Evaluation et amélioration des performances d'une implémentation MPI pour la grille

Ludovic Hablot, Olivier Glück

Laboratoire de l'Informatique du Parallélisme

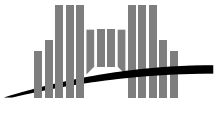
École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

MPICH/Madeleine : Evaluation and optimization of long distance communications

Ludovic Hablot, Olivier Glück Jul 2006

Research Report N° RR2006-26

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr



MPICH/Madeleine : Evaluation and optimization of long distance communications

Ludovic Hablot, Olivier Glück

Jul 2006

Abstract

The MPI standard is often used in parallel applications for communication needs. Most of them are designed for homogeneous clusters but MPI implementations for grids have to take into account heterogeneity and long distance network links in order to maintain a high performance level. These two constraints are not considered together in existing MPI implementations.
State of the art of MPI implementations is followed by analysis of one of them, MPICH-Madeleine, which manages heterogeneity with efficiency. Our optimizations lead to obtain a bandwith of 600 Mbps instead of 95 Mbps for sending MPI messages over a Wide Area Network. The experiments have been performed on the Grid'5000 french national plateform.

Keywords: MPICH-Madeleine, MPI, Grid, WAN, heterogeneity, Grid'5000

Résumé

Les applications parallèles utilisent généralement le standard MPI pour réaliser leurs communications. La plupart des implémentations de MPI sont destinées aux grappes homogènes. Avec l'apparition des grilles de calcul, il est nécessaire de faire évoluer ces implémentations pour les adapter efficacement aux contraintes de ces nouvelles plateformes que sont la gestion de l'hétérogénéité et la prise en compte des liens réseau longue distance permettant l'interconnexion des sites de la grille. Aucune implémentation actuelle ne prend en compte efficacement ces deux paramètres.
Après une étude des implémentations existantes, cet article analyse le comportement sur la grille de l'une d'entre elles, MPICH-Madeleine, qui propose une gestion efficace de l'hétérogénéité des réseaux rapides de grappe. A partir de nos premières expérimentations, nous proposons des optimisations permettant d'améliorer les performances d'exécution sur la grille. Elles nous ont permis d'augmenter très sensiblement la bande passante lors de l'exécution d'un ping-pong MPI : en passant de 95Mb/s à 600Mb/s sur la longue distance. Les expérimentations ont été réalisées sur la grille française Grid'5000.

Mots-clés: MPICH-Madeleine, MPI, grille, longue distance, hétérogénéité, Grid'5000

Sommaire

1 Introduction

Apparu dans les années 70-80, le calcul parallèle permet de répartir un calcul conséquent en plusieurs petits sous-calculs qui seront résolus séparément et en parallèle pour diminuer la durée totale d'exécution. Ceux-ci furent tout d'abord réalisés par les supercalculateurs avec des applications de type financier ou météorologiques. Mais avec le développement des technologies réseaux et du matériel informatique, les grappes, ensemble d'ordinateurs reliés par un réseau rapide, ont fait leur apparition. Ensuite, ces grappes ont évolué vers les grilles permettant de mettre en commun des ressources afin de réaliser ces calculs. Les applications scientifiques visées aujourd'hui sont, par exemple, situées dans le domaine spatial ou dans le domaine de la physique nucléaire.

Cependant, pour qu'une application puisse s'exécuter sur plusieurs machines, il est nécessaire de communiquer entre les différents processus (qui exécutent chacun une partie du calcul). Cette communication peut s'effectuer en utilisant une interface de programmation à passage de messages telle que MPI (Machine Parallel Interface [14]), ou en se placant au-dessus d'une machine virtuelle telle que PVM (Parallel Virtual Machine [27]). Au cours des dernières années, le standard MPI s'est imposé dans le monde des applications parallèles et des grappes de calcul. Avec l'apparition des grilles, il est naturel de vouloir exécuter efficacemenet les applications MPI sur ces plateformes émergentes. Certaines implémentations de MPI prennent partiellement en compte les nouvelles contraintes imposées par la grille mais aucune implémentation actuelle n'est totalement satisfaisante.

Dans le cadre de mon stage de Master, j'ai travaillé sur une implémentation existante de MPI, MPICH-Madeleine. Je me suis attaché à comprendre le comportement de cette implémentation sur la grille et à améliorer les performances dans ce contexte.

Ce travail est divisé en cinq parties. Dans un premier temps, je présenterai le contexte du calcul parallèle et des grilles de calcul. Dans un second temps, je m'attacherai à préciser les problèmes spécifiques qui se posent dans le cas des grilles : la gestion de l'hétérogénéité et la prise en compte des liens longue distance. Dans la partie suivante, je parlerai des différentes implémentations existantes répondant partiellement à nos problématiques et spécifiques aux grilles. Dans la quatrième partie, je présenterai les travaux que j'ai effectués sur MPICH-Madeleine. Pour finir, je présenterai mes conclusions concernant ce travail et les perspectives associées.

2 Contexte

Les calculs parallèles étaient autrefois, réalisés par des supercalculateurs. Mais avec l'évolution des matériels informatiques, les machines cibles ont évolué et les grappes de calcul puis les grilles sont apparues. Cette section présente les caractèristiques des machines pour le calcul parallèle et présente MPI, une solution pour l'exécution d'applications parallèles sur de telles machines.

2.1 Les machines cibles

2.1.1 Les grappes

Les grappes de calcul, aussi appelées clusters, sont un ensemble de machines standards que l'on peut trouver dans le commerce (mais la plupart du temps bi-processeurs), reliées en-

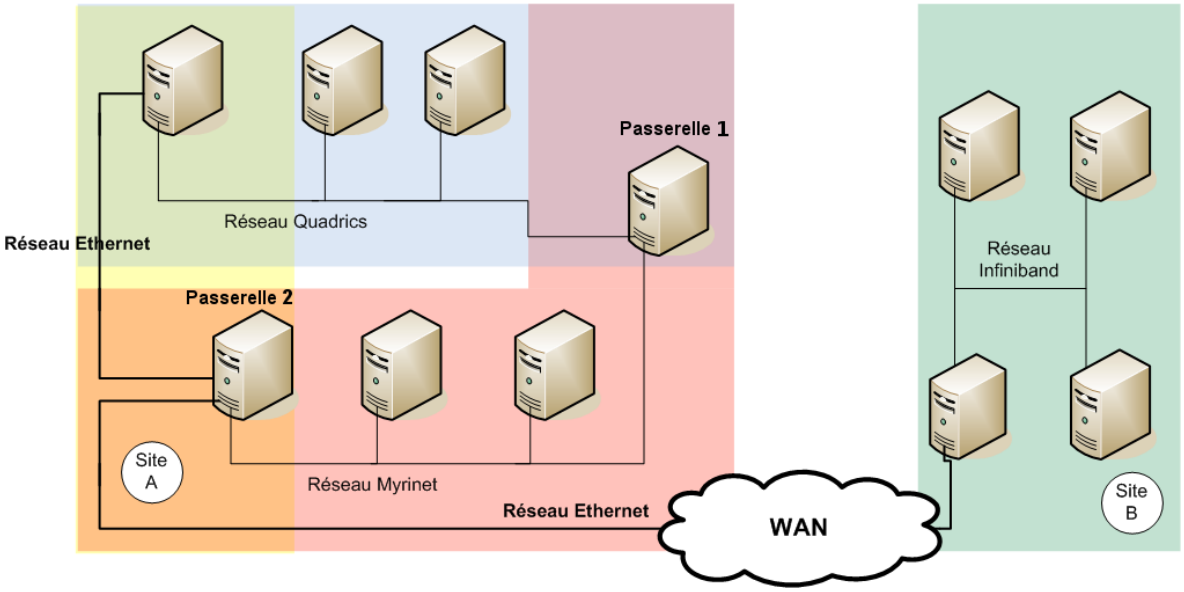


Figure 1: Un exemple de grille de calcul

tre elles par un réseau haut débit. Les machines utilisées coûtent beaucoup moins chers que les supercalculateurs et permettent de pouvoir étendre facilement la grappe. Les technologies utilisées sont homogènes au niveau du réseau et la plupart du temps au niveau des machines. Les différents types de réseaux qui peuvent interconnecter les grappes sont présentés un peu plus loin Table 1.

2.1.2 Les grappes de grappes

Les grappes de grappes sont l'interconnexion de plusieurs grappes par des réseaux hétérogènes. Elles sont apparues pour pouvoir obtenir une plus grande puissance de calculs par rapport aux grappes, en réutilisant une partie du matériel existant. L'interconnexion des grappes se fait par une ou plusieurs passerelles possèdant une interface dans chacun des réseaux qui constituent la grappe de grappes. Les passerelles peuvent éventuellement s'appuyer sur un réseau intermédiraire (généralement Ethernet) pour réaliser le passage d'un réseau rapide à l'autre.

Pour pouvoir mettre en commun des ressources, de manière à réduire les coûts, différentes entités ont cherché à relier les grappes ou grappes de grappes dont elles diposaient : les grilles de calcul ont donc fait leur apparition.

2.1.3 Les grilles

Alors que les grappes de grappes ont introduit l'hétérogénéité, des réseaux rapides en particulier, les grilles de calcul [9] doivent prendre en compte l'acheminement sur un réseau longue distance (WAN). En effet, une grille de calcul, comme le montre la Figure 1, est une interconnexion de grappes (voire de supercalculateurs) via un réseau longue distance. Il peut s'agir de liaisons IP classiques, de réseaux ATM, de liaisons louées ou encore de réseaux Ethernet longue distance. Cette architecture permet à plusieurs sites (potentiellement très

éloignés) de disposer d'une puissance de calcul plus importante grâce à cette agrégation de ressources. Cette interconnexion est rendue possible par l'amélioration des réseaux longue distance et on voit maintenant apparaître des grilles nationales, voire planètaires.

L'exemple de la Figure 1 nous montre un site A interconnecté avec un autre site B. Le site A (la partie gauche de la figure) est une grappe de grappes constituée d'une grappe Quadrics (en rose) et d'une autre connectée en Myrinet (en bleu). La passerelle 1 permet le passage direct des communications entre la grappe Quadrics et la grappe Myrinet. La passerelle 2 peut servir de passerelle entre les deux grappes via le réseau Ethernet mais sert également de passerelle vers le WAN. Le site B possède quant à lui un ensemble de machines reliées par un réseau Infiniband. Afin de mettre en commun leur ressources, les sites A et B ont donc souhaité relier leurs deux ensembles de grappes par un réseau longue distance (WAN). Cet ensemble forme une grille de calcul.

Le contexte des grilles fait apparaitre de nouvelles contraintes telles que :

- la gestion des droits d'accès à cet ensemble de ressources
- la gestion des ressources physiques (processeurs, mémoire) et le partage de celles-ci grâce à de nouveaux algorithmes de répartition des tâches
- la gestion de l'hétérogénéité et le masquage de celle-ci : le but est de donner l'illusion à l'application et à l'utilisateur qu'il travaille sur une seule machine
- ...

Parmi un grand nombre de grilles existant, on peut citer :

- LCG : grille administrée par le CERN, la plus grande du monde (à terme plus de 100 sites dans 31 pays) qui devrait atteindre les 355 TFlops. La plus grosse grappe du monde est seulement de 280TFlops (Nombre d'opérations à virgules flotantes par secondes) selon le TOP500 [33] qui classe les 500 plus gros systèmes du monde.
- DTF : joint trois sites aux Etats-Unis (13.6 TFlops)
- NAREGI : Grille Japonaise pour améliorer l'infrastructure de grille (17TFlops)
- DAS3 : une grille hollandaise (grille de recherche le jour et de production la nuit)
- Grid'5000 [7] : Cette grille française est destinée à la recherche et devrait, à terme, regrouper 5000 processeurs répartis sur 9 sites. Financée par le programme ACI Grid et le Ministère de la recherche, elle regroupe une communauté de chercheurs en informatique et a pour but de fournir un support pour la mise au point d'outils pour l'amélioration du réseaux ou des middleware de grille.

C'est sur cette dernière grille que vont se dérouler les expériences que nous allons mener. Cependant, pour que des applications parallèles puissent s'exécuter sur ces grappes ou sur les grilles, il est nécessaire d'avoir un langage commun pour communiquer : le langage MPI.

2.2 La bibliothèque de communication MPI

MPI ([8] et [14]) est la spécification d'un jeu de fonctions qui fournit un support pour la réalisation d'applications parallèles. Les squelettes des fonctions qu'elle fournit sont disponibles pour Fortran, C et C++. Les tâches d'une application parallèle ont besoin de communiquer pour s'échanger les données nécessaires à l'avancement de chacune des tâches. Pour cela, elles vont utiliser des primitives de communication fournies par l'API MPI. Il s'agit typiquement des fonctions d'envoi et de réception.

Le fonctionnement standard d'une application MPI est le suivant. Un processus (exécution d'une tâche) est lancé sur chaque processeur disponible et va exécuter une partie du calcul. Pour pouvoir se synchroniser, les processus vont utiliser différentes fonctions parmi lesquelles, on peut mentionner :

- des fonctions de communications point à point (`MPI_Send, MPI_Receive`)
- des fonctions de communications collectives. Elles permettent de communiquer une information de 1 vers n, de n vers 1 ou de n vers n processus.

Il existe actuellement deux versions du standard MPI. La version MPI-2 fournit de nouvelles fonctionnalités. Pour compléter MPI qui est une librairie statique, MPI-2[8] ajoute des fonctionnalités pour gérer la mémoire et rendre la bibliothèque dynamique.

3 Problématique

Comme nous venons de le voir dans la section 2, nous nous intéressons aux applications qui effectuent du calcul parallèle en utilisant MPI comme interface de communication entre les différents processus et qui sont susceptibles de s'exécuter sur une grille de calcul. Un utilisateur doit pouvoir écrire une application sans se préoccuper ni de l'implémentation MPI sous-jacente, ni de la plateforme d'exécution cible.

