Comparison and tuning of MPI implementation in a grid context

Ludovic Hablot, Olivier Glück, Jean-Christophe Mignot, Stéphane Genaud, Pascale Vicat-Blanc Primet

Comparison and tuning of MPI implementations in a grid context

Ludovic Hablot #[1], Olivier Glück #, Jean-Christophe Mignot #, Stéphane Genaud *, Pascale Vicat-Blanc Primet #

Université de Lyon, INRIA, LIP (Laboratoire de l'Informatique du Parallélisme), * *LSIIT-IPCS, UMR #7005 CNRS-ULP*

[1] ludovic.hablot@ens-lyon.fr

Abstract— **Today, clusters are often interconnected by long distance networks to compose grids and to provide users with a huge number of available ressources. To write parallel applications, developers are generally using the standard communication library MPI, which has been optimized for clusters. However, two main features of grids - long distance networks and technological heterogeneity - raise the question of MPI efficiency in grids.**

This paper presents an evaluation and tuning of four recent MPI implementations (MPICH2, MPICH-Madeleine, OpenMPI and YAMPII) in a research grid: Grid'5000. The comparison is based on the execution of pingpong and NAS Parallel Benchmarks. We show that these implementations present several performance differences. We show that YAMPII performs better results than the others. But we argue that executing MPI applications on a grid can be beneficial if some specific parameters are well tuned. The paper details, for each implementation, the tuning leading the best performances.

I. Introduction

Today, clusters are often interconnected by long distance networks within grids to provide users with a huge number of available ressources.

Users want to execute their applications, written for clusters, on grids to get better performance by simply using more resources. For example, geophysical applications like ray2mesh [1], a seismic ray tracing in a 3D mesh of the Earth, or medical applications like Simri [2], a 3D Magnetic Resonance Imaging (MRI) simulator, have been executed on grids with quite success and but generally with a sublinear speedup. Indeed, MPI, the standard communication library used to write such parallel applications has been initially implemented and optimized for clusters. Most of implementations used in grids do not consider the specificities of grid interconnections, leading to suboptimal performance.

The grid raises mainly three problems to execute MPI applications. First, MPI implementations have to manage efficiently the long-distance between sites. Actually, the high latency between sites is very costly, especially for little messages. Inter-sites communications take more time than intra-sites ones. On an other hand, inter-sites links may offer higher capacities than cluster links.

Secondly, MPI implementations have to manage heterogeneity of the different high speed networks composing the grids. Intra-site communications like for example, through a Myrinet interconnect and an Infiniband cluster on the same site and also inter-sites communications through a WAN (Wide Area Network) have to interoperate.

Finally, the CPUs heterogeneity has an impact on the overall application performance execution. Indeed, it could be of interest to take into account this heterogeneity in the task placement phase. We do not address these two last problems here. In our experiments, all the communications use TCP and homogeneous CPU architectures.

We consider here several MPI implementations taking some grid characteristics into account to improve the execution of MPI applications in these environments. The MPICH2 [3], GridMPI [4], MPICH-Madeleine [5], OpenMPI [6] implementations can be used on a grid but are not similarly optimized to this specific context. The goal of this paper is to identify which are efficient on a real grid and in which conditions. An other question is to define how to configure these implementations to achieve the best performance on the grid. Finally, it could be usefull to have a better view of which communication patterns better fits grid context. To answer these questions, we conduct experiments in Grid'5000 [7] a french research grid platform gathering more than 3000 processors. Nine sites are connected by a dedicated WAN at 1 or 10 Gbps. This architecture provides a fully reconfigurability feature to dynamically deploy and configure any OS on any host.

This paper is organised as follows. Section II presents a state of the art of MPI implementations usable in a grid context. In Section III, we describe the experimental protocol. Section IV presents the results of our experiments and the necessary optimizations required to achieve good performances. Finally, we conclude and present future work.

