Impact des architectures multiprocesseurs sur les communications dans les grappes de calcul : de l'exploration des effets NUMA au placement automatique

Stéphanie Moreaud

Mémoire de Master Recherche

présenté par

Stéphanie Moreaud

Impact des architectures multiprocesseurs sur les communications dans les grappes de calcul : de l'exploration des effets NUMA au placement automatique

Encadrement : Brice Goglin et Raymond Namyst

Février – Juin 2007

Laboratoire Bordelais de Recherche en Informatique (LaBRI)
Université Bordeaux 1

Table des matières

Introduction

Sismologie, physique des particules, aéronautique, aérospatiale, météorologie, finances, ... Nombreux sont les domaines qui nécessitent une puissance de calcul gigantesque pour traiter rapidement de grandes quantités de données, et dont les besoins ne cessent de grandir. Par exemple, l'utilisation du futur accélérateur de particules LHC, (*Large Hadron Collider*) du CERN, dont la mise en route est prévue en 2008, devrait produire 15 millions de gigaoctets de données à traiter chaque année. Les calculs effectués sur ces données vont requérir une puissance de calcul considérable, apportée par des grappes appartenant à plusieurs pays, et vont générer une quantité gigantesque d'entrées-sorties. De nombreux travaux ont été menés dans le domaine du calcul haute performance, tout d'abord à travers les super-calculateurs massivement parallèles basés sur des technologies spécialisées, puis plus récemment avec la mise en commun de nombreuses ressources plus standard dans les grappes ou les grilles de calcul.

La fréquence des processeurs a longtemps été un repère de l'avancée technologique. Mais depuis peu, les contraintes de dissipation thermique plafonnent la fréquence des processeurs traditionnels autour de 4 GHz. La parallélisation est ainsi devenue la nouvelle voie de développement, avec l'émergence des "multi-cœurs", qui rassemblent plusieurs processeurs sur une même puce, et sont présents dans les nouveaux ordinateurs personnels. On assiste ainsi à une généralisation de la parallélisation interne, et avec elle, à une hiérarchisation des machines. Par exemple, les architectures SMP (*Symetric MultiProcessing*) des machines multi-processeurs les plus répandues, dont les processeurs accèdent à une mémoire commune par un bus mémoire, peuvent maintenant être composées de processeurs multi-cœurs eux même hiérarchisés, par exemple par le partage de certains niveaux de cache.

La généralisation de la parallélisation interne conduit toutefois à la nécessité de distribuer la mémoire lorsque le nombre de processeurs devient élevé, pour éviter le goulet d'étranglement engendré par les multiples accès concurrents aux anciens un bus centralisés. Des machines de types NUMA (*Non Uniform Memory Access*) distribuent ainsi leur mémoire au prix d'iniquités des temps d'accès aux différentes zones mémoire. Il est désormais connu que le temps requis pour accéder aux données varie graduellement avec la distance physique à la mémoire et que ces contraintes doivent être prises en compte lors de l'ordonnancement des tâches et du placement de leurs données. Il est également fréquent que les contrôleurs d'entrées-sorties ne soient pas positionnés centralement et se trouvent donc physiquement plus près de certains processeurs et bancs mémoire que d'autres, imposant là aussi de possibles contraintes de placement sur les performances de l'accès aux périphériques.

Depuis une quinzaine d'années, les super-calculateurs, réservés à un marché élitiste, sont concurrencés, dans le domaine du calcul scientifique parallèle, par des grappes de calcul (*clusters*). Tandis que les super-calculateurs sont fondés sur une parallélisation interne, les grappes de calcul se basent quant à elles, sur une parallélisation externe : plutôt que de multiplier les unités de calcul à l'intérieur de la machine, elles font collaborer un ensemble d'ordinateurs qui communiquent via un réseau rapide dédié. Elles sont évolutives et extensibles, et présentent un bon rapport performance/prix, car sont construites à partir de matériel standard. Ces grappes exposent

toutefois de nouvelles contraintes au programmeur car, contrairement aux super-calculateurs, elles ne proposent pas de communication par mémoire partagée implicite. Elles apparaissent sous la forme d'une agrégation de ressources. Il est nécessaire de concevoir des communications explicites de manière efficace, afin de ne pas limiter les performances des calculs parallèles.

Avec la généralisation du parallélisme interne aux machines et le succès de la parallélisation externe via les grappes de calcul, on tend maintenant de plus en plus à associer ces deux modèles, en utilisant des nœuds plus complexes. On crée ainsi des *constellations*, des grappes dans lesquelles on trouve plus de parallélisation interne que de parallélisation externe. Ces *clusters* de machines parallèles sont plus chers que les grappes de machines mono-processeur, mais conservent un assez bon rapport performances/prix, car la diminution de la parallélisation externe réduit le nombre de communications coûteuses entre les nœuds du grappes. En effets, même si les réseaux hautes performances ne cessent d'évoluer, la latence générée par les communications entre deux nœuds d'une grappe reste bien supérieure au temps de communication entre deux processeurs partageant la mémoire d'une machine. Dans les grappes, il est ainsi important de réduire le nombre et le coût des communications inter-nœuds pour obtenir une efficacité optimale.

L'obtention de performances optimales dans les grappes de nœuds hiérarchiques qui se généralisent passe donc d'une part par la maîtrise des effets NUMA liés à l'architecture interne de la machine, et d'autre part par l'optimisation des communications externes entre les nœuds. On peut donc s'interroger sur l'influence de ces effets sur les performances des communications puisque la distance physique entre les processus, la mémoire et le réseau peut être un facteur limitant les performances. À notre connaissance, ce problème, pourtant connu, n'a encore jamais été étudié en détails et n'a fait l'objet d'aucune publication.

Ce mémoire présente donc une étude de l'impact de l'éloignement entre les processus (et des zones mémoire qu'ils utilisent) et les cartes réseau, sur les communications dans des grappes de machines NUMA. L'objectif de cette étude est dans un premier temps, de proposer une quantification de ces effets, que nous appellerons par extension NUIOA (*Non Uniform Input/Output Access*), et de déterminer les conditions dans lesquelles ils sont significatifs. Dans un deuxième temps, nous cherchons des solutions pour limiter d'éventuels effets pénalisants sur les performances et assurer une bonne efficacité des communications.

Nous présentons en partie 1 l'architecture des machines parallèles actuelles et les problèmes qu'elles soulèvent par l'éloignement des différentes ressources physiques. Nous étudions ensuite dans le chapitre 2 l'impact des différents placements sur les performances des communications entre des machines NUMA sur un réseau rapide. Nous décrivons enfin en partie 3 une méthode que nous avons conçue pour placer automatiquement les tâches dans la bibliothèque NewMadeleine, spécialisée dans les communications réseau pour grappes, et ainsi garantir la portabilité des performances. Nous discutons pour terminer des intérêts de connaître la topologie détaillée des machines et des méthodes envisageables pour obtenir des informations sur celle-ci afin de généraliser et optimiser notre méthode de placement.

Chapitre 1

Contexte

1.1 Des machines de plus en plus complexes

Avec la demande grandissante en puissance de calcul, le parallélisme se développe depuis plusieurs décennies, et la recherche dans ce domaine est florissante. Il reste toutefois réservé à des organismes disposant de moyens financiers importants.

La course à la performance s'est faite à moindre échelle, avec l'augmentation régulière de la fréquence des processeurs. La technologie se heurte désormais à une barrière thermique qui limite cette fréquence, et le parallélisme apparaît comme la voie de développement la plus prometteuse. La recherche dans ce domaine ne cesse de se développer et le parallélisme devient de plus en plus accessible.

Pour augmenter la puissance des machines, des architectures de plus en plus complexes sont créées, et avec elles de nouvelles contraintes. Cette section présente différentes architectures de machines parallèles très répandues.

1.1.1 Architectures SMP et Multi-cœurs

Les machines multiprocesseurs les plus courantes de nos jours sont des machines à accès mémoire "symétriques", les SMP (*Symmetric MultiProcessing*). Dans ces machines, plusieurs processeurs ont accès à la mémoire de façon uniforme et partagent le même bus mémoire, comme illustré sur la figure 1.1.

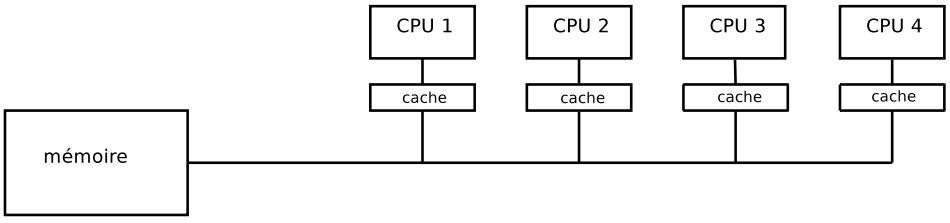


FIG. 1.1 – Machine SMP à quatre processeurs.