Il existe différentes implémentations de MPI mais la plupart sont dédiées à un type de réseau particulier et sont de ce fait destinées aux grappes homogènes. La Table 1 donne un aperçu des différents réseaux rapides utilisés dans les grappes, leur performance brute d'exécution ainsi que les implémentations MPI associées des performances obtenues sur ce type de réseaux ainsi que les implémentations qui leur sont associées. Or, dans le contexte des grilles de calcul, ces différentes implémentations sont peu ou pas utilisables. En effet, le contexte des grilles apporte deux nouvelles contraintes :

- la gestion de l'hétérogénéité des réseaux. Les réseaux rapides utilisés peuvent être différents sur chaque grappe. Il est toujours possible d'utiliser TCP sur Ethernet comme réseau "fédérateur", des différentes grappes constituant la grille mais cela ne permet pas de tirer parti des performances disponibles sur chacune des grappes. En effet, les mauvaises performances de TCP sur Ethernet masquent les très bonnes performances des réseaux rapides de grappes comme Myrinet, Quadrics ou Infiniband.
- la gestion des longues distances. Il est nécessaire de prendre en compte la latence et la bande passante du lien reliant les différents sites : les performances du réseau longue distance sont souvent très inférieures à celles du réseau intersite (en particulier la latence).

Ces contraintes doivent être prises en compte pour créer une implémentation performante de MPI adaptée aux grilles de calcul. En particulier, il parait opportun de pouvoir déterminer dynamiquement l'environnement d'exécution (en local ou sur la grille) et adapter le paramètrage des communications à chacun de ces contextes. De plus, pour permettre une efficacité optimale, l'utilisateur doit pouvoir exécuter son application au mieux sans qu'il se préoccupe de l'environnement et de la plateforme d'exécution : les adaptations les paramètrages et optimisations proposées doivent être transparents pour l'utilisateur.

Les implémentations présentées par la suite [21] permettent de répondre, chacune partiellement, à cette problématique mais aucune d'entre elles ne semble réellement efficace sur la grille.

4 Etat de l'art : les différentes implémentations de MPI pour les grappes et les grilles de calcul

Parmi les nombreuses implémentations de MPI, certaines seulement permettent d'éxécuter des applications sur les grilles. Je vais tout d'abord présenter GridMPI basée sur YAMPII et OpenMPI basée sur quatre implémentations antérieures. Je présenterai ensuite les implémentations basées sur est MPICH : MPICH-G2, MetaMPICH, MPICH-VMI et MPICH-Madeleine.

4.1 GridMPI

GridMPI [13] est une des rares implémentations de MPI à destination des grilles de calcul et à avoir été créée dans ce but. GridMPI dérive de YAMPII qui était elle destinée aux grappes et apporte des modifications pour améliorer les communications longues distances.La gestion de l'hétérogénéité repose sur des versions de MPI propriétaires telles que MPICH-MX ou MPI sur Infiniband. Une fois le message transmis à l'application. Pour être transmis vers un réseau rapide, un message doit donc traverser à la fois l'ADI de GridMPI et à la fois l'ADI propriétaire. Ceci n'est donc pas optimal.

L'architecture de GridMPI est décrite en annexe A.

TCP a été concu pour transférer un flux continu d'octets et non pas des rafales comme c'est le cas dans les applications MPI qui ont plutôt tendance à envoyer de gros messages de

Réseau	**Latence (us)**	**Bande passante max.(Gb/s)**	**Implémentation MPI associée**
GBEthernet	30-120	1	MPICH
10GBEthernet	10	6,9	MPICH
Myrinet MX-2G [16]	2,7	3,9	MPICH-MX
Myrinet 10G [16]	2	9,6	MPICH-MX
Quadrics [28]	1,29	7,2	QsNetMPI/Elan4 [17]
Infiniband [1]	4	7	MPI sur Infiniband [18]
SCI [32]	2	1,3	MPI sur SCI de grappes

Table 1: Comparatif des réseaux rapides de grappes

temps en temps. Les mécanismes à mettre en place dans le cas de MPI sont donc différents de ceux proposés par TCP.

Les modifications apportées par GridMPI pour les communications longue distance sont les suivantes [22]:

- changement de la taille de la fenêtre de congestion à la volée selon le type de message envoyé. Deux valeurs sont permutées : une pour les opérations collectives, une pour les opérations point à point. En effet, les opérations point à point utilisent ainsi toute la bande passante disponible et ne sont pas pénalisées si les opérations collectives précédantes ont fait chuter la fenêtre de congestion. Ceci permet d'avoir une fenêtre de congestion adaptée au type de message.
- diminution du temps de retransmission après un timeout. Dans le cas des grilles, on suppose que les liens réseaux sont rapides, ce qui implique que la congestion qui génère le timeout est plus rapide à se résorber que dans Internet.
- éviter la dégradation due à la phase de slow start en utilisant un mécanisme de "pacing". Celui-ci permet d'éviter l'envoi de paquets en rafales en espaçant les paquets lors de l'envoi de manière à ne pas dépasser la bande passante disponible. Lorsque l'on envoie des paquets en rafales durant la phase de slow start, cela conduit à diminution de la fenètre de congestion. En utilisant ce mécanisme pendant cette phase, on arrive on débit maximal beaucoup plus rapidement.

4.2 OpenMPI

Devant la majorité d'implémentations MPI incompatibles entre elles et proposant toutes des améliorations différentes, les auteurs ont souhaité proposer une nouvelle implémentation de MPI2, OpenMPI [11], basée sur des expériences passées : LAM/MPI [4] (implémentation standard), LA-MPI [31] (tolérance aux erreurs de transmission), FT-MPI [12](tolérence aux pannes) et PACX-MPI [5] (hétérogénéité).

L'architecture générale d'OpenMPI est décrite en annexe D. Dans cette architecture, on peut voir qu'un module entier est dédié aux communications. Celui-ci s'occupe de gérer l'hétérogénéité des réseaux sous-jacents en s'appuyant sur un module par type de réseaux. OpenMPI fournit un mécanisme permettant à ces derniers d'être utilisés conjointement. OpenMPI permet l'utilisation de réseaux hautes performances grâce au développement de pilotes TCP/IP, mémoire partagée, Myrinet (GM et MX), Quadrics, Infiniband. A terme, OpenMPI devrait permettre de gérer dynamiquement la perte d'un matériel réseau (pas encore géré) et d'optimiser la bande passante pour la longue distance.

Cependant, OpenMPI semble être un projet encore un peu "jeune". Le seul article sur OpenMPI [11] n'est pas assez précis pour savoir comment s'effectue le passage d'un réseau à un autre. D'une manière générale, il donne juste de grandes lignes de développement et les idées générales du projet.

4.3 Les architectures basées sur MPICH

4.3.1 MPICH : Une implémentation de base

L'implémentation MPICH [15] a été concue pour les grappes de calcul constituées d'un réseau homogène. Cependant, elle peut être executée sur une grille si l'environnement est

homogène et peut être vu comme une grappe. Mais surtout, elle sert de support de base pour un grand nombre d'implémentations dédiées aux grilles, je vais donc la décrire rapidement ici.

La force de MPICH est de possèder une architecture en couches, ce qui la rend plus facilement adaptable à différentes technologies réseaux. Elle est écrite en C et se compose de quatres couches distinctes :

- La première couche correspond à l'écriture de l'interface du standard MPI, point de jointure entre les applications et l'implémentation.
- La seconde couche implémente les primitives de communications collectives
- La couche suivante est l'ADI (Application Device Interface). Elle implémente les communications point à point.
- Une dernière couche permet de faire l'interface avec les réseaux sous-jacents. Cette partie, la seule à réécrire lors d'un changement de technologie réseau, a fait l'objet de nombreuses recherches et optimisations comme nous le verrons par la suite.

Une nouvelle version MPICH-2 [19] apporte les fonctionnalités liées au standard MPI2.

4.3.2 MPICH-G2

MPICH-G2 [25] dérive de MPICH-G et se base sur Globus [10] qui est une bibliothèque de services pour les grilles.

Lorsque l'application est démarrée, MPICH-G2 sélectionne le meilleur moyen de communication entre deux processus. Si les deux processus se trouvent sur le même cluster et peuvent communiquer via le même réseau, MPICH-G2 choisit l'implémentation MPI associée à ce réseau pour réaliser la communication. Sinon, si les deux processus sont situés sur deux réseaux distincts, MPICH-G2 utilise une conversion vers TCP. Pour déterminer quel est le meilleur moyen de communiquer entre deux processus, MPICH-G2 utilise un système de couleurs décrit en annexe B. MPICH-G2 gère donc l'hétérogénéité des réseaux de la grille mais pas de manière très efficace. Par ailleurs, elle n'adresse pas la problèmatique de la longue distance et impose l'utilisation de Globus, ce qui est relativement contraignant.

4.3.3 MetaMPICH

MetaMPICH [23] est une approche qui a deux buts principaux : cacher l'hétérogénéité au programmeur et aider à l'exploitation des passerelles hétérogènes des grappes de grappe.

Un langage nous permet de décrire les différentes grappes de la grille, les interfaces de communications ainsi que les routes permettant les communications d'une grappe à l'autre. pour le routage (statique), les clusters et les interfaces de communication. Cette implémentation supporte les réseaux IP et ATM (Asynchronous Transfert Mode) pour les communications longue distance.

L'implémentation de MetaMPICH est effectuée sous forme d'une implémentation MPICH standard qui utilise deux implémentations de l'ADI différentes. L'architecture de MetaMpich est présentée en annexe C. MetaMpich utilise des noeuds particuliers, dédiés au routage, qui peuvent communiquer selon deux modes :

- ch_tunnel : permet de distribuer les messages dans le réseau local

- ch_gateway : permet de router les messages vers les processus externes (via IP ou ATM uniquement)

La gestion de l'hétérogénéité s'appuie sur les implémentations propriétaires de MPI et intègre des noeuds "passerelles" pour les communications hétérogènes qui doivent nécessairement transiter via IP ou ATM. Comme pour MPICH-G2, ce mécanisme n'est pas très efficace et le problème de l'optimisation des communications longue distance n'est pas géré. Le fait de traverser deux fois toutes les couches MPI pour transmettre les messages doit être assez peu performant. Il serait préférable d'effectuer le routage à un niveau inférieur.

4.3.4 MPICH-VMI

MPICH-VMI [26] est une autre implémentation de MPI reposant sur MPICH. Cette implémentation est très proche de MPICH-MAD que présentée à la section suivante. La principale différence réside dans le fait que les communications hétérogènes doivent obligatoirement transiter par TCP dans MPICH-VMI alors que MPICH-Madeleine supporte les passrelles "rapides". L'architecture de Mpich-VMI est décrite en annexe E.

Deux mécanismes sont proposés pour améliorer les communications :

- optimisation selon le profil de l'application
- optimisation des opérations collectives

Le premier consiste à optimiser le placement de tâches sur la grille selon le schéma de communication reliant ces tâches entre elles. Pour ce faire, les informations concernant le nombre de tâches, le type de tâches, et le schéma de communication de ces tâches sont regroupés dans une base de données lors de la première exécution de l'application. Lors des exécutions suivantes, le placement des tâches est optimisé à partir des informations extraites de la BD et de la topologie connue de la grille.

La Figure 2 [26] montre que les optimisations du placement, peuvent diminuer le nombre de communications intersites généralement plus coûteuses. Par exemple, si une application possède le schéma de communications montré à gauche de la figure, il peut y avoir deux placements : celui du haut et celui du bas. Le nombre sur les arêtes représente le nombre de communiations entre les deux processus reliés par l'arête. Le placement du bas est optimisé par rapport à celui du haut car il évite au maximum les communications inter-sites. A priori, ce placement optimisé n'est pas encore effectué dans MPICH-VMI, seul le stockage du schéma de communications est implémenté.

Le second mécanisme permet d'améliorer les opérations collectives en n'utilisant qu'une seule fois les liens longue distance lors de l'exécution de ces opérations. L'application va tout d'abord envoyer un gros message vers une première machine de chaque site. Ensuite, c'est celle-ci qui se charge de faire exécuter l'opération à chaque machine, puis retourne le résultat sur la machine source.

Ces deux optimisations permettent d'éviter les communications longue distance, qui sont plus coûteuses. MPICH-VMI propose donc non seulement la gestion des communications hétérogènes (mais forcément par TCP donc pas très efficace) mais aussi intègre des mécanismes permettant de diminuer les communications longue distance (mais seule une partie des optimisation semble être implémentées à ce jour).

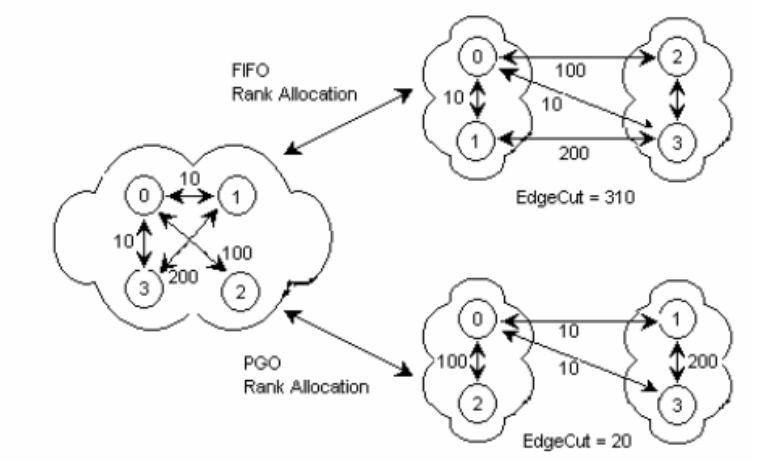


Figure 2: Mécanisme de placement des tâches dans MPICH-VMI

4.3.5 MPICH-Madeleine

MPICH-Madeleine [6] et [24] est une implémentation francaise issue du laboratoire LIP de l'ENS Lyon et du laboratoire LaBRI de Bordeaux (), repose sur l'architecture de MPICH. La Figure 3 présente l'architecture de MPICH-Madeleine qui ajoute le module CH_MAD aux modules CH_SMP et CH_SELF de MPICH. Elle s'appuie également sur trois autres bibliothèques, MADELEINE et MARCEL, et SHARED :

- MADELEINE [2]: bibliothèque de gestion des communications. Elle s'occupe des transferts sur les différents réseaux en choisissant le réseau le mieux approprié lors de chaque communication. C'est choix est effectué grâce à un fichier de configuration qui contient l'ordre de sélection des réseaux. Les réseaux supportés actuellement sont Myrinet (MX et GM), SCI, GBEthernet.
- MARCEL [29]: bibliothèque de processus légers. Elle permet de mettre en place une gestion peu couteuse des threads et un ordonnancement efficace.
- SHARED : bibliothèque de mémoire partagée. Elle s'occupe des échanges intra-noeuds. Ces communications, pour des questions d'efficacité sont effectués en mémoire partagée.

