Etude d'implémentations MPI dans une grille de calcul

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

RenPar'18 / SympA'2008 / CFSE'6
Fribourg, Suisse, du 11 au 13 février 2008

Etude d'implémentations MPI pour une grille de calcul

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

LIP (Laboratoire de l'Informatique du Parallélisme), École Normale Supérieure, Université de Lyon

Résumé
De nos jours, les grappes de PC ou clusters sont souvent interconnectés par des réseaux longue-distance de manière à former une grille afin d'offrir à un grand nombre d'utilisateurs un nombre plus conséquent de ressources. MPI, la bibliothèque de communication la plus utilisée pour les applications parallèles, a été efficacement implémentée dans un contexte de clusters. Deux caractéristiques des grilles, les réseaux longue-distance et l'hétérogénéité des processeurs et des réseaux, posent la question de l'efficacité de MPI sur les grilles.
Cet article présente une évaluation sur la grille de recherche française GRID'5000, de 4 implémentations récentes de MPI : MPICH2, MPICH-Madeleine, OpenMPI et GridMPI. La comparaison est basée sur un pingpong, les NAS Parallel Benchmarks. Nous mettons en évidence les différences de performance obtenues avec les 4 implémentations. GridMPI montre les meilleures performances. L'exécution d'applications MPI sur la grille peut être bénéfique à condition de régler finement certains paramètres des implémentations. Cet article détaille les paramètres mis en jeu et leurs réglages.

Mots-clés : MPI, Grille, Communications, Grid'5000

1. Introduction

De nos jours, les clusters sont souvent reliés par des réseaux longue distance, formant des grilles, qui permettent d'offrir une quantité de ressources très importante aux utilisateurs. MPI est la bibliothèque de communication standard utilisée pour écrire des applications parallèles pour ces grilles.
Les utilisateurs souhaitent exécuter sur des grilles les applications qu'ils avaient écrites pour des clusters afin de disposer du plus grand nombre de ressources. Par exemple, des applications comme ray2mesh [10], une application sismique de projection de rayons dans un maillage 3D de la Terre, ou des applications médicales comme Simri [4], un simulateur d'imagerie à résonance magnétique en 3D ont été portées sur la grille avec succès. Cependant, MPI a été initialement écrit pour des clusters et ne prend pas en considération les spécificités des grilles.
L'exécution des applications MPI pose essentiellement trois problèmes quand on veut les porter sur des grilles de calcul. Premièrement, les implémentations de MPI doivent gérer de manière efficace les liens longue-distance entre les différents sites. La forte latence entre les sites est très coûteuse, particulièrement pour les petits messages. Les communications inter-sites prennent plus de temps que les communications intra-sites. Les liens inter-sites offrent des débits plus élevés que les communications intra-sites.
Deuxièmement, les implémentations MPI doivent prendre en compte l'hétérogénéité des différents réseaux rapides présents dans les grilles. Il s'agit d'une part de permettre des communications intra-site (par exemple communiquer entre un réseau Myrinet et un réseau Infiniband) et d'autre part de gérer les communications inter-sites c'est à dire entre deux cluster Myrinet séparés par un WAN (Wide Area Network) sur lequel TCP est utilisé. Ce problème n'est pas traité dans notre étude. Dans toutes nos expérimentations, les communications utilisent TCP.
Enfin, l'hétérogénéité des processeurs a un impact sur la performance des applications. En particulier, il peut être intéressant de prendre en compte les différentes capacités des processeurs lors du placement des tâches.
Dans cette optique, MPICH2 [12], GridMPI [11], MPICH-Madeleine [2], OpenMPI [9] ou MPICH-G2 [14] peuvent être utilisés sur des grilles mais ne sont pas optimisés de la même manière. Le but de cet article est d'identifier les implémentations réellement efficaces sur les grilles et dans quelles conditions elles le sont. Une autre considération de cet article est de définir le bon paramétrage de ces implémentations pour obtenir les meilleures performances.
Enfin, il peut être intéressant de dégager parmi les applications celles dont le schéma de communication est le plus adapté à la grille.

Pour répondre à ces questions, nous avons réalisé des expériences sur Grid'5000 [5] une grille de recherche française regroupant plus de 3000 processeurs. Neuf sites sont reliés par un WAN dédié à 1 où 10 Gbps. Cette architecture offre aux chercheurs la possibilité de reconfigurer et de déployer dynamiquement n'importe quel système d'exploitation sur n'importe quel nœud.
Cet article est organisé comme suit. La section 2 présente un état de l'art des implémentations de MPI les plus proches d'un contexte de grille. Dans la section 3, nous décrivons le protocole expérimental que nous avons utilisé pour conduire nos évaluations. La section 4 présente nos résultats expérimentaux et les optimisations nécessaires pour obtenir de bonnes performances. La dernière section est consacrée à la conclusion et aux travaux à venir.