C'est sur ce modèle que sont construits les processeurs actuels. La loi de Moore promettait une amélioration considérable de la puissance des processeurs grâce à une meilleure finesse de gravure. Alors que celle-ci semblait compromise par les problèmes thermiques, la miniaturisation a permis le développement de processeurs "multi-cœurs".

Les multi-cœurs ont la particularité de regrouper plusieurs processeurs sur la même puce. Ils s'appuient ainsi sur le principe des SMP, (chaque cœur à un accès à la mémoire symétrique à celui des autres), mais avec un coût de production réduit. Des processeurs multi-cœurs de plus

en plus travaillés sont produits, ils possèdent de plus en plus de cœurs, mais aussi des formes de connection différentes, et plusieurs niveaux de cache.

En intégrant des processeurs multi-cœurs dans des machines SMP, on obtient des machines disposant potentiellement d'un premier niveau de hiérarchie, puisque les cœurs d'une même puce peuvent disposer d'un niveau de cache commun.

Les SMP présentent un modèle de programmation simple, puisque chaque processeur a un accès à la mémoire commune comparable à celui des autres, ce qui leur a valu un franc succès pendant des années. Elles présentent toutefois un inconvénient majeur : une contention pour l'accès à la mémoire peut se créer sur le bus. En effet, si de nombreux accès à la mémoire ont lieu, le bus mémoire peut devenir un goulot d'étranglement. Les SMP possèdent donc un nombre limité de processeurs, le plus souvent deux ou quatre, et généralement pas plus de seize.

Pour créer des machines parallèles de plus grande taille, il faut pallier le problème suscité par l'accès à la mémoire via un unique bus.

1.1.2 Architectures NUMA

Les machines parallèles NUMA (*Non Uniform Memory Access*) sont des machines multi-processeurs à mémoire partagée, dans lesquelles la mémoire est morcelée en plusieurs bancs. Chaque processeur accède à une zone mémoire "locale" rapidement, et à des zones mémoire "distantes" avec un temps d'accès plus long. Ces variations pour l'accès aux différents bancs mémoire conduisent à la notion de "facteur NUMA". Un facteur NUMA correspond au temps d'un accès à la mémoire locale, divisé par le temps d'accès à la mémoire distante. Ce facteur traduit la non uniformité d'accès à la mémoire, et contribue à l'évaluation de la machine en terme de performances. Plus celui-ci se rapproche de un, plus le coût supplémentaire, engendré par un accès distant par rapport à un accès local, est faible.

Pour des machines NUMA dont la topologie est complexe, ou hiérarchique à plusieurs niveaux, plusieurs facteurs NUMA caractérisent la machine. Ces architectures sont constituées de "nœuds" NUMA, qui sont des ensembles "banc mémoire, processeurs et caches", reliés entre eux par un réseau rapide ou un commutateur, comme illustré sur la figure 1.2. Chacun de ces nœuds a une topologie semblable à celle d'une machine SMP, et possède donc un nombre limité de processeurs.

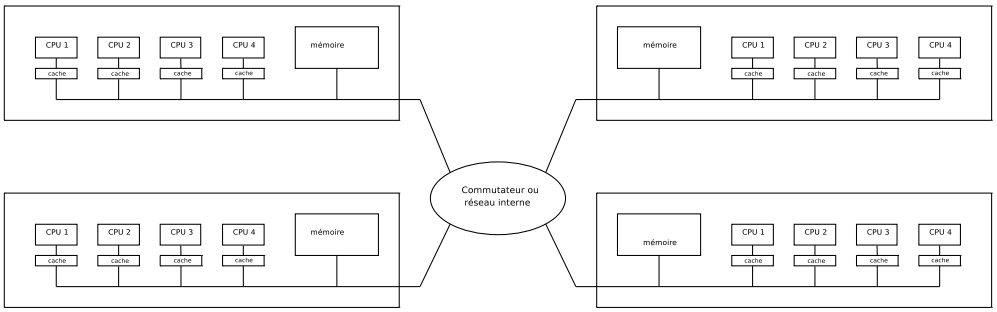


FIG. 1.2 – Machine NUMA composée de quatre nœuds comportant chacun quatre processeurs et un banc mémoire.

De telles machines marquent un niveau de parallélisme élevé. En effet, une machine NUMA peut correspondre à un ensemble de SMP, elles même constituées de multi-cœurs. Ces machines peuvent également être rassemblée en grappes par un réseau haute performance.

Notons que dans de telles grappes, chaque machine NUMA correspond à un nœud de la grappe, le nom "nœud" est alors utilisé en termes de hiérarchie, mais les modes de communication entre deux "nœuds NUMA" et deux "nœuds de grappe" diffèrent complètement, puisque le

premier se fait implicitement par mémoire partagée, tandis que le second est fait par communications explicites.

Les grappes de machines hiérarchiques, appelées *Cluster* ou *Constellations* selon les degrés de parallélisme interne et externe, constituent une part importante des installations utilisées pour les calculs d'ampleur, comme en témoignent les 78 % qu'ils représentent dans le classement Top500 des plus gros calculateurs de la planète [1]. La recherche et la concurrence entre les constructeurs ont abouti à 2 classes principales d'architecture NUMA.

Architectures hiérarchiques

Les architectures classiques basées autour d'un unique bus mémoire auquel sont connectés plusieurs processeurs peuvent être assemblées pour créer des machines hiérarchiques. Les différents bus mémoire sont alors reliés par un réseau interne d'interconnexion. Cependant, la traversée de ce réseau étant plus lente qu'un accès à la mémoire locale, des effets NUMA apparaissent.

Cette architecture est utilisée depuis assez longtemps et subsiste de nos jours, en particulier dans les serveurs basés sur les processeurs Itanium d'Intel. Les machines Bull Novascale assemblent par exemple plusieurs QBB (*Quad Building Block*) composés chacun de 2 ou 4 processeurs et d'un banc mémoire, (illustré par la figure 1.2). La société SGI spécialisée dans ces architectures propose quant à elle des serveurs Altix à plusieurs niveaux de hiérarchie regroupant jusqu'à plusieurs centaines d'Itanium. Le facteur NUMA varie dans ces machines de 1 à 3.

Architectures distribuées

On observe depuis quelques années l'apparition d'architectures non-régulières, contrairement aux systèmes SMP classiques ou aux machines NUMA hiérarchiques. L'assemblage de plusieurs bus mémoire par un réseau d'interconnexion dédié étant onéreux, AMD a développé le système HyperTransport [2], qui est souvent appelé *bus* mais est en fait un réseau d'interconnexion. Au lieu d'être connecté à un bus mémoire centralisée, les processeurs AMD Opteron sont connectés à plusieurs liens HyperTransport (de 1 à 4 suivant les modèles). La mémoire est distribuée et chaque banc est relié à un de ces liens, ce qui le place près d'un processeur et loin des autres. Le facteur NUMA observé varie entre 1 et 2 [3].

Cette architecture est très répandue dans le monde du calcul scientifique, comme en témoigne les 22 % de systèmes l'utilisant dans le classement Top500 [1]. Les périphériques d'entrées-sorties sont eux aussi attachés à un lien HyperTransport, ce qui provoque l'apparition d'effets NUIOA (*Non Uniform Input/Output Access*) pour les entrées-sorties, en particulier pour les communications sur réseau rapide dans les grappes de calcul.

La figure 1.3 illustre ce type d'architecture qui compose l'ensemble de notre plateforme de tests, détaillée dans la section 2.2.

1.2 Importance de la proximité des ressources

De nombreuses études sur les effets du placement des tâches (*threads*) et de la mémoire sur les machines NUMA ont été menées, notamment dans le contexte de l'ordonnancement.