Grâce à Madeleine, la gestion des réseaux hétérogènes est prise en charge et permet l'exploitation de plusieurs réseaux rapides simultanément grâce à un système de passerelles. Chaque passerelle doit possèder deux interfaces, une pour chaque réseau qu'elle interconnecte. L'intérêt de MPICH-Madeleine est de fournir une interface de communication unique -quels que soient les réseaux sous-jacents- permettant de rendre transparente l'hétérogénéité des réseaux constituant la grappe. La topologie du réseau est décrite dans des fichiers de description des grappes et des modes de communication entre ces grappes. Lorsque l'application souhaite communiquer en milieu hétérogène, il est nécessaire de créer un réseau virtuel (décrit dans les fichiers de configuration) qui englobe tous les réseaux que l'on souhaite interconnecter. C'est par ce "canal virtuel" que seront transmises les données à destination d'un réseau différent via une ou plusieurs passerelles. Par exemple dans le cas de la grappe de grappes du site A de la Figure 1, pour pouvoir communiquer entre le réseau 1 et le réseau

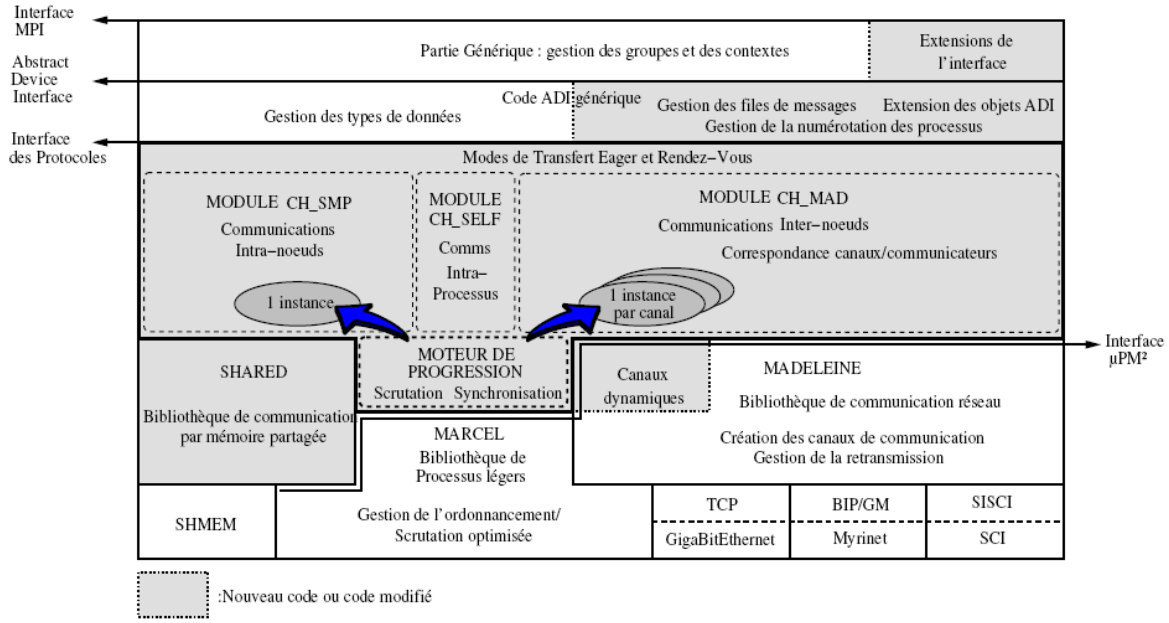


Figure 3: Architecture de MPICH-Madeleine

2, il faut décrire un canal virtuel englobant le réseau Quadrics et le réseau Myrinet. Les communications se font alors via la passerelle 1.

Dans MPICH-Madeleine, les communication sont effectuées selon deux modes de transfert :

- le mode "eager mode" est utilisé pour les messages de taille inférieure à une limite fixée en fonction du réseau. Ce mode de communication envoie tout d'abord un message de contrôle, puis envoie les données à la suite dans d'autres messages.

- le mode "rendez-vous mode" est utilisé pour les longs messages (de taille supérieure au seuil indiqué dans l'alinéa précédant). MPICH-Madeleine envoie tout d'abord un message de controle. La bibliothèque attend ensuite le retour d'un message d'acquittement de la part du destinataire lui indiquant qu'il est prêt à recevoir les données. Elle envoie ensuite les données.

Le grand intérêt de MPICH-Madeleine par rapport à MPICH-VMI, MPICH-G2 ... est de permettre des communications directes entre deux réseaux rapides distincts, c'est à dire sans passer par un réseau intermédiaire tel que TCP. C'est par exemple le cas sur la figure Figure 1 pour des communications entre le réseau Quadrics et Myrinet via la passerelle 1. Cette fonctionnalité est rendue possible grâce à la bibliothèque Madeleine et permet d'obtenir de bien meilleures performances en milieu hétérogène sur grappe. En revanche, l'utilisation de MPICH-Madeleine n'a pas été démontrée sur la grille, aucune mesure de performances n'ayant été réalisées dans ce cadre. En particulier, MPICH-Madeleine ne propose pas d'optimisations des communications longue distance.

4.4 Synthèse

Comme il est assez difficile de comparer les implémentations sur des uniques questions de performances, la Table 2 propose une vue d'ensemble des différentes solutions partielles disponibles dans la littérature pour l'exécution d'applications MPI sur les grilles de calcul. Au vu de ce tableau, il est clair qu'aucune implémentation ne répond entièrement aux contraintes imposées par la grille, à savoir la gestion de l'hétérogénéité et la prise en compte de la longue distance, telle que nous l'avons présentée dans la problématique.

Dans la ligne longue distance du tableau, nous constatons que seules MPICH-VMI et GridMPI proposent des optimisations de ce point, la première par adaptation de TCP aux conditions de la grille, la seconde par optimisation du placement des tâches sur la grille (non opérationnel actuellement) et optimisation des opérations collectives dans MPI.

Dans la ligne hétérogénéité, nous constatons que quatre implémentations sur six gèrent l'hétérogénéité uniquement en utilisant TCP comme réseau intermédiaire ce qui n'est pas très performant. Les deux implémentations restantes proposent un sytème de passerelles. Pour MetaMPICH, ce système de passerelles impose de traverser deux fois toutes les couches MPI, il est donc peu efficace. Dans MPICH-Madeleine en revanche cette gestion des passerelles est optimisée. Madeleine permet en particulier des communications directes entre deux réseaux rapides différents, Myrinet et Infiniband par exemple.

Après ce constat, il est possible de choisir soit une implémentation optimisée pour la longue distance et de lui ajouter la gestion de l'hétérogénéité; soit de faire les choix inverse.

Cependant, il semble plus aisé de mettre en place la gestion de la longue distance par exemple en ajoutant un module dédié à ces communications ou en dérivant le module TCP existant, que d'ajouter la gestion de l'hétérogénéité à GridMPI ou MPICH-VMI dont l'architecture n'est pas conçue dans cette optique. Notre choix s'est donc porté sur MPICH-Madeleine qui est l'application la plus performante pour la gestion de l'hétérogénéité. J'ajouterais que cette implémentation n'est pas trop lourde à mettre en place comme pourrait l'être MPICH-G2 par exemple, qui impose l'installation et l'utilisation de Globus. MPICH-Madelein à l'avantage d'être développé et maintenu par une équipe de recherche avec laquelle nous avons pu établir des contacts.

5 Contributions

Nous présentons dans cette section les contributions de notre travail qui consiste en quelques développements logiciels et propositions d'optimisations de MPICH-Madeleine pour la grille en s'appuyant sur une succession d'expérimentations réelles sur la plateforme Grid'5000. Ces différentes mesures nous ont conduit à un certain nombre de conclusions.

5.1 Démarche entreprise

Pour faire fonctionner des applications MPI sur une grille de calcul, les différentes implémentations doivent prendre en compte deux problèmes principaux : l'hétérogénéité des réseaux interconnectant les différentes grappes de la grille et la longue distance qui sépare les différents sites.

Parmi toutes les implémentations présentées à la section 3, aucune ne répond entièrement à notre problématique, présentée à la section 2 : permettre à des applications MPI de s'éxécuter efficacement sur une grille de calcul composée de réseaux rapides hétérogènes.

Implémentation	**MPICH-G2 [25]**	**GridMPI [13]**	**MPICH-MAD [24]**	**MPICH-VMI [26]**	**Meta-MPICH [23]**	**OpenMPI [11]**
Avantages	Beaucoup utilisé	Adaptation de TCP pour de meilleures com. longue distance	Support efficace de l'hétérogénéité	Optimisation du placement des tâches sur la grille	Support de l'hétérogénéité	Fournit un ens. de services suppl. (tolérance aux pannes...)
Inconvénients	Utilisation lourde de Globus, installation difficile	Très peu documenté	Com. TCP non optimisées, non testé sur la grille	Traitements supplémentaires lents	Routage effectué à haut niveau donc assez lent	Implé. lente, peu de documentation, manque de maturité
Hétérogénéité	Via TCP	Via TCP	Via passerelles	Via TCP	Via passerelles	Via TCP
Longue distance	Non optimisé	Optimisé	Non optimisé	Opérations collectives optimisées, placement des processus	Non optimisé	Non optimisé
Dernière publication	2003	2005	2004	2004	2003	2005

Table 2: Comparaison des différentes implémentations

Seule GridMPI permet d'optimiser partiellement les communications longue distance. Les autres implémentations gèrent de manière plus ou moins performantes l'hétérogénéité des communications .

Nous avons donc choisi de partir de MPICH-Madeleine comme implémentation de base car elle présentes les meilleures performances dans le contexte des grappes de grappes, en particulier grâce à ses passerelles rapides. En revanche, le comportement de MPICH-Madeleine sur une grille reste inconnu : les performances sur le longue distance n'ont pas été testées ni prouvées.

Nous allons donc évaluer les performances de MPICH-Madeleine sur la gille Grid'5000 et essayer d'en extraire des conclusion relatives aux contraintes imposées par le réseau longue distance. Nous tenterons ensuite de proposer des optimisations permettant d'améliorer les performances. Nous regarderons par exemple si les optimisations proposées dans GridMPI peuvent s'adapter à MPICH-Madeleine. Nous commençons par présenter dans la section suivante les outils et la plateforme utilisés pour réaliser nos expériences.

5.2 Contexte d'expérimentation

5.2.1 Outils d'évaluation des performances

Pour comparer les différentes implémentations, on peut soit se baser sur les services fournis ou les fonctionnalités remplies, soit comparer leurs performances intrinsèques.

Afin de comparer les différentes implémentations de MPI, il est nécessaire d'exécuter des expérimentations identiques sur une même plateforme et donc d'utiliser une ou plusieurs applications de référence (*benchmarks*).

Dans le cas des applications parallèles, il s'agit typiquement d'un ping-pong entre deux machines ou de l'exécution d'applications "étalons" permettant d'évaluer de manière identique les différentes implémentations et leur paramètrage.

PingPong Une implémentation efficace devrait tenir compte de ces deux problématiques conjointement.

Il s'agit de mesurer le temps mis par un message (dont on fait varier la taille) pour effectuer un aller-retour entre deux machines. A partir du temps d'un aller-retour, il est alors possible de calculer pour chaque taille de message, la latence (temps de transfert) ou la bande passante en Mb/s. En faisant la moyenne sur un grand nombre d'échanges pour la même taille de messages, le ping-pong permet ainsi d'obtenir les performances brutes d'une couche de communications. Nous évaluerons ainsi les performances aussi bien de MPICH-Madeleine que de Madeleine, TCP ou MX.

NAS Parallel Benchmarks Les NPB (NAS Parallel Benchmarks [3]) sont un ensemble de programmes qui fonctionnent sur des architectures parallèles. Ils ont été écrits par la *NASA Advanced Supercomputing facility* et servent pour les mesures de performances des machines parallèles. Ils sont également utilisés pour tester les différentes implémentations de MPI. Les NPB-2 sont écrits avec la bibliothèque MPI. Cette version contient 8 programmes de résolutions de calculs effectués en parallèle. L'ensemble de ces programmes donne un panel représentatif des applications qui peuvent s'exécuter sur une grappe : une des application communique avec de gros messages mais peu souvent, une autre communique de manière intensive...

Il existe même une version spécifique des NPB pour les grilles qui comprend 4 applications, dérivées des NPB-2 et des NPB-3 et qui symbolisent les différents types d'applications exécutées sur les grilles. Malheureusement, cette version n'est pas disponible pour MPI. Nous utiliserons les NPB pour voir si les optimisations que nous proposerons ont un impact sur les temps d'exécution de véritables applications parallèles.

5.2.2 La plateforme expérimentale et ses contraintes : Grid'5000

Tous nos tests ont été effectués sur la plateforme Grid'5000 que nous avons présenté à la section 2.1.3. Pour réaliser mes tests, je me suis servit de la grille aussi bien comme une grappe en n'utilisant qu'un seul site ou comme une grille "réelle" en utilisant deux sites distants. Je vais commencer par présenter l'architecture de la grille et les réseaux qui la connectent puis je présenterai succintement la manière de lancer une application sur la grille. Enfin, j'expliciterai les difficultés d'utilisation de celle-ci.