II. MPI implementations for the grid

This section presents the following implementations that improve either network heterogeneity or long-distance communications: GridMPI, OpenMPI, MPICH-Madeleine. We first describe MPICH2 because it is a largely used implementation. Moreover, a lot of MPI implementations are based on it.

1) MPICH2[3]: This implementation is the successor of MPICH[8], used by a lot of people. It is not a grid implementation but could be used in a grid if all the communications use TCP. It has a four layered architecture which could be quite easily modified. Results with this implementation are well-known that is why we use it as a reference in our tests.

2) MPICH-Madeleine[5]: This implementation is based both on MPICH[8] and Madeleine[9], a communication library which allows communications between several commonly used high speed networks. Thus, MPICH-Madeleine supports efficiently networks heterogeneity between TCP, SCI, VIA, Myrinet MX/GM and Quadrics. MPICH-Madeleine uses a gateway between two clusters interconnected by different networks. For example, if you have a cluster A using Myrinet, a cluster B using SCI and a node shared by the two clusters, messages can be sent between them directly through Madeleine and without using TCP. It can be used in a grid context using TCP for long-distance communications.

3) OpenMPI[6]: OpenMPI is an open source MPI-2 compliant implementation built from scratch combining strong points and resources from earlier projects as well as new concepts.

In the OpenMPI library, progression of communications is handled by the PML (Point-to-point Management Layer) module. This module can use different BTL (Byte Transfer Layer), one for each protocol, hence providing network heterogeneity. Supported protocols are TCP, Myrinet MX/GM, Infiniband OpenIB/mVAPI. Long distance networks characteristics are not particularly taken into account.

The OpenMPI library automatically handles grid specific configurations. For example, clusters portals and cluster specific network fabrics are automatically detected. Nevertheless, specific tunings are still required to achieve optimal performance of long and fast networks of clusters in most cases.

4) GridMPI[4]: This implementation is specifically designed to be executed in a grid environnement. There is two parts in GridMPI: one dedicated to communications, named YAMPII and the other part is the implementation of IMPI (Interoperable MPI) protocol to handle the heterogeneIty. YAMPII is a communication library. IMPI is an overlay over other MPI implementations. Each cluster could use a dedicated VendorMPI to exploit available high speed interconnects and GridMPI is doing the link between them using IMPI. GridMPI has been designed to optimize long-distance communications in MPI applications. Modifications are done at two levels, in the TCP layer and in MPI collective communications level. First, at start time, GridMPI designers propose to modify TCP as follow:

- adding pacing mechanism to avoid burst effects
- reducing RTO (Retransmit Timeout)
- using two different congestion window sizes for collective and point-to-point communications
- bypassing the socket layer

From our knowledge, the public distribution of GridMPI, provides only the pacing mechanism.

Secondly, YAMPII integrates modifications of algorithms for collective operations (see [10]): in a grid context, communications between clusters have a higher bandwidth than intra-cluster once. YAMPII uses a modified MPI_Bcast and MPI_Allreduce functions to take this into account. These improvements have not been released yet.

5) Synthesis: Table I summarizes the main features of each MPI implementation. Two of these implementations can manage network heterogeneity efficiently using gateways (MPICH=Madeleine and OpenMPI). For GridMPI , the management of the heterogeneity is done using IMPI and an appropriate VendorMPI. This may introduce a large overhead in latency. For long-distance communications, the optimizations are done at two levels. First, YAMPII optimize some collective operations to be efficient on the grid. Secondly, it modifies the TCP behavior to adapt it to the high long-distance latency.

We choose MPICH2 as a reference, and YAMPII for its optimizations on long distance communications. Finally, our study is based on these four implementations: MPICH2, YAMPII, MPICH-Madeleine and OpenMPI.

Some works have been done to compare features of MPI implementations. In [11], the authors compare different methods used in MPI implementations (PACX-MPI, MPICH-G2, Stampi and MagPIe) to communicate between a private and a public network. Performance measurement aims at evaluating these different approaches. MPI experiments are done with PACX-MPI. They do not compare the performances of each MPI implementations together.