Avec la répartition de la mémoire en plusieurs bancs, le placement et l'ordonnancement des tâches devient crucial. En effet, un thread doit être placé près des données qu'il manipule, pour éviter des accès mémoire distants, pénalisants pour les performances, et dont le facteur NUMA augmente avec la taille de la machine [4]. Les affinités des processus et des threads pour le

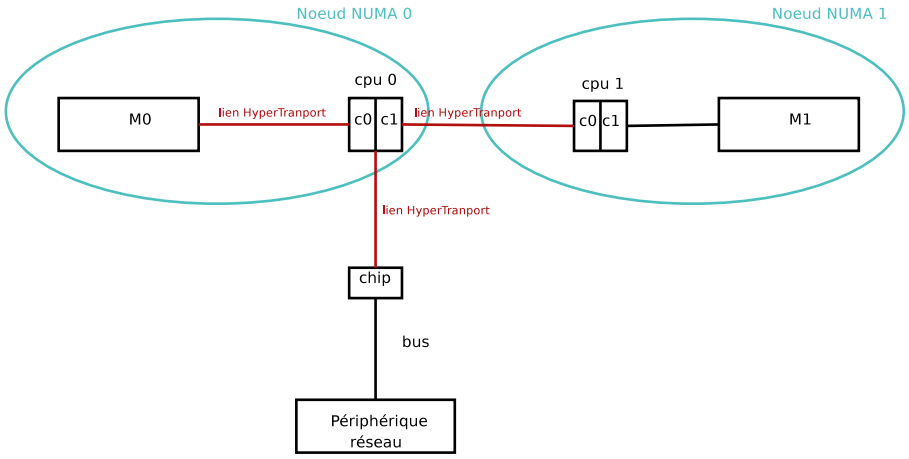


FIG. 1.3 – Machine NUMA composées de deux processeurs OPTERON bi-cœurs, avec un périphérique d'entrées-sorties placé sur le nœud 0.

partage des caches et l'accès aux bancs mémoire ont également un impact important sur les performances générales. Des travaux exposent ainsi l'importance de la localité des données [5], ou encore l'intérêt de faire migrer des pages pour conserver les affinités mémoire [6].

Les effets NUMA sont également connus pour avoir une influence sur les communications. Sur les grosses machines, il a été prouvé que les effets NUMA ont un impact majeur sur les communications MPI à l'intérieur de la machine [7]. Ils ont également une influence sur les réseaux haute performance dans les grappes.

Certains systèmes d'exploitation comme LINUX fournissent des outils de placement NUMA tels que `numactl` ou `libnuma` [8], qui permettent de forcer le placement des tâches et de la mémoire sur certains nœuds. Ces outils de placement sont utilisés dans toutes les mesures de performances des communications réseau impliquant des machines NUMA, pour garantir des performances optimales et reproductibles, ce qui prouve l'existence d'un effet du placement.

Des travaux ont mis en évidence que l'alignement des zones mémoire mises en jeu lors des communications, peut avoir un impact conséquent (jusqu'à 25 %) sur l'efficacité de certains réseaux rapides [9]. Cela montre la sensibilité des performances des communications sur réseaux rapides au placement physique des données en mémoire. Les contrôleurs d'entrées-sorties, et donc les périphériques qui y sont reliés, ne sont pas forcément connectés à la machine de manière symétrique. Ils se trouvent ainsi plus près de certains nœuds que d'autres, ce qui peut entraîner des temps d'accès variables.

Les effets du placement des threads sur des machines NUMA, sur les performances générales, ainsi que les communications sur réseaux rapides, font l'objet de nombreuses recherches. Pourtant, à notre connaissance, encore aucune étude traitant de leur interaction n'a été publiée, malgré le développement des grappes de calcul les faisant directement interagir.

On peut ainsi se demander quelle est la part des effets NUMA, sur les accès aux interfaces d'entrées-sorties et, si elle est conséquente, s'il est possible de mettre en œuvre des moyens pour réduire ces effets et améliorer les performances des communications.

La figure 1.4 présente les résultats d'une expérience de communication entre deux machines NUMA. Les détails de ce test seront explicités dans la section 2.4.2. Nous nous attacherons simplement ici, à regarder l'effet d'un mauvais placement NUMA théorique du thread de communication (c'est-à-dire sur le nœud le plus éloigné de l'interface réseau) par rapport à un bon placement (au plus près de l'interface réseau) sur le débit observé.

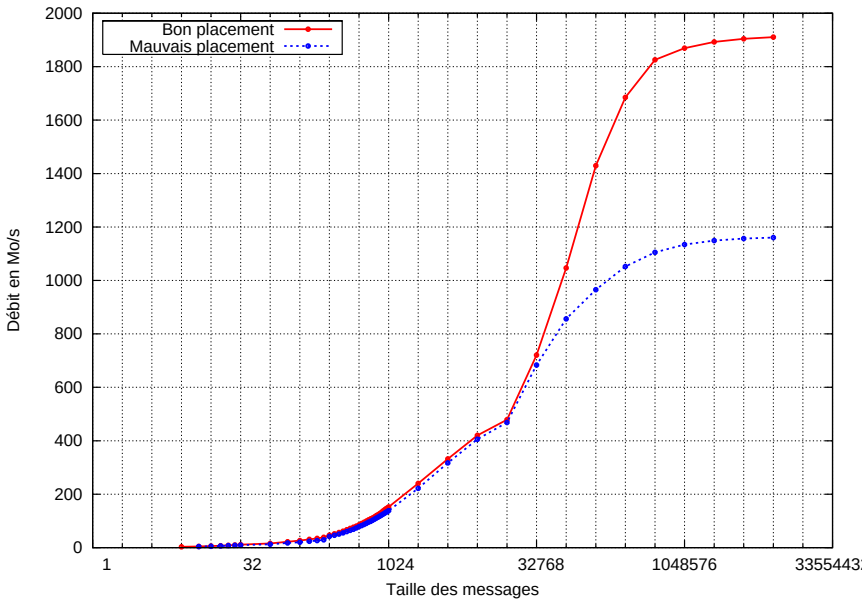


FIG. 1.4 – Débit observé lors d'un ping-pong multirails sur les réseaux MYRI-10G et QSNET.

Alors que pour un bon placement, le débit peu atteindre jusqu'à 1900 Mo/s, il est borné à moins de 1200 Mo/s dans le cas d'un mauvais placement. Dans cet exemple, un mauvais placement fait perdre près de 40% de débit, soit une baisse considérable. Ce test a été réalisé en forçant le placement des tâches et données à la main. Lorsque le placement est laissé libre et est déterminé par l'ordonnanceur, les résultats varient entre ceux des bons et mauvais placements de façon imprévisible et non reproductible. Un tel résultat prouve que le placement des tâches de communication sur des machines NUMA ne peut être négligé.

1.3 Discussion

L'expérience présentée précédemment met en avant l'importance que peut prendre le placement des tâches de communication et des tampons mémoire associés sur les performances des communications dans les grappes de calcul. Par ailleurs, il est courant de nos jours de placer explicitement les processus à la main sur le bon nœud NUMA lors de mesure de performance réseau afin de maximiser ces performances et leur reproductibilité.

Devant ce constat qui tend à prouver que les effets NUIOA existent et peuvent être importants, on est en droit de se demander quel impact précis le placement peut avoir sur les communications dans un cadre général. En particulier, l'impact se fait uniquement sur le débit ou bien également sur la latence des communications ? De plus, pouvoir prédire la dégradation de performances et en comprendre les causes exactes doit permettre de mieux les éviter.

Par ailleurs, la possibilité d'une dégradation hiérarchique des performances est également concevable dans les machines à architecture complexe. En effet, il n'est pas clair si la distance entre les différents nœuds NUMA et les contrôleurs d'entrées-sorties, voire même la topologie globale de la machine, peut avoir elle aussi une influence, par exemple à travers des contention sur le bus mémoire lorsque la machine est chargée. Une telle étude doit permettre d'une part de définir des méthodes minimisations des dégradations de performance et d'autre part des stratégies de placement automatique adapté aussi bien à de simples mesures de performance réseau qu'à des applications complexes qui mélangent calcul intensif sur l'ensemble des cœurs de la machine et communications externes.

Chapitre 2

Impacts du placement NUIOA sur les performances des communications

Ce chapitre a pour but de mettre en évidence les effets NUIOA qui peuvent intervenir dans des communications entre les nœuds d'une grappe de machines NUMA. Pour bien comprendre les facteurs qui peuvent interagir, nous commencerons par résumer les mécanismes qui interviennent dans les transmissions des messages. Nous étudierons alors les résultats de tests de transferts locaux, s'appuyant directement sur ces mécanismes, et leurs répercussions au niveau applicatif, après avoir présenté succinctement notre plateforme de test. Nous discuterons enfin de quelques phénomènes singuliers que nous avons pu observer.

2.1 Mécanismes mis en jeu dans les communications

2.1.1 Réseaux rapides

Dans les grappes de calcul, pour garantir des communications rapides entre les différents nœuds de la grappe (indispensables pour ne pas ralentir le temps de calcul des applications), les nœuds sont reliés via une technologie de réseau rapide. Ces réseaux permettent d'obtenir des temps de transfert de quelques micro-secondes pour de petits messages (1 μs théorique pour QsNet) et des débits pouvant atteindre plus d'un gigaoctet par seconde (1,2 Go/s pour Myri-10G).