Le réseau Grid'5000 Cette grille nationale, dédiée à la recherche, devrait à terme interconnecter 5000 processeurs répartis sur 9 sites. Les différents sites de Grid'5000 sont interconnectés via le réseau RENATER3 et le réseau RENATER4 [30]. Les machines qui composent cette grille ont des architectures similaires mais les réseaux rapides qui interconnectent les grappes sont hétérogènes. La Figure 4 montre une vue d'ensemble de cette grille. Les différents sites sont interconnectés soit par RENATER3 (liens sombres partagés avec le réseau Internet, soit par RENATER4 (liens blancs dédiés au traffic Grid'5000). Les capacités de ces liens varient de 1Gb/s à 10 Gb/s. Cette même figure indique également les différents réseaux rapides présents sur chacun des sites.

Lancement d'une expérience sur Grid'5000 La plateforme Grid'5000 est partagée par plusieurs équipes de recherche françaises qui ont mis en commun un ensemble de ressources constituant la grille. Elle est accessible par plusieurs centaines d'utilisateurs qui peuvent lancer des applications sur la grille (ou une partie) simultanément. La politique d'utilisation actuelle de la plateforme est la suivante : avant de lancer son application, un utilisateur doit :

- réserver des noeuds sur les différents sites qu'il souhaite utiliser : l'utilisateur programme ses réservations en fonction des noeuds restants disponibles ; le critère d'attribution est "premier arrivé, premier servi"
- éventuellement déployer son propre environnement : un utilisateur peut créer sa propre image système Linux qui va être installée sur une des partitions. Cela permet également de pouvoir installer ses propres logiciels et être superutilisateur sur ses machines
- lancer son application dans le créneau de temps imparti par la réservation

Nous avons du déployer notre propre environnement car les logiciels disponibles dans l'environnement de base n'étaient pas suffisants pour mener à bien nos expérimentations. Par ailleurs, dans certains cas, il était nécessaire d'avoir les droits superutilisateur pour modifier certains paramètres du système d'exploitation.

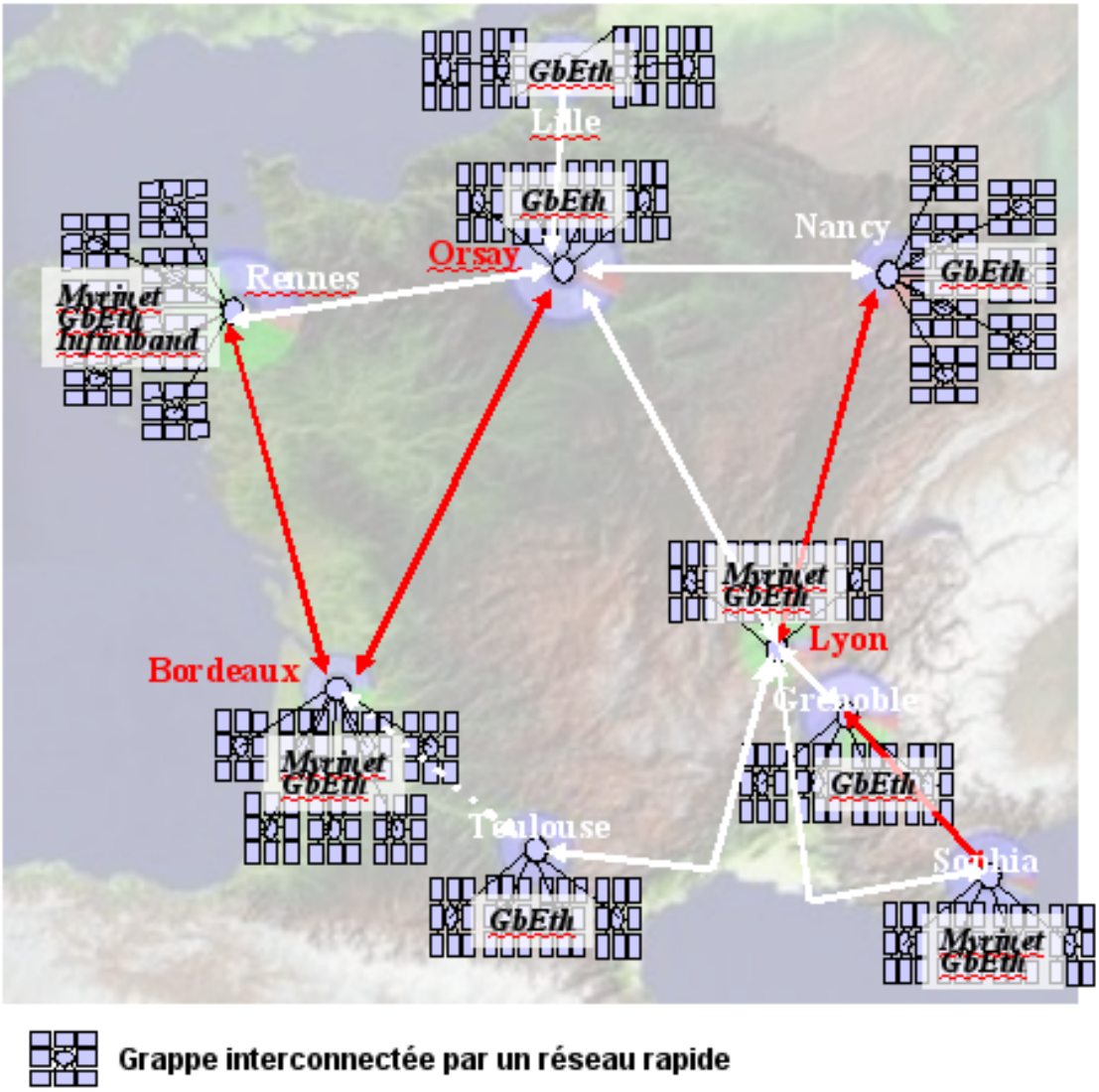


Figure 4: Vue d'ensemble de Grid5000

Difficultés d'utilisation La grille étant partagée entre différents utilisateurs, il n'est pas toujours aisé de pouvoir réserver les ressources souhaitées. Pour pouvoir réaliser des tests dans des conditions optimales, il est parfois nécessaire de réserver l'ensemble des machines de deux sites distincts surtout pour les tests longue distance. Sinon, et même si le lien est dédié à Grid5000 (sur RENATER4), le lien reste partagé entre tous les utilisateurs des deux sites séparés par ce lien. Nous avons commencé nos expériences sur des liens non dédiés, où la bande passante est partagée avec l'ensemble du trafic de RENATER. De ce fait, les résultats étaient assez aléatoires selon l'heure de la journée et l'utilisation du réseau. De plus, il est rarement possible et déconseillé de réserver deux sites complets en journée. Donc, pour avoir des résultats probants, il est parfois nécessaire d'effectuer les tests la nuit. Cela permet de limiter l'utilisation du lien intersites par du trafic parasite mais ne résoud cependant que partiellement le problème.

Comme nous l'avons vu plus haut, le déploiement nécessite la création d'une image d'un système linux. Or, cette image est différente pour chaque site du fait de l'hétérogénéité des machines et des réseaux utilisés. Nous avons donc du construire autant d'images que de plateformes cibles. Le déploiement n'étant pas possible sur tous les sites, il a fallu nous adapter et faire les réservations en conséquence. Par ailleurs, le changement de paramètres entre deux expériences nécessite la recompilation des environnements logiciels sur chaque site.

De plus, comme cette grille expérimentale est en permanente évolution, il arrive souvent que les grappes soient mises à jour et donc indisponibles. Par exemple, nous aurions souhaité réaliser nos expériences entre les sites de Sophia et Lille afin de bénéficier de liens dédiés sur RENATER4, mais cela n'a pas été possible du fait de l'indisponibilité du site de Lille. Nous avons donc été obligés d'utiliser un lien partagé entre Bordeaux et Sophia. Enfin, il est difficile comme sur toute plateforme expérimentale réelle et partagée d'obtenir des résultats reproductibles en lançant plusieurs fois la même expérience.

Notre travail a donc été compliqué par les difficultés d'utilisation de Grid'5000 et les difficultés d'analyse des résultats des expériences menées. Nous allons maintenant décrire les expériences réalisées.

5.2.3 Description des expériences

Les différentes mesures ont été effectuées à la fois sur un même cluster et à la fois sur des machines de grappes différentes séparées par le WAN. Par la suite, je parlerai de mesures locales dans le premier cas et de mesures longues distances lorsqu'elles sont multi-sites.

La figure Figure 5 montre l'environnement que j'ai utilisé pour réaliser mes expériences. Celles-ci ont été réalisées soit entre Bordeaux et Sophia soit sur le site de Sophia uniquement pour les expériences locales.

Les noeuds de Bordeaux sont des AMD Opteron 248, 2,2 GHz équipés de cartes 1Gb Ethernet et de cartes Myrinet MX. Les noeuds de Sophia sont des AMD Opteron 246 2.0GHz équipés également de cartes 1Gb Ethernet et Myrinet MX. Les machines d'un même site peuvent donc communiquer soit via le réseau Ethernet, soit via le réseau Myrinet de la grappe.

J'ai tout d'abord effectué un ping-pong avec une application MPI décrite dans le paragraphe 5.5. Puis, au vu de ces résultats, j'ai réalisé un ping-pong Madeleine, et un ping-pong TCP. Enfin, pour voir le comportement de MPICH-Madeleine avec des applications conséquentes, j'ai également exécuté les NPB, à la fois sur le site de Sophia et sur la grille.

Les mesures des ping-pong ont été effectuées soit entre les noeuds (3) et (4), soit entre

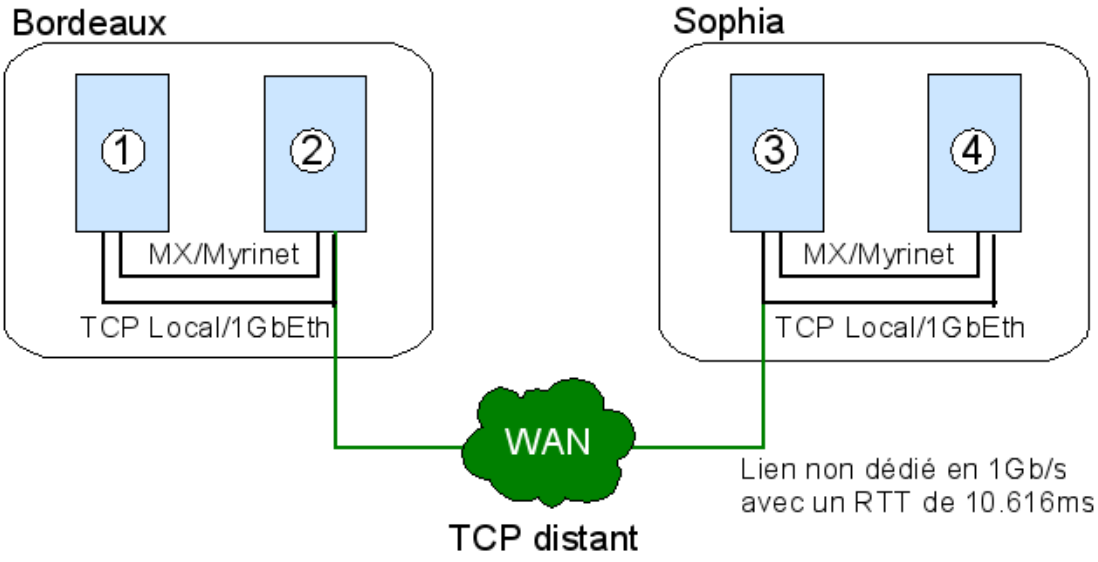


Figure 5: Environnement d'expérimentation

les noeuds (1) et (4), soit entre les noeuds (2) et (3). Pour les mesures locales entre (3) et (4), les communications s'effectuent soit via TCP sur Ethernet, soit via MX sur Myrinet (notés respectivement *MX local* et *TCP local*). Entre (1) et (4), les communications transitent soit par MX, soit par TCP localement et par TCP entre les deux sites. Nous noterons par la suite respectivement *MX-TCP-MX distant* et *TCP-TCP-TCP distant* selon le cas. Entre (2) et (3), les processus communiquent directement via le réseau TCP (*noté TCP distant*).

Les courbes que nous présenterons pour les résultats des ping-pong sont des courbes de latence (en us) et de bande passante (en Mb/s) pour lesquelles nous faisons varier en abscisse la taille des messages. Les courbes de latence permettent de visualiser les performances liées aux petits messages et les courbes de bande passante le comportement de l'application avec de gros messages en atteignant la bande passante maximale disponible au niveau de l'application.

5.3 Performances brutes de MPICH-Madeleine

Le but de cette première série de mesures est de connaître les performances de MPICH-Madeleine et son comportement selon le type de la couche de communication utilisée : MX local, TCP local, TCP distant. Nous allons donc faire ces tests en local (sur une grappe) ou en distant (sur la grille).

