] A. Bouteiller, F. Cappello, T. Hérault, G. Krawezik, P. Lemarinier, and F. Magniette. MPIch-V2: a fault tolerant MPI for volatile nodes based on the pessimistic sender based message logging. In *SuperComputing 2003*, Phoenix USA, Nov. 2003.

[7] G. Burns, R. Daoud, and J. Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings of Supercomputing Symposium*, pages 379–386, 1994.

[8] F. Cappello et al. Grid'5000: A large scale, reconfigurable, controlable and monitorable grid platform. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing Grid'2005*, Nov. 2005.

[9] G. Fagg and J. Dongarra. FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world. In *EuroPVM/MPI 2000*, pages 346–353. Springer, 2000.

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

[11] E. Gabriel, R. Keller, P. Lindner, M. S. Mller, and M. Resch. Software development in the grid: The damien tool-set. In *Proceedings of Computational Science, ICCS 2003*, volume 2659 of *LNCS*, pages 235–244, 2003.

[12] E. Gabriel, M. Resch, T. Beisel, and R. Keller. Distributed Computing in an Heterogeneous Computing Environment. In *EuroPVM/MPI*, LNCS, pages 180–187, 1998.

[13] S. Genaud and M. Grunberg. Calcul de rais en tomographie sismique : exploitation sur la grille. *Technique et Science Informatiques*, 24(5):591–608, 2005.

[14] M. Grunberg, S. Genaud, and C. Mongenet. Seismic raytracing and earth mesh modeling on various parallel architectures. *The Journal of Supercomputing*, 29(1):27–44, July 2004.

[15] N. T. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-enabled implementation of the Message Passing Interface. *Journal of Parallel and Distributed Systems*, 63(5):551–563, May 2003.

[16] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The condor experience. *Concurr. Comput. : Pract. Exper.*, 17(2-4):323–356, 2005.