À ce niveau de performance, le moindre surcoût a son importance car il peut très vite devenir un facteur limitant. Les bibliothèques de communications utilisées doivent ainsi réduire au maximum tout surcoût lors du transfert de données. Pour cela, elles utilisent un compromis entre plusieurs méthodes, pour optimiser les performances, utilisant les modes de transfert les plus efficaces selon la taille des messages. Ces méthodes sont présentées dans la section 2.1.2.

2.1.2 Modes de transfert

Les communications réseau se font par transferts de données de la mémoire de la machine vers l'interface (carte) réseau, puis par transmission de ces données sur le réseau jusqu'à la carte de la machine distante, et enfin par transfert depuis ce périphérique jusqu'à la mémoire locale.

Les transferts locaux entre une carte réseau et la mémoire (et inversement) peuvent s'effectuer de deux manières différentes : en mode PIO (*Programmed Input/Output*) ou en mode DMA (*Direct Memory Access*). Le mode PIO est un transfert généralement découpé en blocs de quelques octets, initié par le processeur. Dans ce mode, le processeur est donc occupé pendant le temps du transfert. Le mode DMA nécessite une initialisation du contrôleur DMA de la

carte réseau par le processeur, opération coûteuse en temps processeur. Le reste du transfert est ensuite traité en arrière plan, sans intervention du processeur.

Par souci de consommation de cycles processeurs, il est donc préférable, à partir d'une certaine taille de transfert de données, d'utiliser le DMA. Par contre, par souci de latence, le mode PIO est plus performant pour les petits messages, car le temps d'initialisation est très court. Ces deux modes de transferts sont illustrés par la figure 2.1.

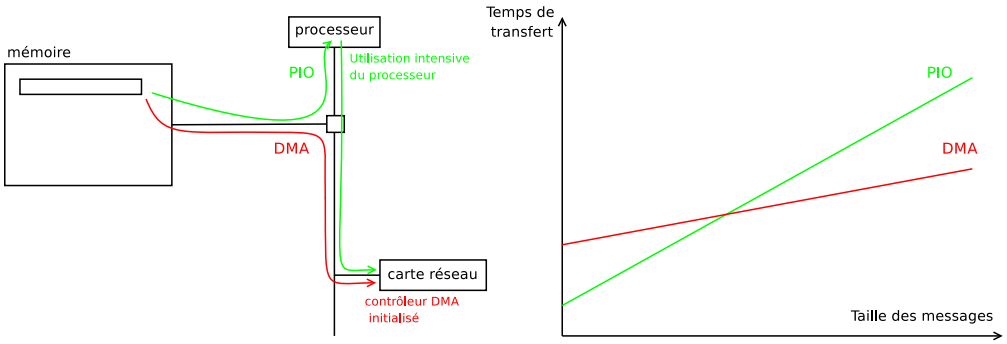


FIG. 2.1 – Transferts de données de la mémoire à la carte réseau en mode PIO et DMA et performances globales comparées.

Le mode DMA a cependant un inconvénient au niveau mise en œuvre car il impose de verrouiller les cadres de mémoire virtuelle en mémoire physique afin que le moteur DMA puisse y accéder sans craindre que le système d'exploitation ne déplace le cadre ou ne l'évince sur le disque. Ce coût de verrouillage étant important, différentes techniques ont été développées pour l'éviter. Certains réseaux utilisent donc une zone statique pré-verrouillée dans laquelle les données sont copiées pour le DMA. Lorsqu'une copie mémoire coûte moins cher que le verrouillage des cadres, cette stratégie est plus efficace. Elle est donc parfois utilisée comme mode transfert (copie+DMA) intermédiaire entre PIO et DMA.

Dans les machines NUMA, les interfaces réseau ne sont pas toujours "centrées" sur la machine. Il arrive ainsi qu'un ou plusieurs nœuds soient plus proches d'une interface réseau que les autres, comme l'illustre la figure 1.3 de la page 10, avec le nœud 0 plus proche du contrôleur d'entrées-sorties que le nœud 1. Les transferts locaux effectués en mode PIO, DMA ou copie+DMA, sont alors soumis à un paramètre de distance qui pourrait influencer le temps de transfert et ainsi les performances de communication générales.

Les tests réalisés dans ce chapitre ont pour but d'évaluer l'impact du placement des threads et buffers de communication sur les performances de communication générales, mais également sur les transferts locaux, puisque les performances des communications dépendent entre autres de celles de ces transferts.

2.2 Plateforme de tests

La plateforme sur laquelle nous avons réalisé les tests décrits dans les sections 2.3 et 2.4, est constituée de plusieurs machines NUMA, basées sur des processeurs bi-cœurs AMD OPTERON 1,8 GHz, chacun consistant un des 2 ou 8 nœuds NUMA de la machine. Ils sont reliés entre eux ainsi qu'aux contrôleurs d'entrées-sorties par des liens spécifiques à ce matériel, les liens HYPERTRANSPORT.

Les machines *Dalton* sont composées de deux nœuds NUMA, et disposent chacune de deux interfaces réseau, placées sur le nœud 0, MYRICOM MYRI-10G et QUADRICS QSNET II (interfaces ELAN4). On obtient sur le réseau MYRI-10G une latence de près de 2,5 μs, et un débit de

l'ordre de 1200 Mo/s, et sur le réseau QsNet environ 1,5 μs de latence et 900 Mo/s de débit.

Les machines *Infini* disposent elles aussi de deux nœuds NUMA et sont reliées entre elles et à la machine *Hagrid* par un réseau InfiniBand 8x (interfaces Mellanox InfiniHost III), dont l'interface est placée sur le nœud 1. Ce réseau atteint environ 4 μs de latence et 1400 Mo/s de débit.

La machine *Hagrid* dispose de huit nœuds NUMA et l'interface de son réseau InfiniBand est placée proche du nœud 0.

Le schéma 2.2 illustre ces architectures.

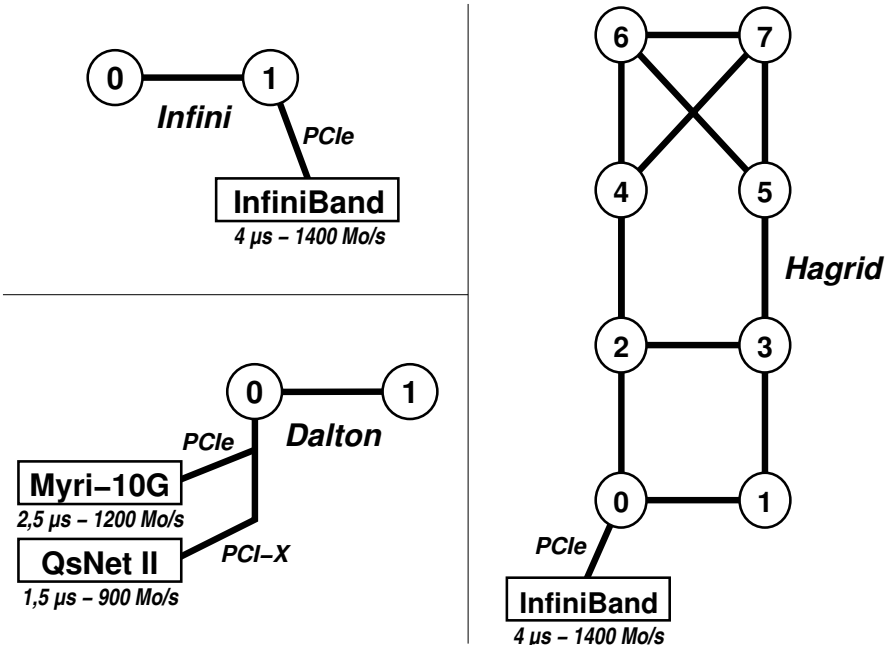


Fig. 2.2 – Architectures NUMA des machines de la plateforme de test.

2.3 Impact du placement sur la latence des communications

2.3.1 Latence des transferts locaux

Pour jauger l'influence du placement des threads et de la mémoire sur la latence, lors du transfert de données entre la machine et la carte réseau, nous avons mesuré des transferts locaux (entre la mémoire et la carte réseau).

Les tests effectués mesurent le délai entre l'envoi d'une requête vide à l'interface réseau et le moment où l'événement correspondant est notifié à l'application. Cette requête est traitée par la carte mais ne génère aucune communication réelle. Ces tests sont répétés en faisant varier le placement des tâches de communication sur les différents nœuds NUMA. Ils nous permettent ainsi d'observer l'impact du placement sur la latence des communications pour de petites requêtes.

Impact sur la latence pour des machines à 2 nœuds NUMA

Le tableau 2.1 présente les résultats obtenus pour les tests de PIO et DMA, effectués entre des architectures NUMA de 2 nœuds, pour les interfaces des trois réseaux rapides de notre plateforme de test : Myri-10G, QsNet et InfiniBand. On notera par la suite "bon placement"

et “mauvais placement” pour, respectivement, “placement sur le nœud le plus près de l'interface réseau” et “placement sur un nœud éloigné”.