5.3.1 Performances sur une grappe

Les mesures ont été effectuées sur la grappe de Sophia. MPICH-Madeleine utilisait soit le réseau TCP, soit le réseau MX en local. La Figure 6 montre les résultats obtenus. Comme escompté, les performances avec MX sont naturellement meilleures que celles sur TCP. On

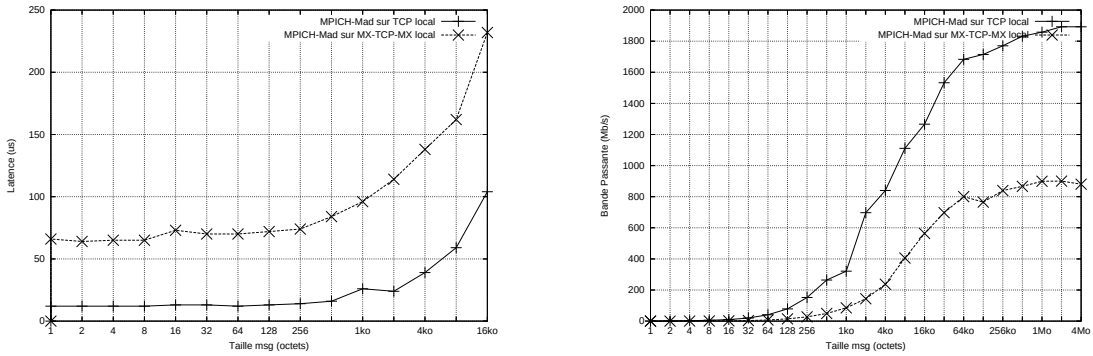


Figure 6: Courbes de latence et bande passante de MPICH-Mad en local sur MX et TCP

obtient une latence de 5us sur MX et 60us sur TCP et une bande passante maximale de 1900 Mb/s sur MX et 900Mb/s sur TCP.

Ces résultats sont des résultats "référence" qui nous servirons d'étalon lors de nos mesures sur la grille.

5.3.2 Performances sur la grille

Pour évaluer les performances sur la grille, nous réalisons les mêmes courbes qu'à la section précédente entre les noeuds (1) et (4) de la Figure 5. Celles-ci sont présentées Figure 7.

Les conclusions sont les suivantes :

- La chute observée sur la courbe en latence entre 1ko et 2ko est anormale. Nous reviendrons dessus par la suite.
- La différence de latence entre TCP-TCP-TCP (56ms) et MX-TCP-MX (55ms) est relativement faible : cela n'est pas surprenant car le surcoût induit par le lien intersite est prépondérant.
- La différence de latence entre les mesures locales (de l'ordre de la dizaine de microsecondes cf Figure 6) et les mesures distantes (de l'ordre de la dizaine de millisecondes) est significative. Là encore l'impact du lien longue distance est important.
- La différence de débit entre TCP-TCP-TCP (75Mb/s) et MX-TCP-MX (75Mb/s) est inexistante. Le plus important est surtout que le débit obtenu est très inférieur non seulement aux résultats obtenus localement mais aussi au débit théorique du lien longue distance.

Ces résultats montrent bien l'impact néfaste du réseau longue distance sur les performances de MPICH-Madeleine. Nous constatons également que l'impact du réseau local (MX ou TCP) n'a que très peu d'influence sur les performances de MPICH-Madeleine sur la grille. Nous allons donc désormais nous concentrer sur l'étude des performances liées au lien longue distance.

La Figure 8 montre l'impact de la traversée des passerelles MPICH-Madeleine (noeuds (2) et (3) sur la figure Figure 5). Pour ce faire, nous avons mesuré les performances de

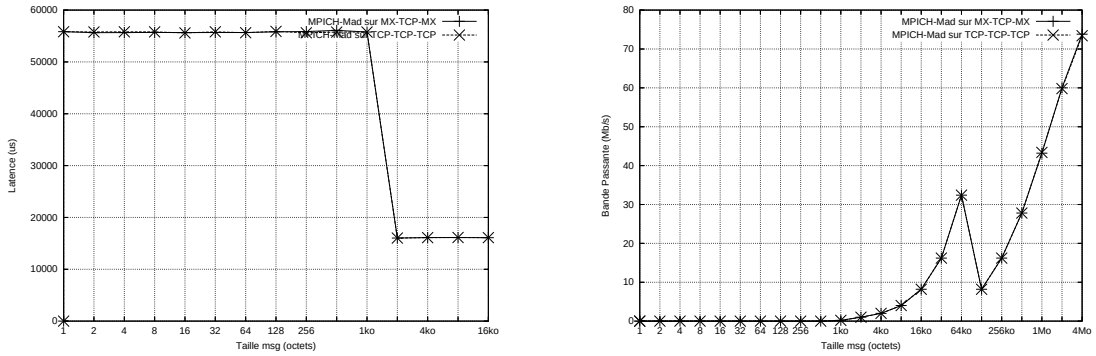


Figure 7: Courbes de latence et bande passante de MPICH-Mad en distant sur MX-TCP-MX et TCP-TCP-TCP

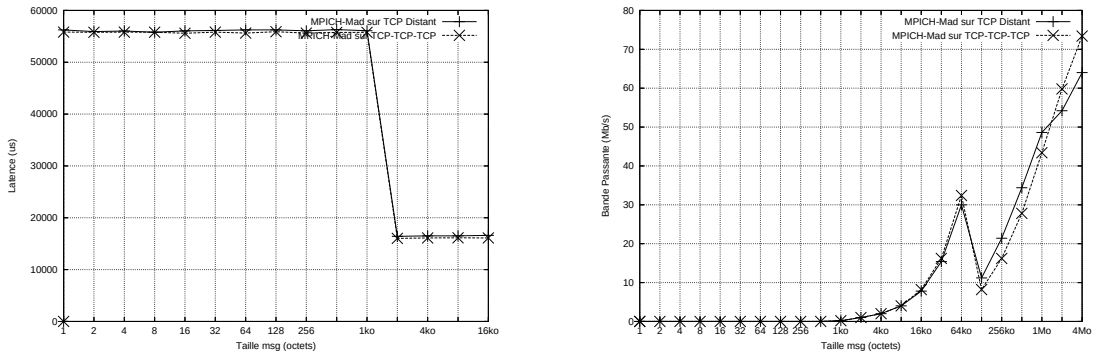


Figure 8: Impact de la traversée des passerelles MPICH-Madeleine

MPICH-Madeleine sur TCP-TCP-TCP distant (entre (1) et (4)) et TCP distant (entre (2) et (3)). Les deux courbes étant quasiement superposées aussi bien en latence qu'en débit, nous pouvons conclure que la traversée des passerelles n'influence que très peu les résultats. Par la suite, nous effectuerons nos mesures entre les deux passerelles uniquement.

La Figure 9 montre l'impact de l'utilisation des canaux virtuels dans la bibliothèque Madeleine et nous permet d'expliquer d'où provient la chute de latence observée sur la Figure 7. Les canaux virtuels (cf. 4.3.5) permettent les communications entre des réseaux hétérogènes. Ils sont par exemple obligatoires pour le passage du réseau MX au réseau TCP. En réalisant des mesures uniquement via TCP avec ou sans canal virtuel, nous constatons que la chute de latence observée disparaît si les canaux virtuels ne sont pas utilisés. Comme dans la suite de nos tests, nous nous limitons à des communications longue distance via TCP, nous n'utiliserons plus les canaux virtuels.

5.3.3 Conclusions

Par les mesures précédentes, nous arrivons aux constatations suivantes :

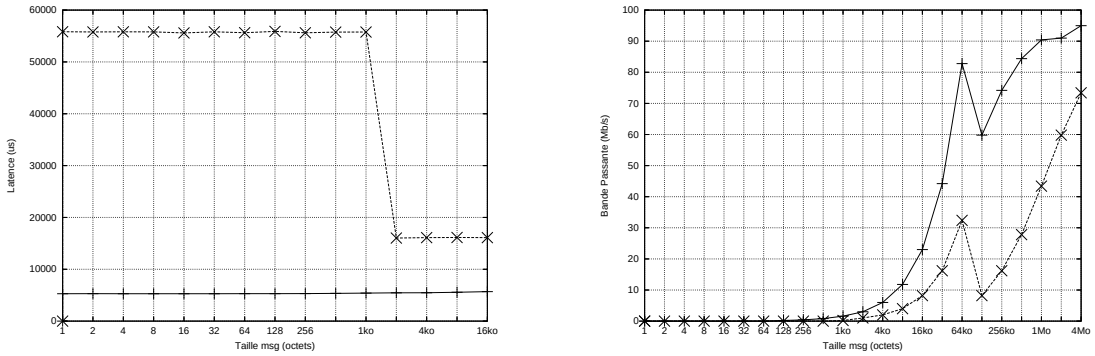


Figure 9: Impact des canaux virtuels sur le longue distance

- La traversée des passerelles Madeleine n'impacte pas les résultats sur les liens longue distance
- L'utilisation des canaux virtuels de Madeleine engendre une chute de performances importante
- Les performances de MPICH-Madeleine sur la grille, c'est à dire dès lors qu'un lien longue distance est traversé, sont très inférieures à celles observées sur une grappe.
- Comme les performances longue distance sont nettement inférieures aux performances locales, le gain potentiel pouvant être obtenu par l'utilisation d'un réseau rapide (par exemple MX sur Myrinet) n'est pas significatif.

Désormais, nos expérimentations seront canalisées sur l'étude et l'optimisation des communications longue distance. De ce fait, il n'est plus nécessaire d'utiliser MX et donc nous pouvons nous affranchir de l'utilisation des canaux virtuels et des passerelles Madeleine.

5.4 Optimisations des performances pour la grille

Nous allons dans cette section commencer pas essayer d'analyser plus finement les mauvais résultats de MPICH-Madeleine sur la grille. Pour ce faire, nous allons étudier les performances au niveau de chacune des couches de communication.

5.4.1 Etude fine des performances de chaque couche

Les différentes couches de communication intervenant dans le ping-pong MPICH-Madeleine entre les noeuds (2) et (3) sont MPICH-Madeleine, Madeleine, TCP/IP et GigabitEternet.Les mauvaises performances obtenues peuvent donc être liées à l'une ou l'autre de ces couches. Nous avons donc exécuté un ping-pong au niveau de chaque couche en allant de la plus haute à la plus basse afin de déterminer quelle est celle (voire celles) qui pose problème dans les communications longue distance :

- MPICH-Madeleine : les mesures sont réalisées avec le pingpong MPI présenté dans la section 5.5.

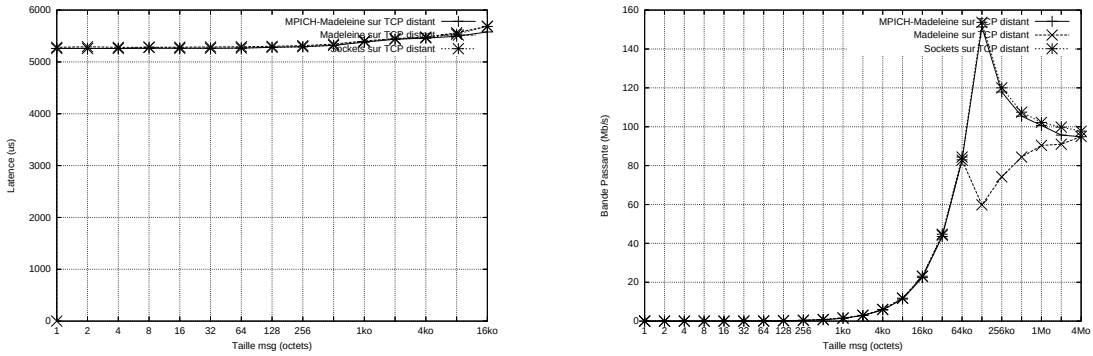


Figure 10: Performances de chaque couche sur la longue distance

- Madeleine : Nous avons utilisé le pingpong fourni avec Madeleine
- TCP : Nous avons réalisé notre propre ping-pong, que nous avons validé avec des utilitaires de mesures réseaux tel que netperf ou iperf, toujours dans la section 5.5.

La Figure 10 nous montre les résultats obtenus. Les conclusions sont les suivantes :

- Les performances de Madeleine sont très proches de celles de TCP. Ces performances sont très faibles avec une bande passante maximale de 153 Mb/s alors que le débit théorique est de 1Gb/s, et une chute de débit étonnante à partir de 128Ko.
- Les performances de MPICH-Madeleine sont très faibles globalement. La bande passante maximale obtenue est de l'ordre de 90Mb/s.
- Les performances de MPICH-Madeleine ne sont pas du tout du même style que celles de TCP et Madeleine, ce qui est surprenant vu qu'elle s'appuie sur ces deux autres couches.

Les résultats montrent que TCP est très peu performant. Ceci est génant car il s'agit de la brique la plus basse de notre architecture. Il apparait sur la figure un seuil pour TCP à 128 Ko que nous expliquerons dans la section suivante. Nous allons donc essayer d'améliorer les performances de TCP afin d'en faire bénéficier Madeleine et MPICH-Madeleine.

5.4.2 Paramètrage de TCP pour la longue distance

Au vu des résultats précédents, il semble judicieux de chercher à améliorer les performances au niveau TCP. La chute soudaine de débit à 128 Ko constatée à la section précédente sur la courbe Sockets sur TCP distant mérite également d'être expliquée. Après investigations, il se trouve que cette taille de 128 Ko correspond à la taille maximale par défaut du *socket buffer* (tampon d'émission ou de réception des données au niveau TCP) dans le noyau Linux. Nous avons donc décidé de changer la valeur de ce paramètre afin d'en voir l'impact sur les performances.

La taille par défaut des tampons TCP dans les systèmes d'exploitation est prévue pour des réseaux lents à faible latence. Il est maintenant connu [20] que la taille des tampons

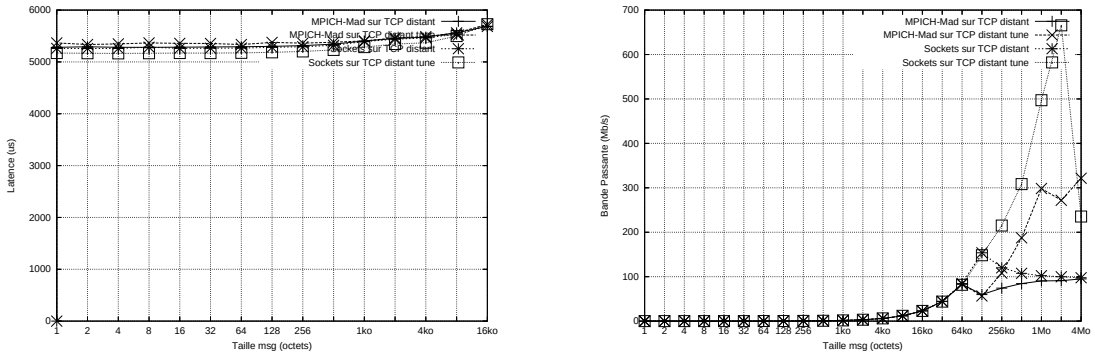


Figure 11: Impact du paramètrage de TCP sur les performances de MPICH-Madeleine

d'émission/réception d'une connexion TCP doit être supérieure au produit du temps aller-retour (RTT) séparant la source de la destination par le débit théorique du lien, et ce afin de pouvoir profiter pleinement des performances du réseau. Ceci permet de ne pas être bloqué par la dimension du tampon et de s'assurer qu'il aura potentiellement toujours plus de données à envoyer que le lien ne peut en écouler. Dans notre cas, entre les sites de Sophia et de Bordeaux, nous avons un RTT d'environ 11ms et un débit théorique de 1Gb/s. Nous avons donc décidé d'augmenter la taille par défaut des tampons TCP et de la fixer à 4 Mo. Nous noterons désormais *TCP tuné* la couche TCP ainsi paramètrée.