2. Les implémentations de MPI et applications pour la grille

Dans cette partie, nous présentons dans un premier temps des implémentations de MPI qui utilisent certaines spécificités des grilles. Dans un deuxième temps, nous donnons des exemples de travaux présentant l'exécution d'applications sur des grilles.

2.1. Les implémentations de MPI pour les clusters de clusters et les grilles

Cette section présente une synthèse des implémentations qui améliorent soit l'utilisation de l'hétérogénéité du réseau, soit les communications longue-distance : GridMPI [11], OpenMPI [9], MPICH-Madeleine [2], MPICH-G2 [14]. Nous présentons de plus MPICH2 [12], implémentation très utilisée.

2.1.1. Prise en compte de la longue distance

Les optimisations des communications longue-distance peuvent être réalisées à deux niveaux :les opérations collectives et les communications sur TCP. Ce problème est pris en compte dans deux implémentations : MPICH-G2 et GridMPI.
Pour GridMPI, dans un contexte de grille, la bande passante entre les clusters est plus grande que la bande passante au sein d'un même cluster alors que dans MPICH-G2, la hiérarchie prise en compte est : WAN $<$ LAN $<$ intra-machine TCP $<$ Vendor MPI (du plus lent au plus rapide). Les deux implémentations proposent donc des algorithmes optimisés pour les opérations collectives pour les hiérarchies de réseaux considérées. GridMPI optimise MPI_Bcast and MPI_Allreduce et MPICH-G2 améliore toutes les opérations collectives excepté MPI_Gatherv, MPI_Scatterv et MPI_Alltoallv.
Deuxièmement, ces implémentations modifient le comportement de TCP pour l'adapter aux fortes latences des réseaux longue-distance. Lors de la phase de démarrage de TCP, les développeurs de GridMPI proposent de modifier TCP de la manière suivante : ajout d'un mécanisme de pacing pour éviter les effets de vague (ce mécanisme espace les paquets pour éviter leur rejet par le switch), réduction des temps de retransmission (RTO), utilisation de deux fenêtres de congestion différentes pour les opérations collectives et les communication point-à-point, contournement de la couche socket. La distribution de GridMPI ne fournit pour l'instant que le mécanisme de pacing. MPICH-G2 permet le transfert de gros messages via plusieurs flux TCP en parallèle. Ce mécanisme est fourni par GridFTP et nécessite quelques modifications du code source.

2.1.2. Gestion de l'hétérogénéité

Deux des implémentations étudiées gèrent l'hétérogénéité des réseaux grâce à des passerelles (MPICH-Madeleine et OpenMPI). Ce mécanisme permet de communiquer entre deux clusters équipés de réseaux haute-performance distincts, sans passer par TCP. Par exemple, si le cluster A est équipé d'un réseau Myrinet, le cluster B d'un réseau SCI et qu'un nœud est partagé par les deux clusters, des messages peuvent transiter directement de A vers B sans utiliser TCP.
Dans MPICH-Madeleine, l'hétérogénéité est gérée par Madeleine[1], une bibliothèque de communication qui permet l'utilisation (homogène ou hétérogène) des réseaux TCP, SCI, VIA, Myrinet MX/GM ou Quadrics. Dans OpenMPI, la progression des communications est gérée par un module appelé PML pour Point-to-point Management Layer (couche de gestion point-à-point). Ce module peut utiliser différentes couches de transfert des données (BTL pour Byte Tranfer Layer), une pour chaque protocole, fournissant ainsi un support pour l'hétérogénéité des réseaux. Les protocoles actuellement supportés sont TCP, Myrinet MX/GM et Infiniband OpenIB/mVAPI.
GridMPI et MPICH-G2 gèrent l'hétérogénéité en utilisant une surcouche au-dessus de MPI propriétaires. Chaque cluster peut utiliser une implémentation MPI spécifique, propriétaire pour les communications locales sur réseau rapide et communiquer avec un autre cluster grâce à la couche supérieure. Dans GridMPI, Cette surcouche est gérée par IMPI (Interoperable MPI).

2.1.3. Synthèse

	Optimisation longue-distance	Gestion de l'hétérogénéité des réseaux	Première/Dernière publi.
MPICH2	Aucune	Aucune	2002 / 2006
GridMPI	Optimisation de TCP Optimisation de Bcast et All_reduce	Utilisation de TCP au-dessus de MPI propriétaires via IMPI	2004 / 2006
MPICH-Madeleine	Aucune	Passerelle entre réseaux rapides Support de TCP, SCI, VIA, Myrinet MX/GM, Quadrics	2003 /
OpenMPI	Aucune	Passerelle entre réseaux rapides Support de TCP, Myrinet MX/GM, Infiniband OpenIB/mVAPI	2004 / 2007
MPICH-G2	Optimisation des opérations collectives Flux parallèles pour les gros messages	Utilisation de TCP au-dessus de MPI propriétaires	2003

TAB. 1 – Comparaison des caractéristiques des implémentations MPI

Le tableau 1 présente les caractéristiques principales des différentes implémentations de MPI. Notre étude sera basée sur quatre implémentations : MPICH2, GridMPI, MPICH-Madeleine et OpenMPI. Du fait de la gestion lourde de certificats nécessaire pour Globus, nous n'avons pas encore pu tester MPICH-G2, le travail est en cours. FT-MPI[7] et MPICH-V[6] sont d'autres implémentations de MPI pour les grilles mais traitent des problèmes de tolérance aux pannes alors que nous nous intéressons aux performances d'exécution.