III. Experimental protocol

A. Goals and Methodology

There are two main goals of our experiments. First, we want to execute different MPI implementations in a real grid context and to see their efficiency on the Grid'5000 grid platform. Then, we want to study the advantages of executing real MPI applications on the grid instead of a cluster.

To compare the different implementations, we run between two sites of the grid a simple MPI pingpong with the default configuration of each implementation. We compare the results obtained to those obtained between two nodes on the same cluster. We have also run a TCP pingpong to evaluate the MPI overhead of each implementation.

Since the MPI pingpong can not represent the behavior of real applications, we also run experiments with the NPB (NAS Parallel Benchmarks [12]).

Pingpong is our own program. One processus sends MPI messages with the MPI_Send primitive to another one which does a MPI_Recv and sends the message back. The size of the messages range from one byte to 64 MBytes. For each size, we obtain the round-trip latency. Then, we can plot the MPI bandwidth available between the two nodes.

The NPB are a set of eight programs (BT, CG, EP, FT, IS, LU, MG and SP) that gives a good panel of the different parallel applications that could be executed on a cluster or a grid. The NPB have been designed to compare performances of MPI implementations and clusters. The NPB use several classes (S, W, A, B, C, D) to represent the size of the problem. We have done our experiments with the B class on four or sixteen nodes. They have different kind of communication schemes. We can see in [13] that all of them are symmetric excepted CG and EP. In [14], the authors give the type of communications (point-to-point or collective) and the number of messages for each NAS for the class A on 16 nodes. We have run each NAS with a modified MPI implementation to find their communication pattern. Table II summarizes

TABLE I

COMPARISON OF MPI IMPLEMENTATION FEATURES

	Long-distance optimizations	**Network heterogeneity management**	**First / Last publication**
MPICH2	None	None	2002 / 2006
GridMPI	Optimization of TCP Optimization of Bcast and All_reduce	Heterogeneity using IMPI above TCP but no support of low latency high-speed networks	2004 / 2006
MPICH-Madeleine	None	Gateway between high-speed networks Support of TCP, SCI, VIA, Myrinet MX/GM, Quadrics	2003 / 2007
OpenMPI	None	Gateway between high-speed networks Support of TCP, Myrinet MX/GM, Infiniband OpenIB/mVAPI	2004 / 2007

TABLE II

NPB FEATURES

	Type of comm.	**Size and number of messages**
EP	P. to P.	192 * 8 B + 68 * 80 B
CG	P. to P.	126479 * 8 B + 86944 * 147 kB
MG	P. to P.	50809 * various sizes from 4 B to 130 kB
LU	P. to P.	1200000 * 960 B<msg<1040 B
SP	P. to P.	57744 * 45 kB<msg<54 kB + 96336 * 100 kB<msg<160 kB
BT	P. to P.	28944 * 26 kB + 48336 * 146 kB<msg<156 kB
IS	Collective	176 * 1 kB + 176 * 30 MB
FT	Collective	320 * 1 B + 352 * 128 kB

the communication features of NAS parallel benchmarks. EP mostly computes and sends very few data. CG communicates with little and big messages. MG sends varing size of messages. LU sends medium size messages and is the most communicating. BT and SP send a lot of big messages. Finally, IS and FT communicate using collective operations with very big messages for IS. FT uses the primitive MPI_Alltoall and IS uses MPI_Allreduce and MPI_Alltoallv.

B. Testbed description

Our experiments are conducted on Grid'5000, which links nine sites in France, having from 5 ms to 21 ms of RTT TCP latency. Each site is composed of an heterogeneous set of nodes and local networks (Infiniband, Myrinet, Ethernet 1 or 10 Gbs). Sites are connected by a dedicated WAN operated by RENATER at 1 or 10 Gbps. This architecture provides researchers with a fully reconfigurability feature to dynamically deploy and configure any OS on any host. This feature allows them to have administrator rights to change TCP parameters for instance.