Pour que cette modification soit prise en compte, il a été nécessaire non seulement de modifier le module de communication TCP de Madeleine ainsi que le ping-pong TCP mais aussi le système d'exploitation pour modifier la valeur du paramètre dans le noyau Linux.

La Figure 11 montre les performances des ping-pong MPICH-Madeleine et Sockets sur le TCP tuné et le TCP non tuné. Nous constatons que le *tuning* de TCP améliore très nettement les performances de MPICH-Madeleine et des Sockets TCP. La bande passante maximale passe de 150Mb/s à plus de 650Mb/s sur TCP et de 95Mb/s à 320Mb/s sur MPICH-Madeleine. Malgré tout, ces résultats ne sont pas encore pleinement satisfaisants puisque le débit maximal atteint avec MPICH-Madeleine est encore très loin de celui obtenu entre deux machines d'une même grappe (900 Mb/s).

5.4.3 Paramètrage de MPICH-Madeleine pour la longue distance

Nous souhaitons maintenant étudier l'influence de certains paramètrages de MPICH-Madeleine permettant d'impacter positivement les résultats obtenus sur la grille. Il s'agit d'une part d'allouer plus de mémoire pour les communications en attente (dites inattendues) dans les couches basses de MPICH-Madeleine et d'autre part de jouer sur la valeur des seuils séparants les différents modes d'envoi utilisés par MPICH-Madeleine.

Augmentation de la mémoire disponible pour les messages inattendus de MPICH-Madeleine Pour essayer d'améliorer les performances de MPICH-Madeleine, sur les conseils d'un de ses concepteurs, nous avons modifié la taille globale disponible pour stocker les messages inattendus de MPICH-Madeleine. Ces zones de mémoire intermédiaires sont allouées statiquement et peuvent être utilisées à tout moment pour stocker les communications MPI non

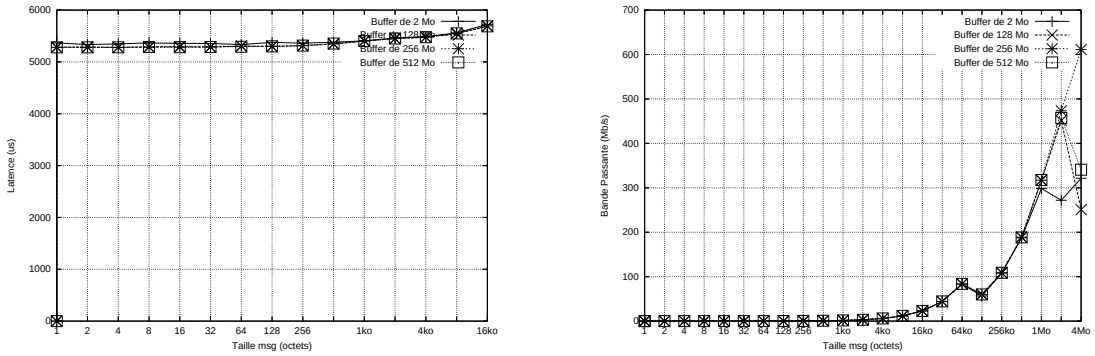


Figure 12: Influence de la taille mémoire disponible sur les performances de MPICH-Madeleine en TCP distant tuné

achevées. En augmentant cette taille, nous pensons ainsi éviter des allocations dynamiques coûteuses en temps. Il est naturel d'augmenter cette zone de mémoire intermédiaire dans le cas de la longue distance car les aller-retours entre la source et la destination étant plus longs du fait de la latence élevée du lien inter-sites, cette zone intermédiaire de mémoire pour les communications en attente a tendance à être pleine plus rapidemment.

La Figure 12 montre les résultats obtenus. Initialement, la taille de mémoire disponible était de 2 Mo. Nous avons fait des tests avec des tailles de 128 Mo, 256 Mo et 512 Mo. La taille de 256 Mo semble correspondre au meilleur compromis : les résultats en débit sont moins bons pour 512 Mo que pour 256 Mo, sans doute car nous consommons alors trop de ressources mémoire relativement à la quantité de mémoire physique disponible pour les applications. Dans tous les cas, cette optimisation fixant la taille à 256 Mo permet encore une nette amélioration de la bande passante maximale disponible pour MPICH-Madeleine puisque nous atteignons désormais 600Mb/s au lieu des 320Mb/s obtenus précédemment.

Modification du seuil de MPICH-Madeleine séparant les transferts en mode *Eager* des transferts en mode *Rendez-vous* Dans MPICH-Madeleine, la façon d'envoyer les messages change selon la taille des messages comme nous l'avons expliqué brièvement à la section 4.3.5. Il y a donc un seuil optimal, dépendant essentiellement de la latence du réseau utilisé, séparant ces deux modes de transfert. L'idée générale de l'existance de ce seuil est la suivante : pour les gros messages, la bibliothèque MPI préfère s'assurer que le récepteur a posté une requête de réception correspondant au message envoyé avant que les données ne quittent effectivement l'émetteur. En effet, sinon la bibliothèque MPI serait obligée de stocker temporairement les données sur le récepteur (induisant ainsi une recopie supplémentaire des données) en attendant que la requête de réception arrive. Pour les gros messages, il est donc préférable de mettre en place un protocole de rendez-vous au niveau MPI avant d'envoyer effectivement les données. Inversement, pour les messages de petites tailles, il est plus performant d'utiliser une recopie intermédiaire côté récepteur si la requête n'a pas été postée. Nous pensons que le seuil optimal séparant les deux modes ne peut être le même sur le lien longue distance et sur un réseau rapide de grappe. En effet, le coût du rendez-vous entre l'émetteur et le récepteur dépend très fortement du temps mis par un message pour aller de l'émetteur au récepteur (et inversement) : de l'ordre de la dizaine de

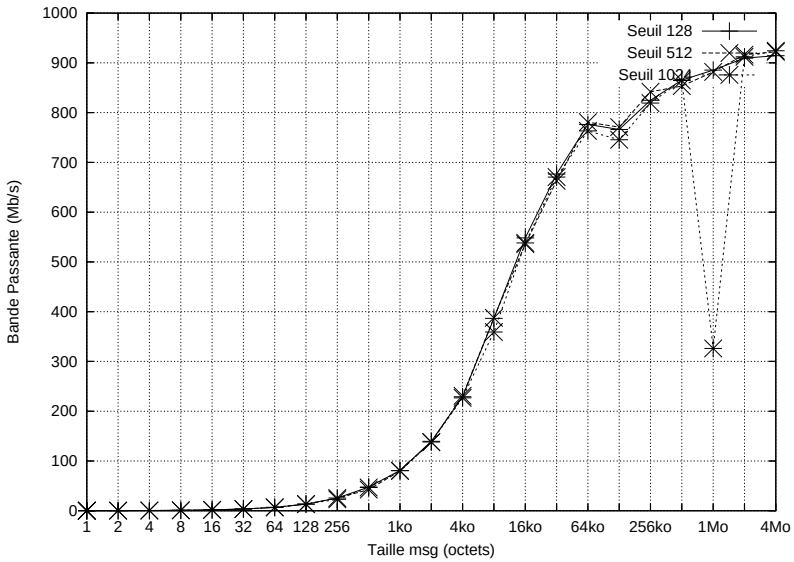


Figure 13: Influence du seuil séparant les modes Eager et Rendez-vous de MPICH-Madeleine sur TCP Local

micro-secondes sur une grappe possédant un réseau rapide et de l'ordre de la dizaine de milliseconde sur un lien longue distance. Il serait donc naturel d'augmenter significativement ce seuil dans le cas des communications sur la grille.

Initialement, ce seuil était fixé à 128 Ko pour les transferts de MPICH-Madeleine sur TCP. Nous avons fait varier ce seuil sur la Figure 13 : 128 Ko, 512 Ko et 1024 Ko. Nous ne représentons pas les résultats de la courbe en latence puisque la valeur du seuil n'a pas d'influence sur les messages de petites tailles (inférieures à la valeur seuil). Les résultats que nous obtenons ne sont pas probants : le saut entre 64 Ko et 128 Ko perdure quand nous changeons la valeur du seuil et les sauts introduits (respectivement à 512 Ko et 1 Mo) par le changement de seuil donnent un débit légérement inférieur à la valeur antérieure. Nous avons essayé sans succès de trouver une explication à ce double phénomène. Nous sommes là encore en relation avec les concepteurs de MPICH-Madeleine et Madeleine. Quoi qu'il en soit, il nous semble que le mode rendez-vous n'est pas utilisé sur la courbe intiale (celle avec le seuil à 128 Ko) et que le décrochement se produisant entre 64 Ko et 128 Ko n'est pas lié à un paramètre de MPICH-Madeleine. Comme nous ne voyons pas ce saut sur la courbe en débit au niveau de TCP, peut-être qu'il provient d'un paramètre de la bibliothèque Madeleine ?

5.4.4 PsPacer : une optimisation issue de l'implémentation GridMPI

L'implémentation de GridMPI propose une optimisation nommée *PsPacer* sous forme d'un module noyau Linux modifiant le comportement de TCP. L'idée générale est d'adapter le comportement de TCP aux flux spécifiques des applications MPI dans le contexte des grilles qui est par nature totalement différent de celui d'Internet. PsPacer met en place un mécan-

isme qui ralentit le flux TCP généré par une application MPI afin d'éviter les rafales qui sont courantes dans ce type d'applications et ainsi de faire du contrôle de congestion préventif. Pour ce faire, il introduit des paquets IP "PAUSE" qui ne sont pas routés sur le réseau afin d'espacer dans le temps la transmission des véritables paquets IP de l'application.

Nous avons pensé qu'il pouvait être intéressant d'incorporer ces optimisations à MPICH-Madeleine et de voir si cela aurait un impact sur les résultats obtenus. Nous avons donc mis en place PsPacer sur Grid'5000 et testé si cela avait une influence sur les résultats obtenus avec le ping-pong MPICH-Madeleine sur TCP distant. Nous avons fait ces tests en journée sur un réseau non dédié, au moment où les congestions sont les plus susceptibles de se produire. Néanmoins, nous n'avons constaté aucune amélioration significative. Cela ne nous surprend pas tellement car les communications provoquées par l'exécution d'un ping-pong ne génèrent pas de trafic en rafale et ne sont pas susceptibles de créer des congestions.

Nous allons très prochainement refaire des expériences avec et sans PsPacer en utilisant des applications plus représentatives des applications réelles en espérant que les résultats seront plus concluants.

5.4.5 Validation des optimisations à l'aide des NPB

Les exécutions successives des ping-pong sur la grille nous ont montré que nos optimisations amélioraient sensiblement les performances de MPICH-Madeleine avec l'application ping-pong. Cependant, cette application n'étant pas représentative des applications parallèles de grille, nous souhaitons mesurer l'impact de nos optimisations sur de véritables applications. Nous pourrons ainsi connaître le comportement de MPICH-Madeleine face à des applications qui communiquent autrement que le simple ping-pong. Nous avons donc choisi d'exécuter les NPB sur la grille en utilisant le TCP "normal" et notre TCP tuné. Les NPB sont une suite d'applications MPI qui possèdent chacune leur propre schéma de communications. Certaines communiquent beaucoup avec de petits messages, d'autres seulement avec de gros messages...

L'environnement d'exécution était semblable à celui décrit à la section 5.2.3, excepté le fait que nous avons réservé 8 noeuds sur chacun des sites (Sophia et Bordeaux). En effet, chacune des applications issue des NPB s'exécute sur 16 processus en parallèle.

La Figure 14 montre le comparatif entre le temps d'exécution des NPB sur TCP et sur TCP tuné. Le temps sur TCP non tuné est pris comme valeur de référence. CG, EP, IS, LU et MG représentent chacun une des applications des NPB.

Nous constatons que les optimisations que nous avons proposées améliorent nettement les temps d'exécution de tous les NPB. En particulier, il y a un gain relatif proche de 70% pour LU et MG. L'impact sur EP est beaucoup plus faible, il s'agit en effet d'une application qui communique peu.

Cependant, nous signalons que ces résultats ont été difficiles à obtenir car difficilement reproductibles. L'encombrement du réseau semble influencer les temps d'exécutions qui peuvent varier fortement. Il sera nécessaire de refaire ces tests sur des liens dédiés.

5.5 Développements réalisés

Nous résumons dans cette section les différents développements que nous avons réalisés pour mener à bien nos expérimentations. Nous ne revenons pas en revanche sur les modifications que nous avons faites dans MPICH-Madeleine, Madeleine ou le système d'exploitation.