	Bon placement	Mauvais placement	Surcoût
Myri-10G	1739	1794	55
QsNet	1610	1670	60
InfiniBand	464	559	95

Tab. 2.1 – Impact du placement NUMA sur la latence (en nanosecondes) pour de petites requêtes aux périphériques des différents réseaux rapides. Tests effectués sur des machines à 2 nœuds NUMA.

Les résultats pour le réseau Myri-10G ont été obtenus à partir du test `mx_piobench` fourni par le constructeur de ce réseau (Myricom). Ce test envoie des requêtes de 64 octets en PIO puis attend une réponse que lui transmet la carte en mode DMA. Le test en mode DMA `qsnet2_dmatest`, mis à disposition par le constructeur Quadrics, donne des informations de latence lorsqu'il est utilisé avec des tailles de messages très petites. Cela nous permet d'évaluer l'impact du placement sur la latence. Enfin, le réseau InfiniBand ne disposant pas de test constructeur, nous avons évalué les variations de latence à partir de notre propre test de PIO, qui effectue une écriture puis une lecture dans la mémoire du périphérique réseau.

Ces trois tests sont chacun très spécifiques au réseau auquel ils sont associés, ils ne sont donc en aucun cas comparables entre eux. Cela n'est pas gênant dans notre expérience, car nous ne souhaitons pas faire de comparaison entre ces réseaux. Il nous suffit de comprendre que ces tests donnent une façon similaire d'évaluer l'effet du placement NUMA sur la latence de chaque réseau.

Ces résultats nous montrent un surcoût entre 55 et 95 nanosecondes pour un aller retour entre le processeur et l'interface réseau, soit en moyenne près de 40 ns pour un accès. Ce surcoût correspond au temps nécessaire pour traverser le lien HyperTransport qui relie les 2 nœuds NUMA (voir figure 2.2) de la machine.

Après avoir observé le surcoût engendré sur la latence, par un mauvais placement sur une machine NUMA composée de deux nœuds, nous allons observer cet effet sur une architecture plus complexe.

Impact sur la latence pour une machine à 8 nœuds NUMA

Pour tenter de généraliser les résultats précédents, nous avons effectué le test de PIO depuis les processeurs vers la carte InfiniBand sur la machine Hagrid, pourvue de huit nœuds NUMA. Logiquement, plus il y a de liens à traverser plus la latence devrait être importante. Les résultats obtenus sont présentés sur la figure 2.3. Comme présumé, la latence de chaque aller retour augmente bien pour chaque lien traversé, et les résultats sont représentatifs de la hiérarchie de la machine. On peut évaluer là aussi le temps de traversée d'un lien HyperTransport à près de 40 ns.

2.3.2 Latence des communications au niveau applicatif

Pour évaluer l'impact du placement sur la latence de communications réelles, nous avons effectué des tests de ping-pong sur les différents réseaux de notre plateforme.

Nous observons une influence négligeable du placement sur le réseau Myri-10G. Le surcoût relevé lors d'un mauvais placement est de environ 25 ns, pour une latence de base d'environ 2,5 μs. Cela correspond à une variation de 2 % sur des machines à 2 nœuds NUMA. La latence

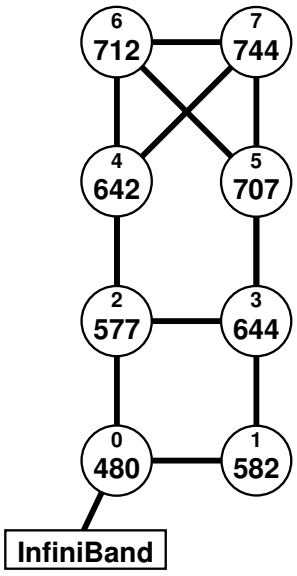


FIG. 2.3 – Latence en nanosecondes d'un aller retour entre le processeur et l'interface réseau pour de petites requêtes. Test effectué sur la machine NUMA *Hagrid*.

du réseau INFINIBAND étant plus élevée que celle de MYRI-10G, l'effet NUIOA en devient insignifiant.

Le réseau QSNET montre des conséquences plus notables d'un mauvais placement, notamment parce que sa latence est bien plus faible, entre 1 et 2 μs selon le mode de communication. Les tests les plus optimisés qui atteignent une latence de 1 μs semblent de plus souffrir tout particulièrement du mauvais placement puisque 100 ns supplémentaires sont observées de chaque coté sur des machines à 2 nœ uds NUMA. Cela se traduit par une variation de 20 % de la latence observée.

Les effets NUIOA sur la latence ont donc un impact qui peut s'avérer important dans le cas d'applications aux besoins de latence très sensibles s'exécutant sur certains réseaux et utilisant certains modes précis de communications. Mais en général, l'impact reste négligeable.

2.3.3 Bilan

Pour conserver une latence minimum, il semblerait judicieux de réduire au maximum le nombre de liens à traverser, puisque les tests prouvent que chaque traversée d'un lien HYPERTRANSPORT augmente la latence de quelques dizaines de nanosecondes, environ 40 ns de latence supplémentaires (soit 80 ns sur une communication complète entre 2 machines).

Il faut toutefois relativiser ces résultats. En effet, pour des communications dont la latence est très faible (de l'ordre de 1 μs), une telle perte de performance correspond à un fort pourcentage de la latence totale, et peut donc être à prendre en compte. Toutefois, cette perte est du même ordre de grandeur que l'accès mémoire à une valeur qui n'est pas dans le cache, et pour des communications de plusieurs μs, cela constitue une perte acceptable (moins de 2 %).

2.4 Impact du placement sur le débit des communications

Nous avons mis en évidence et quantifié dans la section précédente les effets des variations du placement NUMA sur la latence des communications. Nous allons maintenant observer les effets obtenus sur le débit.

2.4.1 Un impact en fonction du débit maximum

Pour évaluer les effets du placement sur le débit, nous avons dans un premier temps observé les performances de débit des transferts locaux. Nous nous sommes servis des tests `mx_dmabench` et `qsnet2_dmatest`, fournis par les constructeurs des interfaces réseau Myri-10G et QsNet. Ces tests consistent à faire des émissions sur la carte réseau, ou des réceptions depuis celle-ci, en utilisant le mode de transfert DMA. Les requêtes ne sont pas réellement transmises sur le réseau. Nous avons également exécuté un test similaire, `dma_bench`, sur le réseau SCI. Ce réseau, disponible sur les machines *Infini* présente des performances moindres que les réseaux jusqu'à présent cités dans ce mémoire (environ 250 Mo/s de débit et 2 μs de latence), mais il nous a servi à généraliser des hypothèses qui auraient pu n'être valables que dans un cas particulier.

Nous avons également observé les effets pour des requêtes sur l'interface InfiniBand, au moyen de notre propre test de DMA. Cependant, l'absence de programme dédié nous a empêché de mesurer ces effets sans réaliser des communications réseau réelles.

Résultats

Les résultats divers sont obtenus et résumés par la table 2.2.

		Bon placement	Mauvais placement	Impact
Myri-10G	Lecture	1308	1295	- 1 %
	Écriture	1518	1162	- 24 %
InfiniBand	Lecture	*1075*	*1075*	0 %
	Écriture	1392	1071	- 23 %
QsNet	Lecture	879	879	0 %
	Écriture	925	925	0 %
SCI	Écriture	251	256	+ 2 %

Tab. 2.2 – Impact du placement NUMA sur la bande passante (en Mo/s), sur des machines NUMA à 2 nœuds pour des transferts locaux en mode DMA de la mémoire vers la carte (écriture), et de la carte vers la mémoire (lecture).

Les variations de débit en fonction du placement sont conséquentes pour le test de DMA sur Myri-10G mais asymétriques. Alors que le débit maximum approche les 1500 Mo/s en émission, et que les performances chutent de près de 25 % dans le cas d'un mauvais placement, le débit stagne autour de 1300 Mo/s en réception et semble peu affecté par le placement. Avec un débit légèrement moins élevé, le test sur InfiniBand montre des résultats semblables. Par contre, le test sur QsNet présente un débit autour de 900 Mo/s, sur lequel les effets du placement sont négligeables. Sur SCI, les effets sont inexistants.

A première vue, plus le débit est grand, plus la perte semble importante dans le cas d'un mauvais placement. De plus, la lecture (réception) ne semble pas ou peu souffrir du placement alors que l'écriture (émission) est affectée.