Figure 1 shows the description of our testbed. We use between 1, 2 or 8 nodes on each cluster (Nancy and Rennes) depending on the experiment (execution of pingpong or NPB). Each node is connected to a switch with a 1 Gbps Ethernet card. Thus, the maximum bandwidth available between one process in Rennes and one process in Nancy is 1 Gbps. The RTT TCP latency is about 11.6 ms.

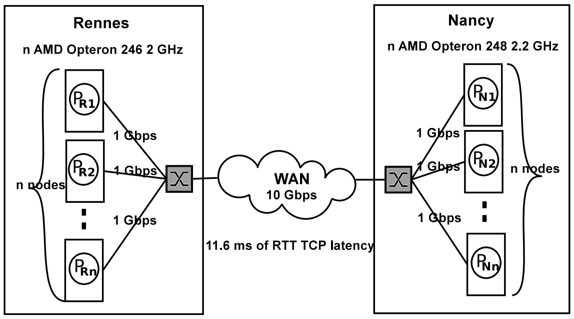

Fig. 1. Configuration used

TABLE III

HOSTS SPECIFICATIONS

	Rennes	**Nancy**
Processor	AMD Opteron 248 2.2 GHz	AMD Opteron 246 2GHz
Motherboard	Sun Fire V20z	HP ProLiant DL145G2
Memory	2 GB	
NIC	1Gbps Eth	
OS	Debian	
Kernel	2.6.18	
TCP version	BIC + Sack	

Table III shows the hardware and software specifications of the nodes.

We have used the following versions of MPI implementations: MPICH2 is the 1.0.5 release using default parameters. GridMPI is 1.1. We do not use the IMPI support as all the communications (intra and inter sites) are achieved by TCP and YAMPII is self-sufficent. The overhead may be important using IMPI. MPICH-Madeleine is the svn version of the 6th of December 2006. It uses thread (`-lib=-lpthread`) and fast buffering `--with-device= ch_mad:--fast-buffer`. Finally, OpenMPI version is 1.1.4.

IV. RESULTS OF EXPERIENCES AND OPTIMIZATIONS

A. First ping-pong results

The experiments within a cluster are done between two nodes of the Rennes cluster corresponding to P$_{R1}$ and P$_{R2}$ on Figure 1 and within the grid between P$_{R1}$ and P$_{N1}$. Pingpong results are done with 200 round-trips with the default configuration of MPI implementations. Then, we take the minimum result (among the 200) for the latency and the maximum for the bandwidth in order to eliminate perturbations due to other Grid'5000 users.

Table IV shows the latency comparison between the different MPI implementations in the Rennes cluster or in the grid (between Rennes and Nancy). The overhead of MPI implementations is almost the same in a cluster or in a grid excepted for MPICH-Madeleine. MPICH-Madeleine has a lower overhead in the grid. These figures confirm the long-distance overhead which corresponds to the network latency (11600 μs of RTT ping corresponding to 5800 μs one way).

TABLE IV
COMPARISON OF LATENCY IN A CLUSTER AND IN A GRID (IN US)

	In the Rennes cluster	**In the grid: btwn Rennes and Nancy**
TCP	41	5812
MPICH2	46 (+5)	5818 (+6)
YAMPII	46 (+5)	5819 (+7)
MPICH-Madeleine	62 (+21)	5826 (+14)
OpenMPI	46 (+5)	5820 (+8)

Figure 2 shows the bandwidth obtained on a cluster with the default parameters. All the implementations reach 940 Mbps that is the maximum goodput of TCP on a 1 Gbps link. Each implementation has a threshold around 128 kB excepted YAMPII. These thresholds are due to the different MPI modes (eager and rendez-vous).

Figure 3 represents the results of the same experiment on the grid. Results are very bad. None of the implementations nor direct TCP implementation of the pingpong reached a higher bandwidth than 120 Mbps. These behaviors are due to both bad TCP and MPI configuration. We describe in the next section how to tune them.