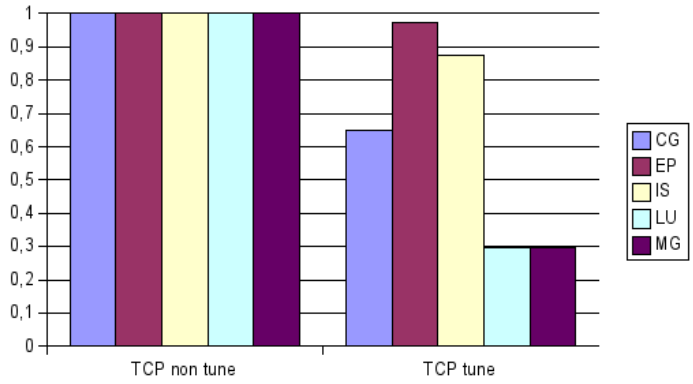


Figure 14: Comparaison relative des temps d'exécution des NPB sur TCP et TCP tuné

5.5.1 Réalisation d'une bibliothèque mesurant précisément les temps d'exécution

Afin de mesurer les performances de l'ensemble des expériences, nous avons été amené à développer une bibliothèque permettant d'appeler facilement, depuis les applications lancées sur la grille, des primitives permettant de mesurer facilement les temps d'exécutions. Cette bibliothèque utilise l'horloge système pour mesurer les temps écoulés.

Nous avons validé notre bibliothèque grâce à un appel à la primitive système *sleep()* pour vérifier que le timer calculait correctement le temps écoulé.

5.5.2 Réalisation d'un ping-pong MPI

Le ping-pong est basé sur l'envoi d'un message MPI grâce à la fonction *MPI_Send()*. Le processus MPI destinataire reçoit le message grâce à la fonction *MPI_Recv()* puis le renvoie à son expéditeur. Le temps est mesuré grâce à notre bibliothèque présentée à la section précédente : le timer est démarré lors de l'envoi du message sur le processus émetteur et arrêté à la reception du message de retour sur le même processus.

5.5.3 Réalisation d'un ping-pong TCP

J'ai également réalisé un ping-pong TCP pour connaître les performances de cette couche. Il s'agit toujours de l'envoi d'un message TCP vers un destinataire qui retourne le même message lorsqu'il l'a entiérement reçu. Les communications se font en utilisant les primitives d'émission/réception sur les sockets en mode connecté.

Nous avons réalisé notre propre ping-pong TCP afin de pouvoir plus facilement le modifier en particulier pour la prise en compte du *tuning* de TCP qui n'a pas été très facile à mettre en oeuvre. Cependant, nous avons validé les résultats obtenus avec notre ping-pong TCP en les comparant à ceux obtenus avec des outils existants comme *Netperf* ou *Iperf* qui permettent également de mesurer la latence et le débit au niveau TCP. *Netperf* permet d'effectuer un ping-pong entre deux machines en précisant la taille des tampons d'émission/réception des sockets. La (Figure 15) montre que notre ping-pong donne les mêmes mesures que *Netperf* ce qui est rassurant !

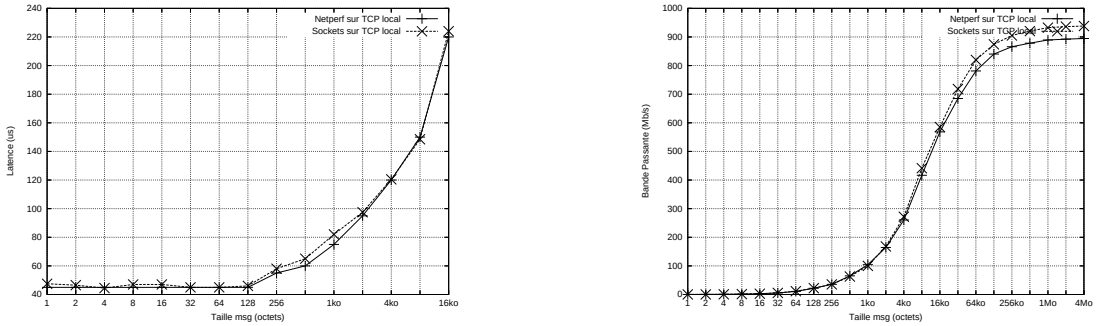


Figure 15: Comparaison des mesures obtenues avec Netperf et notre ping-pong TCP

6 Conclusion et perspectives

6.1 Conclusion

C'est au cours de ces dix dernières années que sont apparues les grilles de calculs, une grille étant l'agrégation de grappes de différents sites interconnectés par un réseau longue distance. L'utilisation de MPI ayant prouvé son efficacité dans les grappes et étant devenu le standard de fait utilisé par les applications parallèles, il est naturel de s'intéresser à l'étude et à l'optimisation des performances de MPI sur les grilles. Cependant, les différentes implémentations de ce standard disponibles pour les grappes sont peu ou pas capables de gérer efficacement les deux contraintes principales liées aux grilles de calculs que sont l'hétérogénéité et la longue distance.

Après une étude comparative des différentes implémentations existantes transposables aux grilles, nous avons choisi MPICH-Madeleine pour mener à bien nos expérimentations car c'est l'implémantation la plus efficace et la plus aboutie permettant de gérer efficacement l'hétérogénéité des réseaux de grille. De plus, il semblait assez aisé de modifier le module TCP de Madeleine pour réaliser des optimisations. MPICH-Madeleine est donc une bonne candidate pour les grilles de calcul, même si elle n'a pas été conçue dans cette optique et n'a pas été validée sur ce type de plateforme.

Nous avons donc mené une série d'expérimentations sur la plateforme nationale Grid'5000 pour étudier et optimiser le comportement de MPICH-Madeleine sur la grille. Dans un premier temps, nous avons constaté que les performances de MPICH-Madeleine sur la grille n'étaient pas bonnes, en tous cas très inférieures à celles constatées entre deux machines d'une même grappe. Dans un second temps, nous avons affiné nos expérimentations afin d'en déduire où se situaient les goulots d'étranglement dans le but de proposer des optimisations. Nous avons donc étudié les performances de chacune des couches de communications l'une après l'autre. Nous avons pour ce faire réalisé deux ping-pong (un au niveau MPICH-Madeleine et un au niveau TCP) et une bibliothèque permettant de mesurer précisément les temps d'exécution.

Cette étude nous a permis de constater que les performances du ping-pong TCP sur la longue distance étaient elles-aussi mauvaises. Nous avons donc proposé des optimisations à ce niveau qui se sont traduites en particulier par un paramètrage non seulement de la taille des tampons TCP dans le système d'exploitation mais aussi par la prise en compte de cette

modification dans la librairie Madeleine que nous avons du modifier. En effet, la taille optimale des tampons d'émission et réception au niveau TCP dépend du debit et de la latence du lien réseau ; elle doit donc être différente pour les liens intra et inter sites. Nous avons ainsi obtenu un gain de performances important : la bande passante maximale est ainsi passée de 95Mb/s à 320Mb/s. Cependant, les performances restaient très inférieures à celles obtenues sur une grappe. Nous avons donc proposé une optimisation dans MPICH-Madeleine consistant d'une part à vérifier que le seuil séparant les deux modes de communication était bien adapté à la grille, et d'autre part à augmenter la quantité de mémoire disponible pour l'émission et la réception des messages dans la couche basse de MPICH-Madeleine. Nous avons ainsi réussi à atteindre une bande passante maximale de 600Mb/s s'approchant des 900Mb/s obtenus sur une grappe. Enfin, nous avons tenté d'introduire une optimisation (nommée PsPacer) proposée par les concepteurs de GridMPI et différente de celle que nous avons proposée, consistant à modifier le comportement de TCP pour l'adapter aux applications MPI. Nous avons incorporé PsPacer à MPICH-Madeleine en modifiant le système d'exploitation. Les résultats ne sont pas probants avec l'application ping-pong mais nous pensons qu'ils pourraient l'être avec des véritables applications de grille. Il est donc nécessaire de faire des expérimentations complémentaires pour achever notre travail.

6.2 Perspectives

Certains points soulevés par les conclusions de nos expériences méritent des explications complémentaires voire des optimisations. Il est nécessaire de continuer à investiguer un réglage optimal des seuils séparant les différents modes de communications aussi bien au niveau MPICH-Madeleine que Madeleine. Il serait opportun de permettre un réglage dynamique de ces seuils pour permettre aux applications de s'adapter à la latence et bande passante disponibles. Enfin, nous allons refaire nos mesures sur un lien dédié (non partagé avec le trafic internet de RENATER, par exemple entre Lille et Sophia) afin d'être sûrs qu'aucun trafic parasite n'a faussé nos mesures. Nous devons également poursuivre les expérimentations réalisées avec de véritables applications de grille et en particulier celles effectuées avec les NAS Parallel Benchmarks.

Nous pensons que les améliorations que nous avons proposées sont efficaces mais qu'elles pourraient être encore bien plus omptimisées en repensant le cheminement des communications sur les passerelles à l'interface entre les réseaux rapides de grappes et le réseau longue distance de la grille. Par exemple, sur une grille telle que Grid'5000, l'utilisation de TCP n'est peut-être pas forcément nécessaire et s'avère sans aucun doute très coûteuse. En effet, d'une part les liens longue distance interconnectant les sites fournissent directement un accès Gigabit Ethernet sur chaque site comme s'il s'agissait d'un réseau local. D'autre part, la grille étant un réseau fermé et contrôlé, les mécanismes de TCP qui ont été conçus il y a une plusieurs dizaines d'années pour répondre aux caractéristiques d'Internet ne sont pas applicables aux caractéristiques d'une grille comme Grid'5000. Nous souhaiterions par exemple s'affranchir de l'utilisation de TCP pour traverser le réseau longue distance en proposant une réorganisation des communications sur les passerelles entre les réseaux Myrinet et Ethernet. La Figure 16 montre comment s'effectuent les communications dans l'état actuel et ce que nous proposons de mettre en oeuvre. Actuellement, le passage du réseau Myrinet au réseau Ethernet nécessite de remonter au niveau de la couche Madeleine (passage par les étapes 1 et 2 sur la Figure 16) en utilisant une recopie supplémentaire dans un tampon Madeleine intermédiaire. Il serait probablement possible de court-circuiter les étapes 1 et

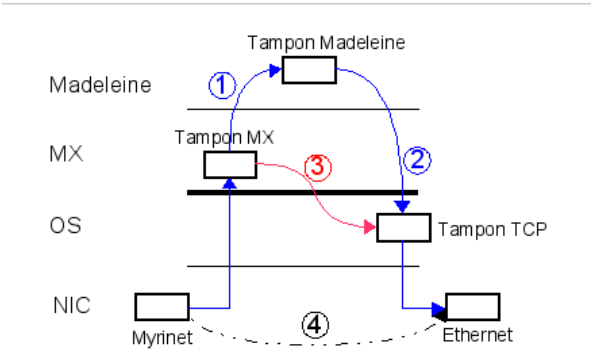


Figure 16: Passage de Myrinet à Ethernet sur la passerelle Madeleine

2 en passant par le chemin 3. Cela devrait permettre d'améliorer encore les performances des communicateurs longue distance grâce à la suppression d'une recopie et en évitant de remonter au niveau de la couche Madeleine sur la passerelle. Cette optimisation aurait lieu lors du passage de Myrinet à Ethernet sur le site source mais également lors du passage d'Ethernet à Myrinet sur le site destination.

Enfin, il serait peut-être même envisageable de réaliser le passage de Myrinet à Ethernet directement sans passer par TCP. Dans la proposition précédente, nous utilisons encore le passage par la couche TCP pour les communications inter-sites. Or, dans le contexte des grilles de calcul, les liens sont généralement dédiés, il y a donc peu de problèmes de congestion et de pertes de paquets. Il est donc envisageable de réaliser directement le passage de Myrinet à Ethernet sans remonter au niveau TCP, ni même au niveau IP, tel que cela est schématisé par le chemin 4 de la figure.

References

[1] Infiniband Trade Association. Infiniband architecture specifications, 2001. `http://www.infinibandta.org/home`.

[2] O. Aumage. Heterogeneous multi-cluster networking with the madeleine iii communication library. In *IPDPS*, 2002.

[3] NAS Parallel Benchmarks. `http://www.nas.nasa.gov/Software/NPB/#info`.

[4] Greg Burns, Raja Daoud, and James Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings of Supercomputing Symposium*, pages 379–386, 1994.

[5] Thomas Beisel Edgar Gabriel, Michael Resch and Rainer Keller. Distributed computing in a heterogenous computing environment. *Recent advances in Parallel Virtual Machine and Message Passing Interface*, pages 180–188, 1998.

[6] Olivier Aumage et Guillaume Mercier. MPICH/MADIII : a cluster of clusters enabled MPI implementation. In *Proceedings, CCGrid*, 2003.

[7] M. Dayde E. Jeannot Y. Jegou S. Lanteri N. Melab R. Namyst P. Primet O. Richard E. Caron J. Leduc G. Mornet F. Cappello, F. Desprez. Grid'5000: A large scale, reconfigurable, controlable and monitorable grid platform. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, Seattle, Washington, USA, November 2005. https://www.grid5000.fr/.

[8] Message Passing Interface Forum. MPI-2: Extensions to the message-passing interface. November 2003. http://www.mpi-forum.org/docs/mpi2-report.pdf.

[9] Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure*. MorganKaufmann, July 1998.

[10] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[11] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[12] Antonin Bukovsky Graham Fagg and Jack Dongarra. HARNESS and fault tolerant MPI. *Parallel Computing*, 27(11):1479–1496, 2001.

[13] GridMPI. http://www.gridmpi.org/gridmpi.jsp.

[14] W. Gropp and E. Lusk. The MPI communication library : its design and a portable implementation. In IEEE Computer Society Press, editor, *Proceedings of the Scalable Parallel Libraries Conference*, pages 160–165, October 1994.

[15] William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. High-performance, portable implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, 22(6):789–828, 1996.

[16] Myricom Inc. Myrinet Express (MX) : A High Performance, Low-Level, Message-Passing Interface for Myrinet. 2006. http://www.myri.com/scs/MX/doc/mx.pdf.

[17] Quadrics Inc. Release notes for MPI.1.24. August 2005.