Interprétations et hypothèses

Devant ces premières observations, nous pouvons faire quelques hypothèses. Tout d'abord, l'impact semble minime pour de faibles débits, il est possible que les effets ne soient visibles que lors de la saturation des bus mémoire. Le débit pour QsNet et SCI serait alors insuffisant pour atteindre cette saturation. Il existerait donc un "seuil" de débit à partir duquel on observerait un effet, et qui correspondrait au débit maximum avant saturation du bus mémoire. Une autre

hypothèse serait fondée sur la différence de bus. En effet, Myri-10G et InfiniBand sont reliés au contrôleur d'entrées-sorties par un bus de type PCI-Express, tandis que SCI et QsNet sont reliés par un bus PCI. Toutefois, la différence majeure entre ces bus est leur débit, de très loin supérieur pour le bus PCI-Express, il est donc tout à fait probable que l'effet soit dû à la différence de débit et non aux caractéristiques intrinsèques des bus d'entrées-sorties. Les transferts en direction de la carte réseau semblent souffrir davantage de la perte de débit, que ceux effectués dans le sens inverse.

Les résultats des tests de transferts locaux en mode DMA nous ont conduits à plusieurs constatations et hypothèses. Celles-ci ne sont cependant pas suffisantes pour être exploitables à elles seules. Il est donc nécessaire d'effectuer des tests complémentaires.

2.4.2 Test de ping-pong et ping-pong multirails

Dans cette partie, nous allons observer l'impact NUIOA sur le débit des communications grâce à des tests au niveau applicatif. Ces tests sont spécifiques à la bibliothèque de communication NewMadeleine [10]. Ils sont basés sur un ping pong, entre deux *Dalton* sur les réseaux Myri-10G et QsNet, ou entre deux *Infini* sur InfiniBand. Le test de ping-pong est effectué sur un réseau unique (QsNet, Myri-10G ou InfiniBand), tandis que le test de ping-pong multirails utilise simultanément deux réseaux (QsNet et Myri-10G sur les *Dalton*), ce qui permet une augmentation théorique du débit maximum en comparaison avec les tests "mono-rail".

Résultats

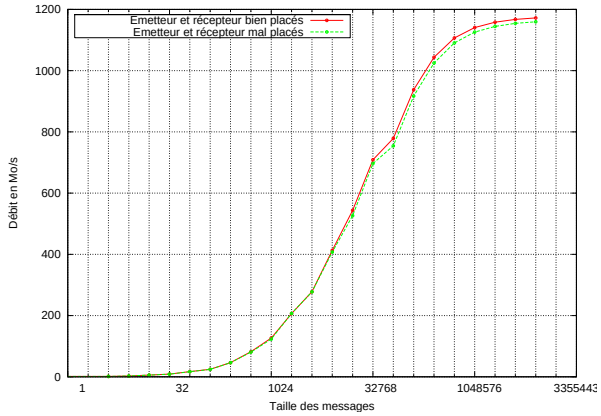
Les courbes de résultats pour le test de ping-pong sont présentées figure 2.4. Bien que les mesures précédentes sur Myri-10G présentaient d'importantes variations de débit en fonction du placement, les tests au niveau applicatif ne présentent pour ce réseau que des variations minimes (figure 2.4(a)). Il est à noter que le test précédemment étudié offrait un débit maximum de 1500 Mo/s, contre seulement 1200 Mo/s dans le test au niveau applicatif (le lien réseau est limité à 10 Gbit/s soit 1250 Mo/s). Les résultats pour le réseau QsNet s'avèrent similaires à ceux de Myri-10G pour un débit moins élevé. Par contre, on observe une variation du débit en fonction du placement pour le test sur InfiniBand (figure 2.4(b)), qui présente un débit plus élevé. Ces résultats tendent à valoriser l'hypothèse d'un seuil à partir duquel on observerait l'impact d'un mauvais placement. Celui-ci se situerait autour de 1300Mo/s.

Les résultats du ping-pong multirails présentés figure 2.4.2 créditent également cette hypothèse. Notons l'impact considérable qu'a le placement pour un tel débit, puisqu'on perd près de 700 Mo/s soit environ 40% du débit maximum. Dans ce cas, les performances sont pires que celles du test "mono-rail" sur le réseau Myri-10G. Une telle baisse de performance pourrait être favorisée par une contention au niveau du lien HyperTransport reliant les contrôleurs d'entrées-sorties, car sur notre plateforme de test, les deux contrôleurs y sont attachés, comme illustré sur le schéma de topologie figure 2.2 page 15.

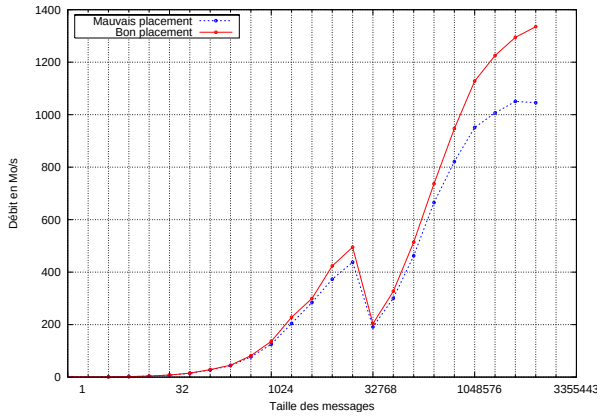
2.5 Autres phénomènes observés

2.5.1 Effets dissymétriques

Les tests applicatifs présentés précédemment montrent que l'impact du placement NUMA sur le débit est "visible" au delà d'un certain seuil, et est d'autant plus important que le débit est élevé. Ces tests sont symétriques et ne donnent par conséquent aucune information sur la dissymétrie observée dans les tests de la section 2.4.2.



(a) Myri-10G



(b) InfiniBand

Fig. 2.4 – Débit de ping-pong sur les réseaux Myri-10G et InfiniBand.

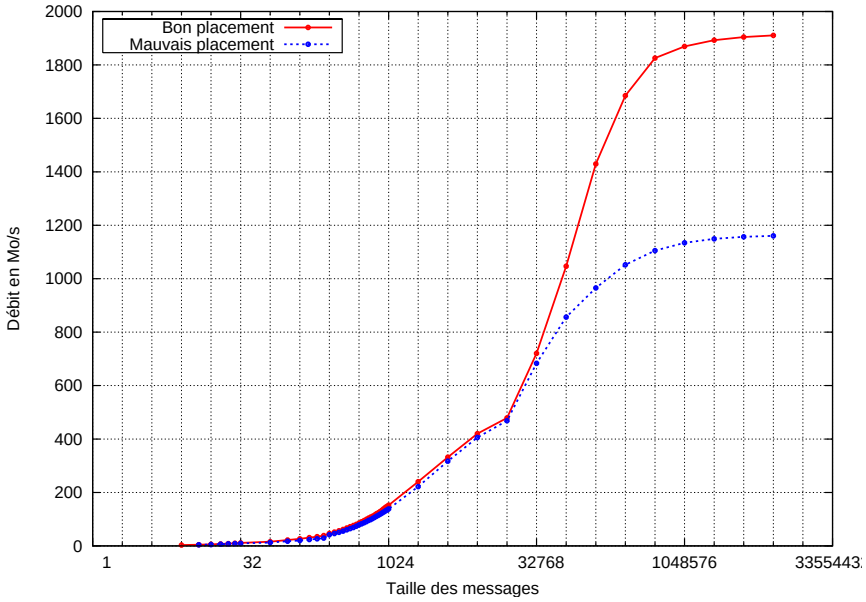


Fig. 2.5 – Débit d'un ping-pong multirails sur les réseaux Myri-10G et QsNet II.

Flux unidirectionnel multirails

Nous avons conçu un test applicatif émettant un flux continu unidirectionnel multirails sur la bibliothèque de communication NewMadeleine. Alors que le ping-pong multirails décrit précédemment (2.4.2) consistait en des transmissions de même taille alternant les 2 directions, ce test concentre les transferts dans une unique direction. La figure 2.6 en présente les résultats.