B. Tuning to be done on the grid

In TCP, communications are done through sockets. TCP uses a congestion window to limit the number of 'on the fly' packets that are sent but not acknowledged. The sender does not know that a packet is received until being acknowledged, so it must keep this packet in a buffer to restransmit it if needed. Hence, the TCP socket buffers must be able to contain at least a congestion window. The size of these buffers has to be set at least at the product of RTT * bandwidth [15].

Two linux kernel variables allow to modify the size of these buffers.

- The first value specifies the maximum size that a socket can allocate to its buffer: */proc/sys/net/core/rmem_max* and *wmem_max* have to be increased (see [16]).
- The linux kernel can adapt these values dynamically with a mechanism called auto-tuning. The first and last value of */proc/sys/net/ipv4/tcp_rmem* and */proc/sys/net/ipv4/tcp_wmem* represent the bounds of auto-tuning in the linux kernel. The middle value is the size given initially by the kernel when a socket is created. The last value has to be increased to RTT * bandwidth.

With the first method, we have to modify the code of each implementation to set the initial size of the socket. With the second, the size is set automatically.

In our experiments, between Rennes and Nancy, the socket buffer has to be set to at least at 1.45 MB (RTT=11.6 ms, bandwidth=1 Gbps). However, for compatibility with the rest of the grid, we set it at 4 MB. This tuning is self-sufficient for MPICH2 and MPICH-Madeleine but not for YAMPII and OpenMPI. In YAMPII, the middle value of TCP socket buffer has to be increased. In OpenMPI, the size of the buffers is specified when a socket is created. This size is 128 kB by default. The modification of this size is done dynamically, passing two parameters to mpirun: `-mca btl_tcp_sndbuf 4194304 -mca btl_tcp_rcvbuf 4194304`

Figure 4 shows the new pingpong results after TCP tuning. The maximum MPI bandwidth available is now around 910 Mbps in the grid which is a good result regarding the 940 Mbps in the Rennes cluster. However, the half bandwidth is only reached around 1 MB in the grid against 8 kB in the cluster. This big difference is not surprising and is explained by the fact that the latency in the Rennes cluster is around 40 μs whereas the grid latency is about 5800 μs. TCP and YAMPII have the same behavior. YAMPII clearly reaches a higher bandwidth for messages bigger than 64 kB. We currently investigate why.

C. NAS Parallel Benchmarks

The two goals of the next experiments are to compare the performances of each implementation on the grid (Figure 5) and to see if the grids can well execute MPI applications (Figures 6 and 7).

In the next experiments, we execute the NPB 2.4. We used two different configurations: on the same cluster (using P_{R1} to P_{R16} see 1) or between two clusters joined by a WAN (using P_{R1} to P_{R8} and P_{N1} to P_{N8}) We execute each NPB 5 times and take the better time of these experiments.

Figure 5 shows the NAS execution time comparison of each MPI implementation. MPICH2 is taken as the reference implementation. The performance of the other implementations are relative to MPICH2.

The relative time variation of YAMPII is very important for the applications that communicate with collective operations (FT and IS). This is due to the fact that YAMPII uses better algorithms for them. On MG, the performances are quite similar for each implementation excepted for MPICH2. However, the performances of MPICH2 are better on LU. Finally, execution time of BT and SP is only a little bit better with YAMPII. BT and SP applications timeout when executing with MPICH-Madeleine.

The next experiment aims at evaluating the overhead the grid introduces compared to the execution on a cluster. As Fig. 5 shows that YAMPII has a better behavior for each application, we choose it to compare the NPB on the grid.