2.2. Exemple d'exécution d'application MPI sur la grille

ray2mesh [13] est une application de physique composée d'un ensemble de logiciels de sismologie qui utilise MPI. Dans [10], les auteurs présentent les expérimentations qu'ils ont réalisé sur Grid'5000. Les expériences ont été réalisées sur un nombre et une distribution variable des nœuds en utilisant LAM/MPI. Les speed-up obtenus sont linéaires en fonctions du nombre de nœuds mais le speed-up de la phase de communication reste constant quand le nombre de nœuds dépasse 192. En fait, cette phase échange les résultats entre tous les nœuds et devient très couteuse lorsque le nombre de nœuds est élevé.

2.2.1. Simri

Simri[4] est un simulateur d'imagerie à résonnance magnétique. Cette application est à parallélisation triviale. Les expériences ont été réalisées sur un cluster de 8 Pentium III, 1 Ghz en utilisant MPICH-G2 et Globus 2.2. Il est à noter que la synchronisation et la communication ne prend que 1.5% du temps total d'exécution si la taille de l'objet en entrée est supérieure à 256*256. L'exécution sur un cluster de 8 nœuds atteint une efficacité proche de 100%, i.e. la phase de calcul prend 7 fois moins de temps sur 8 nœuds que sur 1 (NB : le nœud maître n'effectue aucun calcul).

3. Protocole expérimental

3.1. Buts et méthodologie

Nos expériences ont deux buts principaux. Nous souhaitons tout d'abord exécuter des applications MPI sur différentes implémentations existantes dans un contexte de grille de calcul. Ensuite, nous comparons l'exécution sur une grille à l'exécution sur une grappe.

Pour comparer les différentes implémentations, nous exécutons un pingpong MPI entre deux nœuds appartenant à deux sites différents avec les paramètres par défaut de chaque implémentation. Nous comparons ces résultats avec la même expérience entre deux nœuds d'un même cluster. Nous exécutons également un pingpong TCP pour mesurer le surcoût dû à MPI pour chaque implémentation. Cependant, un pingpong MPI n'est pas représentatif des applications réelles qui s'exécutent sur une grille. Nous avons donc exécuté également les NPB (NAS Parallel Benchmarks [3]).

Le pingpong MPI est notre propre programme. Un premier processus MPI envoie des messages (de 1 octet à 64 Mo) à l'aide de la primitive MPI_Send à un autre processus. Ce dernier retourne le message aussitôt qu'il l'a reçu. Nous mesurons le temps aller-retour, qui nous permet ensuite de tracer la courbe de bande passante entre deux nœuds.

Les NPB sont un ensemble de 8 programmes (BT, CG, EP, FT, IS, LU, MG and SP) qui donnent un aperçu représentatif des applications parallèles qui peuvent s'exécuter sur un cluster ou sur une grille. Ces programmes ont été choisis pour mesurer les performances des clusters mais peuvent également être employés pour comparer

	Type de comm.	Taille et nombre des messages	Temps d'exécution sur un cluster
EP	P. à Point	192 * 8 o + 68 * 80 o	64 s
CG	P. à Point	126479 * 8 o + 86944 * 147 ko	75 s
MG	P. à Point	50809 * diff. tailles de 4 o à 130 ko	11 s
LU	P. à Point	1200000 * 960 o<msg<1040 o	250 s
SP	P. à Point	57744 * 45 ko<msg<54 ko + 96336 * 100 ko<msg<160 ko	280 s
BT	P. à Point	28944 * 26 ko + 48336 * 146 ko<msg<156 ko	430 s
IS	Collective	176 * 2 * 1 ko + 176 * 16 * 128 ko	5 s
FT	Collective	352 * 16 * 128 ko	90 s

TAB. 2 – Caractéristiques des NPB

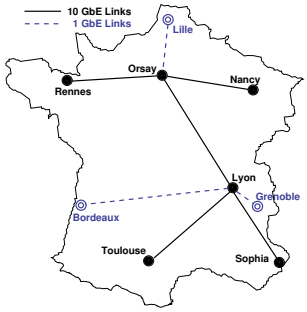

FIG. 1 – Réseau d'interconnexion de Grid'5000

des implémentations MPI. Les données fournies en entrée des NPB sont de différentes tailles classées en six classes (S, W, A, B, C, D). Pour nos mesures, nous avons utilisé la classe B entre 4 ou 16 nœuds.