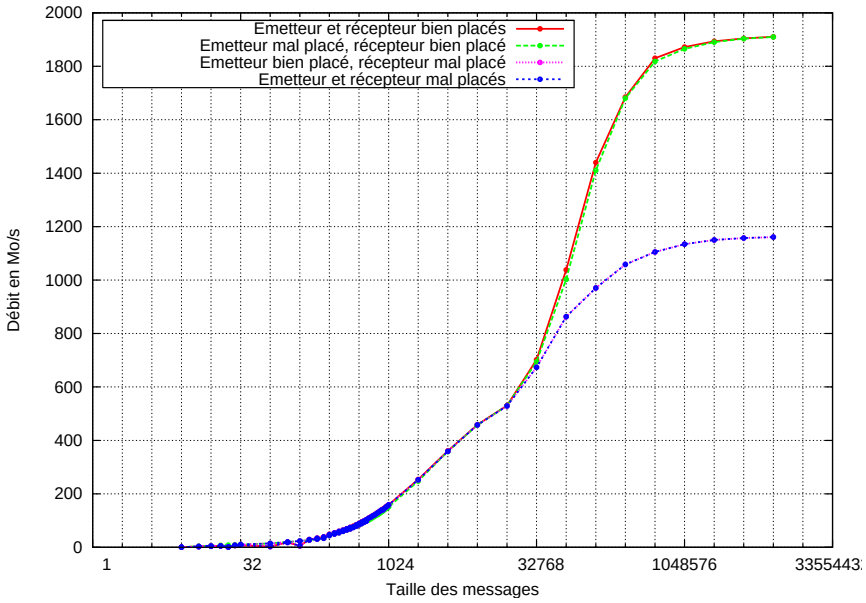


Fig. 2.6 – Effet du placement NUIOA sur le débit d'un flux continu unidirectionnel multirails : mise en évidence d'effets dissymétriques.

Celle-ci met en valeur la dissymétrie de l'impact du placement. En effet, la place seule du récepteur importe.

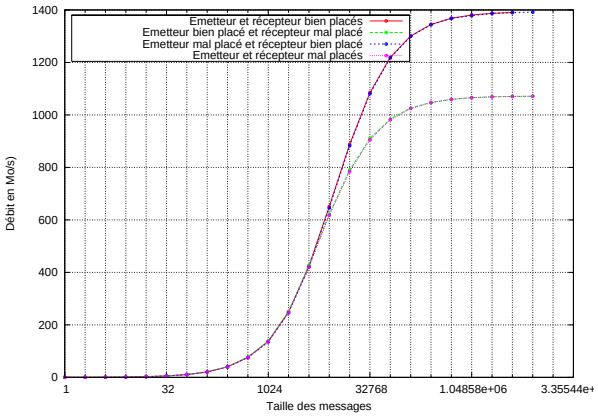
Flux unidirectionnel de RDMA

Le test précédent utilisait des communications de type send/receive en multirails sur les réseaux Myri-10G et QsNet. Nous avons poursuivi par l'étude d'un flux unidirectionnel utilisant les capacités d'écriture en mémoire distante (RDMA) du réseau InfiniBand.

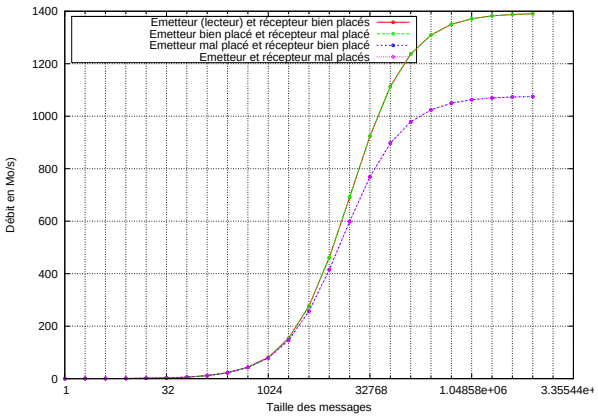
Le test en écriture présente un résultat similaire à celui du précédent : seul l'emplacement du récepteur importe. Par contre, les résultats de ce test en lecture montrent que c'est dans ce cas l'emplacement du récepteur, (du lecteur), qui produit un effet. Il semblerait ainsi que ce soit le placement des buffers dans lesquels les données sont placées (celui de l'émetteur pour une lecture, et celui du récepteur dans le cas d'une écriture) et des threads qui leurs sont associés qui soit déterminant sur les effets du débit.

2.5.2 Variation de l'effet avec l'augmentation de la distance

Nous avons cherché à étendre les résultats en reproduisant notre test de RDMA pour observer ce qui ce passe lorsque la distance par rapport à la carte réseau augmente. Des accès mémoires sont effectués en écriture sur les huit nœuds de la machine *Hagrid*, depuis l'une des machines *Infini*.



(a) RDMA en écriture



(b) RDMA en lecture

FIG. 2.7 – Variation de débit lors de transferts RDMA entre deux machines à 2 nœuds NUMA, sur le réseau INFINIBAND.

Résultats

Contrairement au tests de latence présentés en section 2.3.1, les résultats présentés en figure 2.8, ne sont pas du tout représentatifs de la hiérarchie de la machine. En effet il y a très peu de variations du débit avec l'augmentation de la distance, le meilleur placement mis à part. Concrètement, il y a un bon placement, et tout autre implique une dégradation constante du débit, quelque soit le nombre de liens traversés.

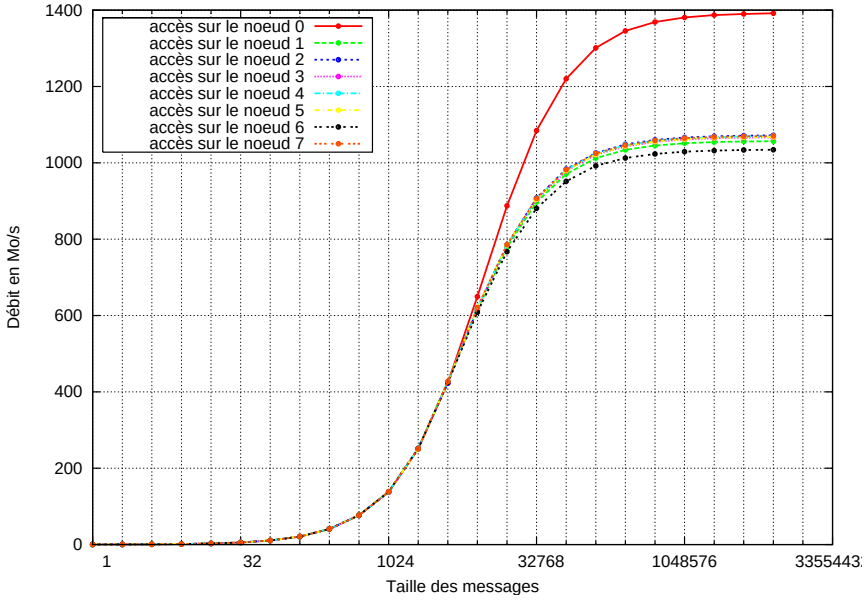


FIG. 2.8 – Variation de débit lors d'écritures RDMA sur les différent nœud de la machine NUMA 8 nœuds *Hagrid*, sur le réseau INFINIBAND.

Cette expérience montrerait à première vue l'inefficacité d'un placement hiérarchique sur le débit. Cela impliquerait que le processus de communication doit être impérativement lié sur le nœud NUMA le plus proche de la carte réseau, et qu'en cas de concurrence pour l'accès à la carte réseau, une seule tâche peut être privilégiée et correctement placée, tandis que le placement des autres n'influencerait pas les performances de débit. Il ne semble cependant pas réaliste de se fier à cette hypothèse. Si aucune dégradation de débit n'a été observée avec la variation du nombre de liens HYPERTRANSPORT traversés, il faut prendre en compte que les mesures ont été prises sur une machine non chargée. En augmentant la charge de travail sur la machine, il est très probable que des congestions créent, avec l'éloignement des processus de communication de l'interface réseau, des chutes de performances tant pour les calculs effectués que pour le débit des communications.

2.5.3 Influence des congestions mémoire

Nous avons reproduit notre test de RDMA sur INFINIBAND, tout en produisant des écritures continues en mémoire pour créer des contentions sur certains liens de la machine, de sorte à simuler les effets d'une charge de travail sur la machine. La figure 2.9 présente un exemple de perturbation étudiée.

Les résultats observé sous différents types de perturbations sont résumés par le tableau 2.3. Comme nous l'avons supposé, la contention entraîne une chute du débit tant sur les écritures en mémoire que sur le débit des communications, que ce soit des écritures en mémoire locale ou distante, dans la même direction sur le bus HYPERTRANSPORT ou non. En conditions réelles,

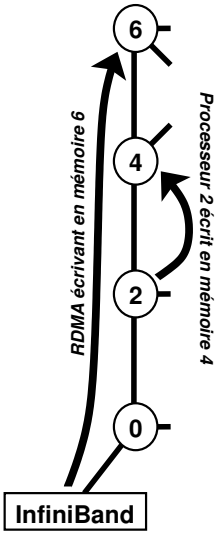


FIG. 2.9 – Perturbation d'un RDMA écrivant depuis le réseau en mémoire du nœud 6 par une écriture du processeur du nœud 2 en mémoire 4 sur la machine *Hagrid*.