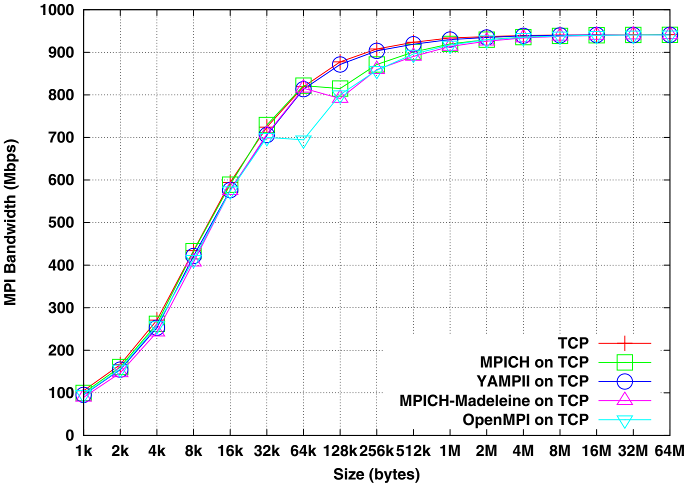

Fig. 2. Comparison of MPI bandwidth of the different MPI implementations on a local network (cluster) with the default parameters

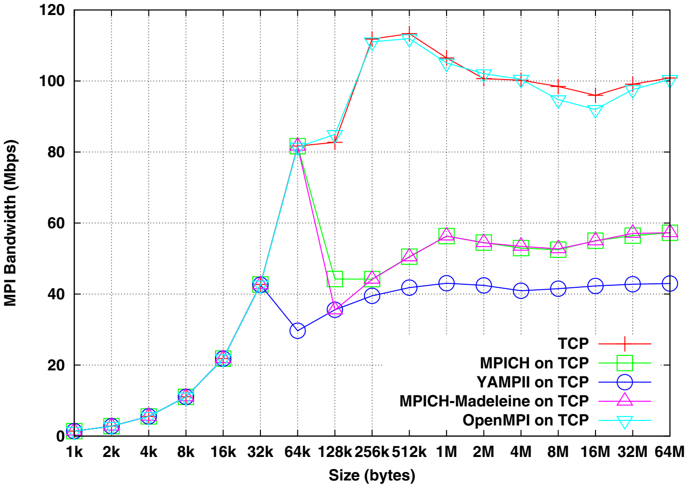

Fig. 3. Comparison of MPI bandwidth obtained with different MPI implementations on a distant network (grid) with the default parameters

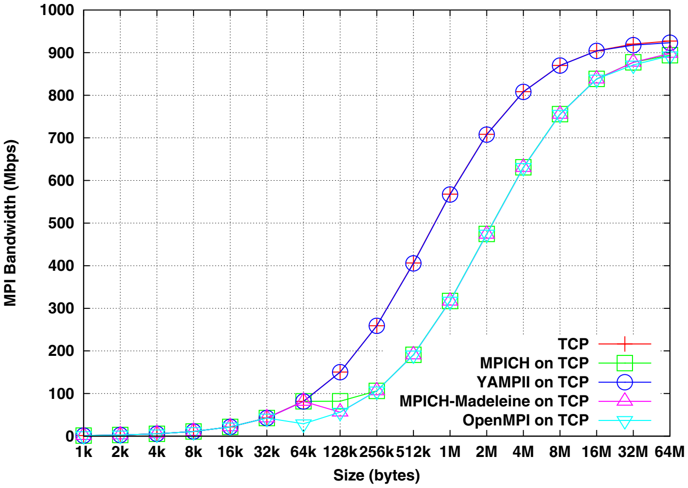

Fig. 4. Comparison of MPI bandwidth of the different MPI implementations on a distant network (grid) after TCP tuning

Figure 6 represents the relative performance of 16 nodes divided in two groups by a WAN to 16 nodes on the same cluster. This figure shows the impact of the latency for each kind of applications. EP has few communications thus the relative performance is close to 1. CG and MG bad performance because they communicate with little messages. The latency has a most important impact on little messages. LU's performance is good because of the message sizes. SP and BT communicate with big messages, thus the high latency does not impact them so much. The performance of IS is quite bad because the IS algorithm is affected by long distance. FT takes advantage of the optimization done on the MPI_Alltoall primitive in YAMPII.

If we compare strictly the grid and the cluster with the same number of nodes, there is always an overhead. This overhead is less than 20% for half of the NAS. However, the main interest of the grids is to find resources that are not available on the cluster that's why Figure 7 represents the relative performance of 4 nodes in the same cluster to 16 nodes on two clusters.