Les programmes ont différents schémas de communication. Dans [17], les auteurs montrent que toutes les applications sont symétriques excepté CG. L'article [8] donne le type de communications et le nombre de messages pour la classe A sur 16 nœuds. Nous avons fait le même type d'expériences avec un MPI instrumenté pour obtenir le nombre, la taille des messages et le temps d'exécution. Les résultats sont résumés dans le Tableau 2. EP fait principalement du calcul et ne communique que très peu. CG et MG communiquent avec des messages de tailles variées. LU envoie des messages de taille moyenne. BT et SP communiquent avec de nombreux messages de grande taille. Enfin, FT et IS communiquent à l'aide d'opérations collectives : FT utilise la primitive MPI_Alltoall et IS les primitives MPI_Allreduce et MPI_Alltoallv. L'exécution de IS s'effectue en un temps beaucoup plus réduit que FT, la proportion du temps de communication par rapport au temps de calcul est donc beaucoup plus importante pour IS.

3.2. Plateforme d'évaluation

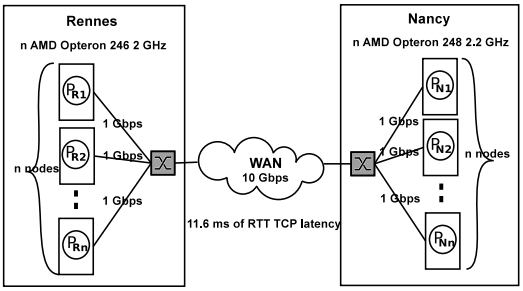

FIG. 2 – Configuration utilisée

	Dans le cluster de Rennes	Dans la grille : entre Rennes et Nancy
TCP	41	5812
MPICH2	46 (+5)	5818 (+6)
GridMPI	46 (+5)	5819 (+7)
MPICH-Mad.	62 (+21)	5826 (+14)
OpenMPI	46 (+5)	5820 (+8)

TAB. 3 – Comparaison de la latence dans un cluster et dans une grille (en us)

Nos expériences ont été menées à l'aide de la grille de recherche Grid'5000. La Figure 1 représente l'interconnexion des 9 sites de la grille (Bordeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia et Toulouse) répartis sur toute la France. Les sites sont interconnectés par un WAN à 1 ou 10 Gbps opéré par RENATER [16]. Les latences entre les sites sont hétérogènes. Par exemple, la latence TCP entre Lyon et Grenoble est de 4 ms alors que celle entre Rennes et Sophia est de 19 ms. Les puissances des nœuds sont hétérogènes de même que les réseaux rapides intra-sites (Infiniband, Myrinet, Ethernet 1 ou 10 Gbps).

Cette architecture fournit également aux chercheurs la possibilité de configurer aisément leurs nœuds grâce à un système de déploiement. Ils peuvent ainsi exécuter leurs applications avec leur propre système d'exploitation et obtenir les droits d'administration.

Nos expériences se déroulent sur deux sites Rennes et Nancy. Les machines utilisées sont des Opterons (resp. 2.2 Ghz et 2 Ghz) munies de cartes 1 GbE. Le système d'exploitation est une Debian, avec un noyau 2.6.18. La version de TCP est BIC, utilisée avec SACK. Nos expériences se déroulent entre 1, 2 ou 8 nœuds sur chaque site (Nancy ou Rennes) selon l'expérience réalisée (exécution du pingpong ou NPB). Chaque nœud est connecté au

switch grâce à une carte Ethernet 1 Gbps, ce qui implique que la bande passante disponible de bout en bout est de 1 Gbps également. La latence TCP aller-retour est de 11.6 ms.
Les versions des implémentations MPI sont les suivantes : MPICH2 est la version 1.0.5. GridMPI est la version de GridMPI 1.1. Nous n'utilisons pas le support de IMPI puisque toutes les communications utilisent TCP. MPICH-Madeleine est une version SVN du 6 décembre 2006. Elle utilise les threads (`-lib=-lpthread`) et le fast buffering (`-with-device= ch_mad :-fast-buffer`. Enfin, OpenMPI est la version 1.1.4.

4. Résultats des expériences et optimisations

4.1. Paramètrage de TCP

Pour permettre de bonnes performances sur TCP, des études [15] ont montré que la taille des buffers de sockets doit être fixée au minimum au produit RTT * bande passante. Nous avons donc augmenté les valeurs maximales de ces valeurs dans le noyau linux.
Dans nos expériences entre Rennes et Nancy, la taille des tampons de réception doit être au moins égale à 1.45 Mo (RTT=11.6 ms, bandwidth=1 Gbps). Cependant, pour prendre en considération le reste des latences de la grille, nous l'avons fixée à 4 Mo. Ces modifications sont suffisantes pour MPICH2, GridMPI et MPICH-Madeleine. Pour OpenMPI, la taille des tampons doit être fixée à l'exécution. La taille par défaut est de 128 ko. Cette taille est modifiable dynamiquement, en passant des paramètres à mpirun : `-mca btl_tcp_sndbuf 4194304 -mca btl_tcp_rcvbuf 4194304`.