[18] J. Wu J. Liu and D.K.Panda. High Performance RDMA-Based MPI Implementation over Infiniband. *The International Journal of Parallel Programming*, 32(3), June 2004.

[19] Argonne National Laboratory. MPICH2 : A portable implementation of MPI. 2005.

[20] Lawrence Berkeley National Laboratory. *TCP Tuning Guide*. 2004.

[21] Jeff Layton. Clusters interconnects : The whole shebang. *ClusterMonkey*, Avril 2006.

[22] Y. Kodama R. Takano Y. Ishikawa M. Matsuda, T. Kudoh. TCP adaptation for MPI on long-and-fat networks. In *Cluster*, 2005.

[23] T. Bemmerl M. Poeppe, S. Schuch. A message passing interface library for inhomogeneous coupled clusters. In *IPDPS, Workshop for Communication Architecture in Clusters*, April 2003.

[24] Guillaume Mercier. Communications à hautes performances portables en environnements hiérarchiques, hétérogènes et dynamiques. *Thèse de doctorat*, 2005.

[25] I. Foster N. Karonis, B. Toonen. MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing*, pages 551–563, 2003.

[26] A. Pant and H. Jafri. Communicating efficiently on cluster based grids with MPICH-VMI. In *Proceedings of Cluster Computing*, 2004.

[27] Site PVM. Parallel virtual machine. `http://www.csm.ornl.gov/pvm/`.

[28] Quadrics. `http://www.quadrics.com/`.

[29] JF.Méhaut R. Namyst. PM2 : Parallel multithreaded machine, a computing environment for distributed architectures. In *Parallel Computing (Par'Co 95)*, pages 279–285, 1995.

[30] Site RENATER. Renater4. `http://www.renater.fr/`.

[31] David J. Daniel Nehal N. Desai Ronald G. Minnich Craig E. Rasmussen L. Dean Risinger Richard L. Graham, Sung-Eun Choi and Mitchel W. Sukalski. A Network-Failure-Tolerant Message-Passing System For Terascale Clusters. In *ICS02*, New York, USA, June 2002.

[32] SCI. `http://www.lfbs.rwth-aachen.de/~joachim/SCI/`.

[33] TOP500. `http://www.top500.org/`.

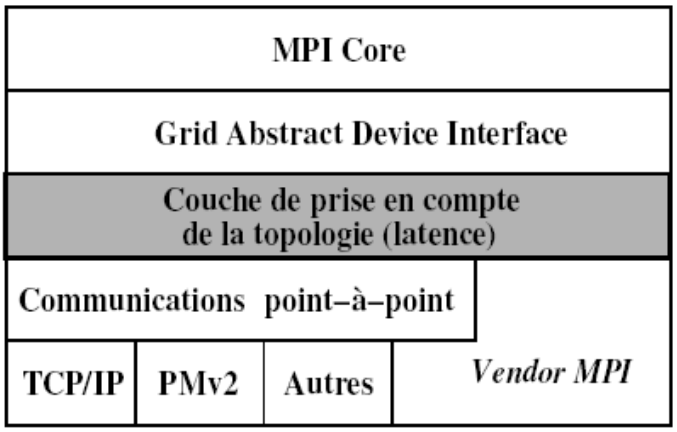


Figure 17: Architecture de GridMPI

A GridMPI

La Figure 17 nous montre l'architecture adoptée par les auteurs. Comme on peut le voir sur la figure, une couche de gestion de la latence, permet des optimisation pour le réseau TCP. C'est dans cette couche que sont effectuées les optimisations vues plus haut.

B MPICH-G2

Architecture générale Cette implémentation propose différents services qui sont, pour la plupart fournis par Globus, ensemble de logiciels pour aider au developpement d'applications dans les grilles de calcul. On peut notamment gérer :

- la sécurité (authentification, droits d'accès)
- l'infrastructure et sa découverte
- les calculs qui s'exécutent
- les données
- la collaboration
- la distribution et la construction de packages

Dans cette implémentation, cet ensemble d'outils permet essentiellement de gérer les problèmes d'authentification et de gestion des applications.

Pour son fonctionnement MPICH-G2 utilise :

- RSL : script de configuration du nombre de machines, des ressources nécessaires (nombre de processeurs) et des paramètres (variables d'environnements)
- DUROC (librairie de co-allocation) : permet de planifier et de démarrer l'application sur les différentes machines.

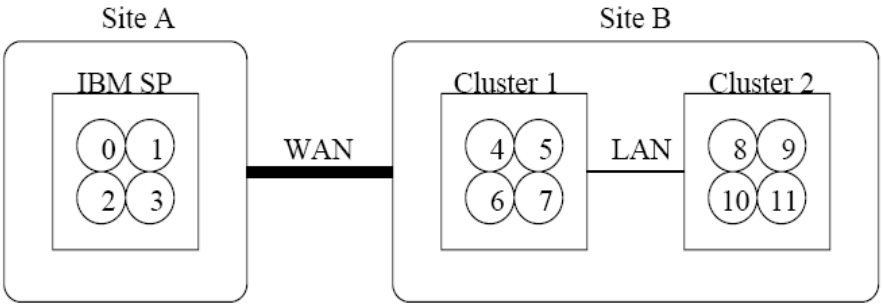


Figure 18: Répartition des processus de Mpich-G2

Rank		0	1	2	3	4	5	6	7	8	9	10	11
Depth		4	4	4	4	3	3	3	3	3	3	3	3
Colors	wide area	0	0	0	0	0	0	0	0	0	0	0	0
	local area	0	0	0	0	1	1	1	1	1	1	1	1
	system area	0	0	0	0	1	1	1	1	2	2	2	2
	*v*MPI	0	0	0	0								

Figure 19: Niveaux d'exécution selon les couleurs

- GRAM : permet la gestion et l'allocation des ressources de la grille. Elle permet d'autentifier l'utilisateur, d'autoriser les éxecutions, et ensuite interagir avec l'ordonnanceur local.

La gestion de la topologie La topologie du réseau est gérée grâce à un système de "couleurs". Pour chaque niveau de réseau, il est associé une couleur. Celle-ci est ensuite utilisée dans le mécanisme de découverte de la topologie et dans les opérations collectives. Elle permet notamment de déterminer lors de la programmation à quelle distance devront s'éxecuter les opérations.

La Figure 19 montre les couleurs associées à la répartition de processus de le Figure 18. Par exemple, lorsque le processus 1 veut communiquer avec le processus 3, leur couleur commune, 0, est située au niveau VendorMPI (Implémentation de MPI réalisée par les constructeurs pour leur réseau). Ils utiliseront donc ce réseau pour communiquer entre eux. Par contre, si les tâches 4 et 9 veulent communiquer ensemble, elles utiliseront le LAN car leur couleur commune est située à ce niveau.

C MetaMpich

La Figure 20 nous montre le schéma de fonctionnement de MetaMPICH.

Gestion de la distribution des processus Il est utile de connaitre le nombre de processus que chaque noeud peut supporter. Pour que chaque processus connaisse sa configuration

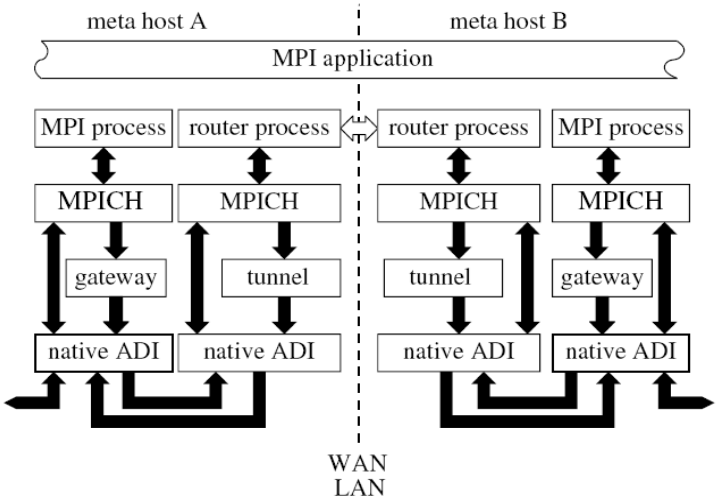


Figure 20: Architecture de MetaMPICH

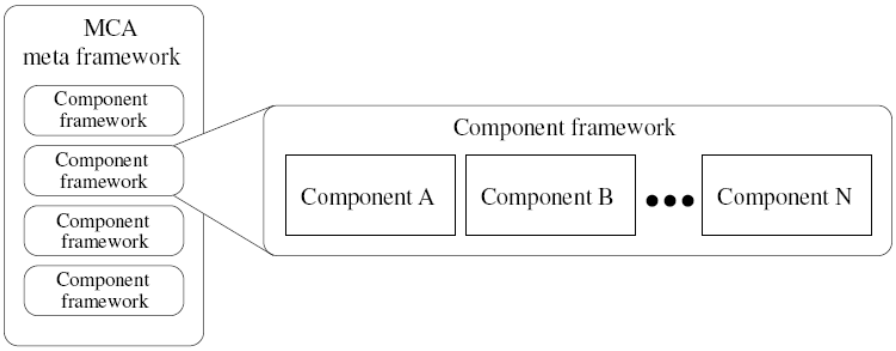


Figure 21: Architecture d'OpenMPI

(s'il est routeur ou processus normal), il est nécessaire d'analyser et de distribuer le fichier de configuration.

Les auteurs proposent des résultats qui ne montrent pas de comparaison de performance avec d'autres implémentations (ce qui est très dommage). Pour le moment, MetaMpich n'a pas été porté sur d'autres protocoles que TCP et ATM.

D OpenMPI

La Figure 21 montre le schéma de l'architecture en 3 composants imbriqués :

- MCA : contient l'architecture complète et fournit des services comme la configuration des composants au lancement.
- Composants du framework : fournissent un service particulier. Ils se chargent de découvrir, de charger, et d'utiliser les modules qui les concernent. Par exemple, on peut

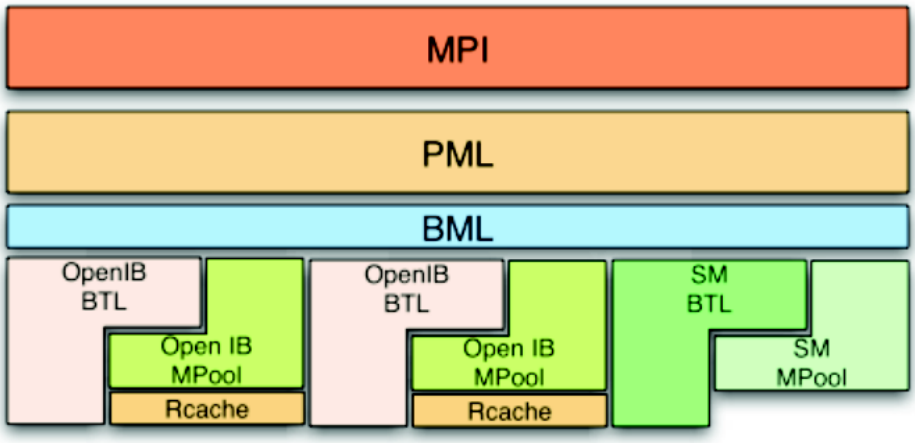


Figure 22: Les différents modules d'OpenMPI

citer :

- couche de transport : Ses modules sont dédiés à une technologie particulière (TCP/IP, Quadrics elan4 ...)
- couche de gestion du point à point : implémentation de la syntaxe MPI, gestion de la fragmentation et du réassemblage, de l'ordonnancement
- communications collectives : Support pour les opérations MPI

- Les modules : Les modules sont découvert pendant la phase de configuration, soit en précisant directement leur emplacement, soit en les placant dans le bon répertoire (découverte automatique).

Architecture des communications point à point

Comme le montre le schéma Figure 22 l'architecture des communications point à point est basée sur différents modules empilés :

- Le module PML (Point-to-point Messaging Layer) : implémente les modes de communications (avec mémoire tampon, synchrone, mode "ready")
- La couche BML (BTL Management Layer) : sert au moment de la découverte des modules. Il permet de signifier aux couches hautes la présence de ceux-ci. Ensuite, il est outre-passé.
- Le module BTL (Byte Transfert Layer) : fournit un ensemble de primitives pour la communication. Chaque BTL correspond à un module réseau.

E Mpich-VMI

Comme le montre la Figure 23, l'architecture de MPICH-VMI est assez similaire à MPICH-MAD. Les communications réseaux sont effectuées par la bibliothèque VMI (qui est le pendant de Madeleine). Celle-ci intègre différentes optimisations pour la disponibilité, l'utilisabilité et la gestion des grilles :

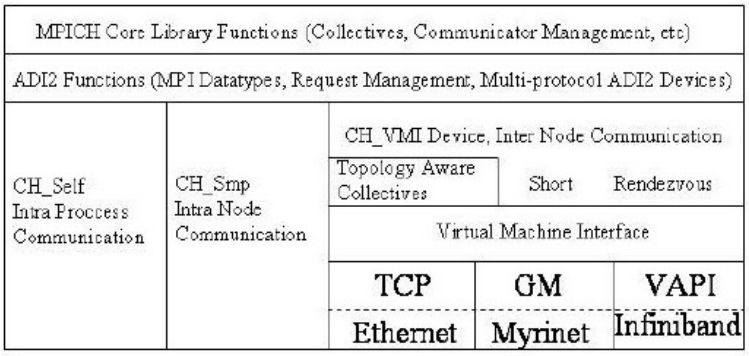


Figure 23: Architecture de MPICH-VMI

- partage des données sur les différents réseaux disponibles
- changement dynamique de réseau si l'un tombe, passage par une autre interface, s 'il y en a une de disponible (ce mécanisme n'est pas encore implémenté)
- chargement dynamique des modules réseau
- portabilité sur différentes architectures
- gestion des noeuds à distance

Elle supporte de TCP, GM et VAPI respectivement sur les réseaux Ethernet, Myrinet et Infiniband. Au lieu d'utiliser une bibliothèque de threads légers comme Marcel, les processus sont des démons.