The results for the performances of each application on the grid are quite similar of the last experiment. In this experiment, the speed-up should be equal to four if the grid had no impact. For CG and MG, even if we raise up the computing power four times, there is few speed-up. The long distance is very costly on applications with little messages. LU and BT have a very good speed-up near of 4 on the grid. FT and SP performances are quite good (the speed-up is at least 3).

These results confirm that it is efficient to execute an MPI application on the grid because there is a speed up for each application. However, applications with little messages have very bad performance due to high latency. Collective operations have then to be highly optimized to be efficient.

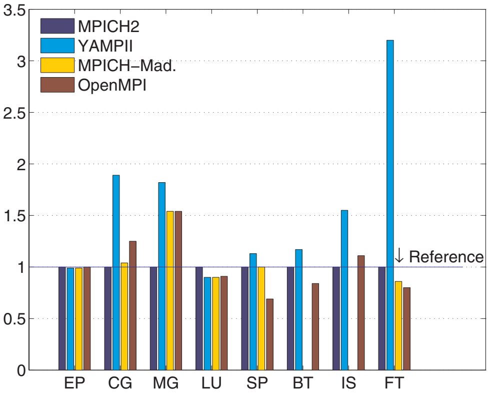

Fig. 5. Comparison of the different MPI implementations: relative (Ref. MPICH2) execution time of NPB on the grid (8-8 nodes on two different clusters)

V. Conclusion and Future Work

In this paper, we have presented a study of different MPI implementations in a grid context. We aimed at finding a good implementation to execute an application in this context. We also wanted to identify which application types and communication patterns are best suited to grids.

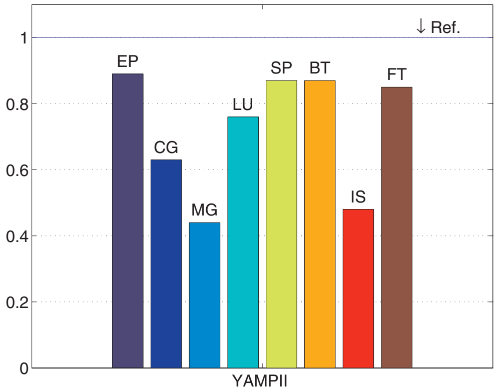

Fig. 6. Comparison of YAMPII on one cluster (16 nodes) and on the grid (8-8 nodes on two different clusters): relative (Ref. is one cluster) execution time of the NPB

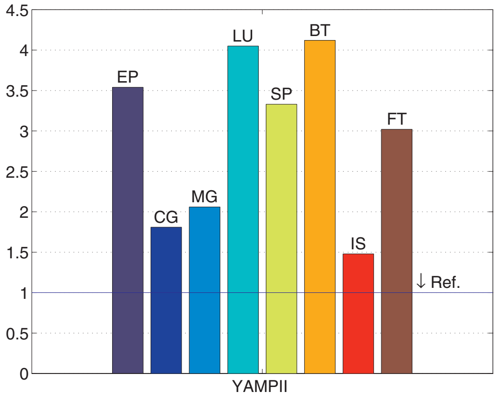

Fig. 7. Comparison of YAMPII on one cluster (4 nodes) and on the grid (8-8 nodes on two different clusters): relative (Ref. is one cluster) execution time of the NPB

Our experiments are based on four MPI implementations (MPICH2, YAMPII, MPICH-Madeleine and OpenMPI). They show that TCP tuning and optimization in MPI implementation are necessary to obtain reasonable performances. After tuning, each MPI implementation can reach as good performance as TCP. To complete our study, we evaluate each implementation with the NPB. Results show that YAMPII behaves better than the others on Grid'5000 due to its optimizations of collective operations and of a simple implementation of its TCP version. However, YAMPII can not manage heterogeneity. Execution of an application on a grid is efficient for applications with quite big messages for point to point communications. Collective operations have to be improved to be efficient.