4.2. Résultats du pingpong

Le pingpong mesure 200 aller-retour entre deux nœuds, avec les implémentations MPI par défaut. Les résultats sur un cluster ont été obtenus entre deux nœuds du cluster de Rennes, ce qui correspond aux processus P_{R1} et P_{R2} sur la Figure 2. Nous prenons ensuite le meilleur résultat parmi les 200 pour éliminer les perturbations éventuelles dues au trafic de fond de Grid'5000.
Le Tableau 3 montre une comparaison de la latence induite par chaque implémentation MPI, soit entre deux nœuds d'un même cluster, soit entre deux nœuds de la grille (l'un situé sur Rennes, l'autre sur Nancy). Le chiffre entre parenthèses représente le surcoût lié uniquement à l'implémentation MPI (sans tenir compte de la latence TCP). Ce surcoût est similaire pour toutes les implémentations MPI, excepté pour MPICH-Madeleine où il est plus important. Il y a peu de différence entre le surcoût lié à la traversée de MPI pour un cluster ou pour la grille. En effet, il est naturel que le temps de traversée de la bi bliothque MPI ne dépende pas de la latence dans le cas d'un pingpong. Ces chiffres confirment le coût de la longue distance (11600 µs de ping RTT équivalent au 5800 µs d'un aller simple).
La Figure 3 représente la bande passante obtenue sur un cluster. Toutes les implémentations atteignent les 940 Mbps qui correspond à la bande passante utile maximale atteignable en TCP sur un lien 1 Gbps. Toutes les implémentations montrent un seuil pour des messages supérieurs à 128 ko excepté GridMPI. Ce seuil est dû au différents modes d'envoi de MPI (eager et rendez-vous).
La Figure 4 montre les performances obtenues après l'optimisation de TCP et les modifications d'OpenMPI. La bande passante maximale obtenue est de 900 Mbps dans la grille et s'approche des 940 Mbps obtenus dans un cluster. Cependant, le demi-débit est seulement atteint pour des messages de 1 Mo contre 8 ko dans un cluster. Cette différence s'explique du fait de la latence inférieure dans un cluster (40 µs) que dans une grille (5800 µs). Finalement, les performances de GridMPI sont meilleures que les autres implémentations et très proches des résultats obtenus avec le pingpong TCP. Nous observons encore des seuils aux alentours de 128 ko excepté pour GridMPI (mode eager et rendez-vous).
Une analyse des résultats du pingpong nous a montré que l'écart entre le temps minimun et le temps maximum d'envoi d'un message (parmi les 200 envoyés) était important. Nous avons donc utilisé le pingpong pour mesurer le temps d'envoi de chacun des 200 messages. La Figure 5 représente la bande passante obtenue pour chacun des messages. Nous pouvons remarquer que l'impact du slowstart et du contrôle de congestion inclus dans TCP est très important. Ces mécanismes interviennent sur chaque nœud impliqué dans le pingpong. TCP atteint la bande passante maximale après 5s. GridMPI a un comportement similaire. Cela correspond à 35 aller-retours entre les deux nœuds. Cependant, dans les autres implémentations, le nombre d'aller-retours nécessaire pour quitter la phase de contrôle de congestion est plus important (de l'ordre de 84 pour MPICH-Madeleine et 96 pour OpenMPI et MPICH2).

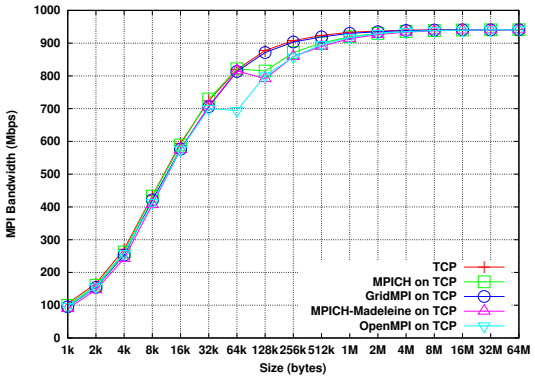

FIG. 3 – Comparaison de la bande passante MPI pour différentes implémentations sur un réseau local (cluster) avec les paramètres par défaut

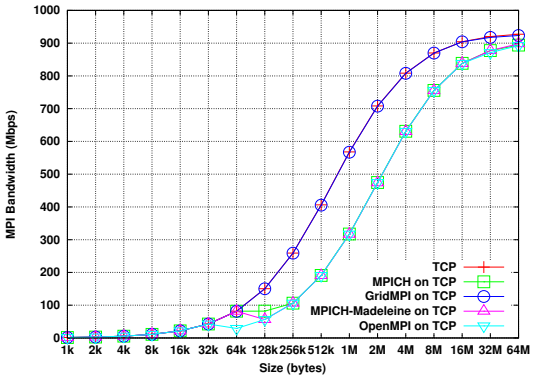

FIG. 4 – Comparaison de la bande passante MPI pour différentes implémentations sur un réseau de grille après optimisation de TCP

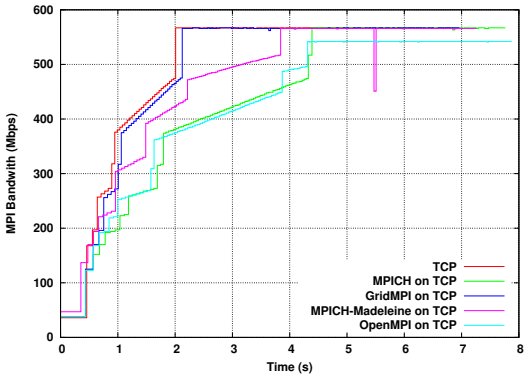

FIG. 5 – Impact du slowstart TCP lors d'un ping-pong de messages de 1 Mo sur la grille pour les implémentations MPI

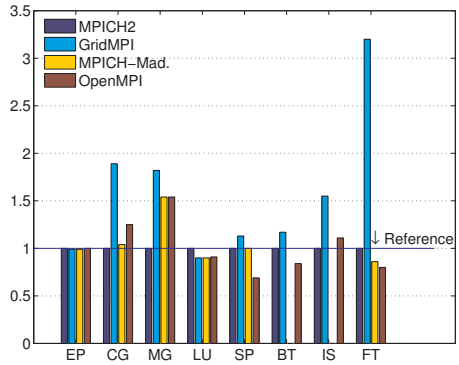

FIG. 6 – Comparaison relative des différentes implémentations sur 8-8 nœuds sur deux clusters, MPICH2 est la référence

4.3. NAS Parallel Benchmarks

Les expériences suivantes ont pour but de comparer les performances sur une grille des implémentations MPI (Fig. 6) et de voir si les applications peuvent tirer efficacement profit des ressources d'une grille de calcul (Fig. 7 et 8). Nous avons donc pour cela exécuté les NPB version 2.4 dans deux configurations différentes : sur les nœuds d'un même cluster (en utilisant P_{R1} à P_{R16}, cf Fig. 2) ou sur les nœuds de deux clusters séparés par un WAN (en utilisant P_{R1} à P_{R8} et P_{N1} à P_{N8}). Chaque NPB est exécuté 5 fois, la valeur retenue étant le minimum des temps d'exécution. La Figure 6 montre le temps d'execution des NPB pour chaque implémentation relativement à MPICH2.

Le speed-up de GridMPI est important sur la grille dû à ses opérations collectives. En effet, les autres implémentations proposent des optimisations efficaces pour les clusters. Or, celles-ci ne sont plus utiles et même pénalisantes dans le cas des grilles. De ce fait, les performances de GridMPI sont très bonnes sur la grille pour les programmes qui utilisent des opérations collectives. Les performances avec MG sont similaires pour toutes les implémentations excepté MPICH2. Cependant les performances sur LU sont meilleures avec cette dernière. Enfin, GridMPI montre des performances un peu meilleures avec BT et SP. Les résultats pour BT et SP n'ont pas pu être obtenus avec MPICH-Madeleine (le programme ne termine pas).

Les expériences suivantes (Figures 7 et 8) ont pour but de voir le surcoût provoqué par l'exécution dans une grille comparé à l'exécution dans un cluster. Comme nous l'avons vu précédemment sur la Figure 6, GridMPI montre le meilleur comportement pour toutes les applications. Nous avons donc choisi cette implémentation pour comparer l'exécution des NPB sur la grille à celle sur un cluster.

La Figure 7 représente les performances relatives des NPB sur 16 nœuds divisés en deux groupes de 8 par un WAN à 16 nœuds sur un même cluster. Cette figure montre l'impact de la latence pour chaque type d'application. EP communique très peu, aussi ses performances relatives sont proches de 1. CG et MG sont peu performants

sur la grille du fait de leurs petits messages, la latence ayant un impact important sur ces derniers. LU, SP et BT communiquent avec des tailles de messages assez importantes et leur temps de calcul est assez important. Le recouvrement des temps de communication et de calcul est donc plus efficace. IS effectue de nombreuses communications collectives pendant un temps très court. La longue-distance a donc un impact important. Enfin, FT communique également avec des opérations collectives (dont les messages sont de tailles équivalentes à IS) mais le temps d'exécution est suffisament grand pour que l'impact sur les communications soit moindre.