In a near future, we will pursue our experiments with other Grid-aware implementations like MPICH-G2 and with other real MPI applications. Our results have shown problems with the slowstart and the congestion avoidance mechanisms of TCP when executing an application on the grid. We want to further explore these issues and study optimizations within TCP. Finally, we will test the network heterogeneity management of each implementation with different high performance interconnects. Using these networks for local communications can be efficient to improve performance but has to remain simple. The overhead introduced by the management of heterogeneity has to be less important than the TCP cost.

Acknowledgment

This work has been funded by the French ministry of Education and Research, the ANR HIPCAL project, INRIA GridNet-FJ associated team and the EU IST FP6 EC-GIN Project. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see https://www.grid5000.fr).

References

[1] S. Genaud, M. Grunberg, and C. Mongenet, "Experiments in running a scientific MPI application on Grid'5000," in *4th High Performance Grid Computing International Workshop, IPDPS conference proceedings*. Long beach, USA: IEEE, March 2007.

[2] H. Benoit-Cattin, F. Bellet, J. Montagnat, and C. Odet, "Magnetic Resonance Imaging (MRI) simulation on a grid computing architecture," in *Proceedings of IEEE CGIGRID'03-BIOGRID'03*, Tokyo, 2003.

[3] W. Gropp, "MPICH2: A New Start for MPI Implementations," in *Recent Advances in PVM and MPI: 9th European PVM/MPI Users' Group Meeting*, Linz, Austria, Oct. 2002.

[4] GridMPI Project, http://www.gridmpi.org/gridmpi.jsp.

[5] G. Aumage, Olivier et Mercier, "MPICH/MADIII : a Cluster of Clusters Enabled MPI Implementation," in *Proceedings of CCGrid*, Tokyo, 2003.

[6] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[7] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lantéri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.-G. Talbi, and T. Iréa, "Grid'5000: a large scale and highly reconfigurable experimental grid testbed." *International Journal of High Performance Computing Applications*, vol. 20, no. 4, pp. 481–494, Nov. 2006, https://www.grid5000.fr/.

[8] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "High-performance, portable implementation of the MPI Message Passing Interface Standard," *Parallel Computing*, vol. 22, no. 6, pp. 789–828, 1996.

[9] O. Aumage, "Heterogeneous multi-cluster networking with the Madeleine III communication library," in *16th International Parallel and Distributed Processing Symposium*, Florida, USA, 2002.

[10] M. Matsuda, T. Kudoh, Y. Kodama, R. Takano, and Y. Ishikawa, "Efficient MPI Collective Operations for Clusters in Long-and-Fast Networks," in *Proceedings of Cluster06*, Barcelona, Spain, Sept. 2006.

[11] M. Müller, M. Hess, and E. Gabriel, "Grid enabled MPI solutions for Clusters," in *CCGRID '03: Proceedings of the 3st International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2003, p. 18.

[12] D. Bailey, E. Barscz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga, "The NAS Parallel Benchmarks," NASA Ames Research Center, Moffett Field, California, USA, Tech. Rep. RNR-94-007, 1994.

[13] J. Subhlok, S. Venkataramaiah, and A. Singh, "Characterizing NAS Benchmark Performance on Shared Heterogeneous Networks," in *IPDPS '02: Proc. of the 16th International Parallel and Distributed Processing Symposium*. Washington, DC, USA: IEEE Computer Society, 2002.

[14] A. Faraj and X. Yuan, "Communication Characteristics in the NAS Parallel Benchmarks." in *IASTED PDCS*, 2002, pp. 724–729.

[15] J. Semke, J. Mahdavi, and M. Mathis, "Automatic TCP buffer tuning," in *Proceedings of the ACM SIGCOMM '98*. New York, NY, USA: ACM Press, 1998, pp. 315–323.

[16] M. Mathis and R. Reddy, "Enabling High Performance Data Transfers on Hosts," Pittsburgh Supercomputer Center, Tech. Rep., 1999.