La comparaison stricte du même nombre de nœuds dans la grille et dans un cluster montre nécessairement un surcoût pour la grille. Cependant, celui-ci est inférieur à 20% pour la moitié des NAS. Or le principal intérêt d'une grille de calcul est de mutualiser des ressources pour pouvoir exécuter des applications à plus large échelle. Il faut donc se demander si l'utilisation de la grille avec plus de noeuds que sur un cluster permet un gain de performances. Ainsi, la Figure 8 représente la performance relative de 4 nœuds sur un cluster à 16 nœuds sur deux clusters. Ces résultats confirment qu'il est efficace d'exécuter des applications MPI sur une grille de calcul si on peut disposer de plus de ressources. Nous n'abordons pas ici le problème du goulot d'étranglement que constitue l'accès au réseau WAN. Cependant, les applications avec de petits messages et beaucoup de communications ont de très mauvaises performances à cause de la latence. Les opérations collectives doivent être optimisées pour être efficaces.

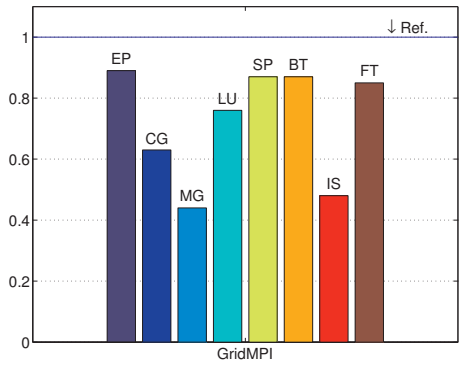

FIG. 7 – Comparaison relative de 16 nœuds sur le même cluster et 8-8 nœuds sur deux clusters différents, 16 nœuds sur un cluster est la référence (Ref.)

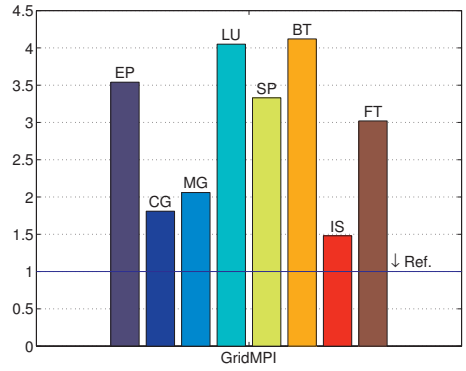

FIG. 8 – Comparaison relative de 4 nœuds sur un cluster et 8-8 nœuds sur deux clusters, 4 nœuds sur le même cluster est la référence (Ref.)

5. Conclusion et travaux futurs

Dans cet article, nous avons présenté une étude des différentes implémentations de MPI dans un contexte de grille. Notre but était de trouver une bonne implémentation pour l'exécution d'applications dans ce contexte mais aussi d'identifier les types d'applications et les schémas de communication les mieux adaptés aux grilles.

Nos expérimentations sont basées sur 4 implémentations de MPI : MPICH2, GridMPI, MPICH-Madeleine et OpenMPI. Elles montrent qu'un réglage approprié de TCP dans les implémentations de MPI est nécessaire pour obtenir de bonnes performances. Après optimisations, toutes les implémentations de MPI peuvent obtenir des performances proches de celles de TCP. Pour compléter notre étude, nous avons évalué les implémentations en nous basant sur les NPB. Les résultats montrent que GridMPI se comporte mieux que les autres implémentations sur Grid'5000. Ceci est principalement dû au meilleur comportement de GridMPI avec TCP et à ses opérations collectives plus efficaces sur le longue distance. Cependant, GridMPI ne gère pas l'hétérogénéité des réseaux rapides et impose de disposer de TCP de bout en bout. L'exécution d'applications sur la grille est efficace quand les applications utilisent de gros messages en communication point-à-point. Les opérations collectives doivent être optimisées pour obtenir de bons résultats.

Dans un futur proche, nous poursuivrons nos expériences avec d'autres implémentations de MPI, comme MPICH-G2, et des applications du monde réel. Nos résultats ont mis en évidence des problèmes liés aux mécanismes de slowstart et d'évitement de congestion dans un contexte de grille. Nous allons poursuivre notre étude pour mieux interfacer MPI et TCP dans un contexte de grille.

Nous poursuivrons notre étude sur un plus grand nombre de noeuds afin d'étudier le comportement des différentes implémentations MPI en cas de saturation de l'accès au réseau WAN.

Enfin, nous allons étudier le comportement de la gestion de l'hétérogénéité des réseaux avec différents types de réseaux rapides. L'utilisation de ces réseaux pour les communication locales peut avoir un impact primordial pour

les exécutions sur la grille. Mais le surcoût engendré par la gestion de l'hétérogénéité doit rester à un niveau faible pour ne pas rendre TCP plus intéressant que les réseaux rapides.

Remerciements

Ce travail a été financé par le ministère de l'éducation et de la recherche et le projet EU IST FP6 EC-GIN. Les expériences ont été menées en utilisant la plateforme expérimentale Grid'5000, une initiative du ministère de la recherche via l'action incitative ACI GRID, INRIA, CNRS, RENATER (https ://www.grid5000.fr)

Bibliographie

1. Aumage (Olivier). – Heterogeneous multi-cluster networking with the Madeleine III communication library. *In : 16th International Parallel and Distributed Processing Symposium*. – Florida, USA, 2002.
2. Aumage (Olivier) et Mercier (Guillaume). – MPICH/MADIII : a Cluster of Clusters Enabled MPI Implementation. *In : Proceedings of CCGrid*. – Tokyo, 2003.
3. Bailey (D), Barscz (E), Barton (J), Browning (D), Carter (R), Dagum (L), Fatoohi (R), Fineberg (S), Frederickson (P), Lasinski (T), Schreiber (R), Simon (H), Venkatakrishnan (V) et Weeratunga (S). – *The NAS Parallel Benchmarks*. – Rapport technique n RNR-94-007, Moffett Field, California, USA, NASA Ames Research Center, 1994.
4. Benoit-Cattin (H), Bellet (F), Montagnat (J) et Odet (C). – Magnetic Resonance Imaging (MRI) simulation on a grid computing architecture. *In : Proceedings of IEEE CGIGRID'03-BIOGRID'03*. – Tokyo, 2003.
5. Bolze (Raphaël), Cappello (Franck), Caron (Eddy), Daydé (Michel), Desprez (Frédéric), Jeannot (Emmanuel), Jégou (Yvon), Lantéri (Stephane), Leduc (Julien), Melab (Noredine), Mornet (Guillaume), Namyst (Raymond), Primet (Pascale), Quetier (Benjamin), Richard (Olivier), Talbi (El-Ghazali) et Iréa (Touche). – Grid'5000 : a large scale and highly reconfigurable experimental grid testbed. *International Journal of High Performance Computing Applications*, vol. 20, n4, novembre 2006, pp. 481–494. – https ://www.grid5000.fr/.
6. Bosilca (George), Bouteiller (Aurélien), Cappello (Franck), Djilali (Samir), Fédak (Gilles), Germain (Cécile), Hérault (Thomas), Lemarinier (Pierre), Lodygensky (Oleg), Magniette (Frédéric), Néri (Vincent) et Selikhov (Anton). – MPICH-V : Toward a Scalable Fault Tolerant MPI for Volatile Nodes. *In : Proceedings of The IEEE/ACM SC2002 Conference*.
7. Fagg (Graham E.) et Dongarra (Jack). – Ft-mpi : Fault tolerant mpi, supporting dynamic applications in a dynamic world. *In : PVM/MPI*, pp. 346–353.
8. Faraj (Ahmad) et Yuan (Xin). – Communication Characteristics in the NAS Parallel Benchmarks. *In : IASTED PDCS*, pp. 724–729.
9. Gabriel (Edgar), Fagg (Graham E.), Bosilca (George), Angskun (Thara), Dongarra (Jack J.), Squyres (Jeffrey M.), Sahay (Vishal), Kambadur (Prabhanjan), Barrett (Brian), Lumsdaine (Andrew), Castain (Ralph H.), Daniel (David J.), Graham (Richard L.) et Woodall (Timothy S.). – Open MPI : Goals, Concept, and Design of a Next Generation MPI Implementation. *In : Proceedings, 11th European PVM/MPI Users' Group Meeting*, pp. 97–104. – Budapest, Hungary, September 2004.
10. Genaud (Stéphane), Grunberg (Marc) et Mongenet (Catherine). – Experiments in running a scientific MPI application on Grid'5000. *In : 4th High Performance Grid Computing International Workshop, IPDPS conference proceedings*. – Long beach, USA, March 2007.
11. GridMPI Project. http ://www.gridmpi.org/gridmpi.jsp.
12. Gropp (William). – 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.
13. Grunberg (Marc), Genaud (Stéphane) et Mongenet (Catherine). – Seismic ray-tracing and Earth mesh modeling on various parallel architectures. *The Journal of Supercomputing*, vol. 29, n1, July 2004, pp. 27–44.
14. N. Karonis, B. Toonen (I. Foster). – MPICH-G2 : A Grid-Enabled Implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing*, 2003, pp. 551–563.
15. Semke (Jeffrey), Mahdavi (Jamshid) et Mathis (Matthew). – Automatic TCP buffer tuning. *In : Proceedings of the ACM SIGCOMM '98*. pp. 315–323. – New York, NY, USA, 1998.
16. Site RENATER. – Renater4. http ://www.renater.fr/.
17. Subhlok (Jaspal), Venkataramaiah (Shreenivasa) et Singh (Amitoj). – Characterizing NAS Benchmark Performance on Shared Heterogeneous Networks. *In : IPDPS '02 : Proc. of the 16th International Parallel and Distributed Processing Symposium*. – Washington, DC, USA, 2002.