		Débit processeur	Débit RDMA
RDMA en mémoire 6, seul			1049
Processeur 2 écrit en mémoire 4	seul	2171	
	pendant RDMA	1232	883
Processeur 4 écrit en mémoire 2	seul	2177	
	pendant RDMA	1611	971
Processeur 2 écrit en mémoire 2	seul	2740	
	pendant RDMA	2040	1002
Processeur 4 écrit en mémoire 4	seul	2730	
	pendant RDMA	1616	859

TAB. 2.3 – Impact d'une écriture en mémoire par un processeur sur le débit d'un transfert RDMA entrant sur la machine *Hagrid* (débits en Mo/s). Un processeur d'un nœud NUMA écrit dans la mémoire d'un autre nœud NUMA pendant que le RDMA écrit depuis le réseau en mémoire 6.

c'est-à-dire lorsqu'une application charge la machine et utilise donc intensément le bus mémoire, un placement des tâches de communication éloigné du périphérique réseau tend donc bien à réduire les performances de communication davantage qu'un placement plus proche du périphérique. La notion de placement "hiérarchique" n'est donc pas à exclure. Toutefois ces résultats restent difficiles à expliquer en détail, par exemple la différence entre les impacts des accès locaux sur 2 nœuds différents. La mise en place d'un placement hiérarchique est donc un problème délicat qui nécessite des études supplémentaires pour être envisagée.

2.6 Bilan

Les expériences menées pour évaluer les effets de divers placements NUMA des tâches de communication et des zones mémoire qui leur sont associées, ont mis en évidence plusieurs aspects intéressants.

Nous retiendrons tout d'abord que les effets d'un mauvais placement NUMA se font ressentir sur le débit à partir d'un certain seuil, qui correspond sans doute au débit à partir duquel les liens sont saturés. Lorsque la machine n'est pas chargée, l'intérêt de prendre soin du placement des processus de communication ne prend son sens que pour des débits très élevés. Toutefois, la perte, quand elle existe, est d'autant plus forte que le débit est élevé et peut prendre des proportions considérables.

Nous avons remarqué une dissymétrie quant à l'importance du placement des zones mémoire, puisque seul le placement des zones mémoire dans lesquels les données sont déposées s'avère avoir une forte influence. Cet aspect est potentiellement dû à un choix de configuration au niveau matériel. Il pourrait être un cas particulier visible sur cette génération de matériel mais non représentatif de l'ensemble des machines NUMA, et qui pourrait être corrigé modifié dans les prochaines architectures.

Enfin, il est notable qu'à première vue, l'augmentation du nombre de liens séparant un processus de communication de la carte réseau n'affecte pas le débit. Cependant, la présence d'une charge de travail sur la machine dément ce constat, et prouve que la chute de performances est conséquente tant pour le débit des communications que pour les performances de calcul. On peut donc envisager de tenir compte des distances précises, voire de la topologie globale de la machine, pour le placement des processus de communication.

Chapitre 3

Placement automatique et découverte de topologie

Nous avons prouvé dans le chapitre précédent que le placement des tâches de communication et des tampons mémoire associés a un impact sur les performances de communication. Celui-ci s'avère relativement faible sur la latence, mais peut prendre des proportions considérables sur le débit. Pour garantir des communications rapides et efficaces, qualités indispensables pour les transmissions de messages dans les grappes de calcul, il est donc important de veiller à placer les processus responsables au plus près des périphériques réseau.

Des outils sont utilisés pour placer ces processus "à la main" lors des mesures performances. Mais cette méthode oblige dans un premier temps l'utilisateur à trouver les informations concrètes quant au placement physique des cartes, pour lesquelles il n'existe aucun standard, comme le montre par exemple notre plateforme de test, puisque les cartes sont placées sur le nœud 0 pour les machines *Dalton*, mais sur le nœud 1 sur les *Infini*. Il n'y a donc pas par cette méthode de portabilité directe des performances.

Il est également nécessaire de "verrouiller" le processus pour éviter une éventuelle migration de celui-ci provoquée par l'ordonnanceur, possible par exemple dans le cas où un démon se réveille en arrière plan. De plus, le placement "à la main" ne peut pas convenir aux cas plus complexes, comme par exemple pour des applications disposant de plusieurs threads, dont certains seulement effectuent des communications.

Devant toutes ces contraintes, l'intérêt d'un placement automatique des tâches de communication prend son sens. Il permettrait de ne pas contraindre l'utilisateur à "chercher" comment placer correctement ses threads ou processus, et cela pour chaque nouvelle grappe. Ce placement pourrait être engendré par les intergiciels, et l'optimisation des performances de communication et de leur portabilité pourraient ainsi être prises en charge sans que l'utilisateur n'ait à intervenir.

3.1 Trouver le nœud le plus proche de chaque carte réseau

Pour envisager un placement automatique, il faut dans un premier temps savoir à quels nœuds NUMA sont attachés les périphériques. Pour cela, nous avons envisagé plusieurs méthodes.

Il est possible d'obtenir cette information, en utilisant des tests de latence. En effet, les expériences présentées dans la section 2.3 permettent d'évaluer la distance entre les nœuds et l'interface réseau. Il est ainsi possible de trouver le nœud NUMA auquel celle-ci est connectée. Notons qu'il est préférable d'utiliser des mesures de latence plutôt que des mesures de bande

passante car nous avons constaté qu'il est difficile de traduire clairement ces dernières en informations de topologie.

L'utilisation des tests de latence pose toutefois plusieurs points délicats. Tout d'abord, ils n'ont de valeur que s'ils sont effectués sur une machine dépourvue de toute autre charge de travail, ce qui peut être une condition difficile à obtenir, d'autant plus qu'il faut refaire les mesures pour chaque carte réseau. Plus contraignant encore, certains de ces tests peuvent nécessiter des droits privilégiés, car ils peuvent être amenés à effectuer des opérations potentiellement dangereuses (PIO de test), et ne sont donc pas à la portée de tout utilisateur.

Un autre moyen d'obtenir des informations consiste à interroger le système d'exploitation. En effet, celui-ci sait parfois où sont placés les cartes réseau sur la machine, et pourrait ainsi renseigner les applications sur les informations qu'il possède. Toutefois, tous les systèmes d'exploitation ne proposent pas de telles informations. Dans le système d'exploitation LINUX, des fichiers spéciaux (dans `sysfs`) exposent quels cœurs de processeurs sont près des périphériques (par un masque de bits).

Il serait ainsi intéressant de récupérer les informations disponibles à partir du système, puis de les transmettre à l'intergiciel qui se chargerait alors de verrouiller les threads et les buffers (en utilisant `libnuma` par exemple). Notons que l'ordonnanceur MARCEL [11] utilise déjà une méthode similaire pour placer les threads. L'idéal serait d'étendre cette méthode à toutes les ressources de la machine.

Cette proposition pose une difficulté majeure : les attributs fournis par le système de fichiers ne sont pas directement utilisable comme information de placement au niveau applicatif. En effet, l'application ou l'intergiciel manipule des structures de haut niveau décrivant des canaux de communication parfois virtuels dans le interfaces réseau physiques. Il est donc nécessaire de traduire ces descripteurs de haut niveau en périphériques physiques, pour pouvoir enfin utiliser les informations de placement NUMA fournies par le système d'exploitation.

Cette correspondance peut être faite à partir de fichiers spéciaux fournis par certains pilotes de périphériques. C'est par exemple le cas pour le réseau INFINIBAND, ce qui permet de traduire facilement un descripteur de canal de communication INFINIBAND en une carte réseau physique et d'obtenir ensuite son placement NUMA. Pour faciliter l'utilisation de `libnuma`, nous avons rajouté dans la version 2.6.21 du noyau LINUX une extention pour récupérer un numéro de nœud NUMA directement exploitable.

Par contre, pour d'autre réseaux, le système ne donne aucun indice pour effectuer cette traduction. C'est notamment le cas pour MYRI-10G, QSNET et ETHERNET. Dans le cas de MYRI-10G, nous avons dû rajouter un support explicite dans le pilote MX (*Myrinet Express*), pour obtenir directement l'attribut NUMA associé à un canal de communication MX manipulé par l'application. A priori, pour les autres technologies de réseaux (QSNET, ETHERNET, ...) nous ne disposons pas pour le moment de moyen d'obtenir l'emplacement de la carte réseau à partir du système.

L'idéal serait de disposer d'une méthode uniforme pour tous les réseaux, mais ceux-ci n'exposent pas les mêmes structures et descripteurs aux applications. Il faudrait donc ajouter une couche de virtualisation dans l'intergiciel, comme on le fait par exemple déjà dans pour uniformiser les communications de différents réseaux rapides dans les intergiciels MPI.

3.2 Placement automatique

Devant l'importance des effets NUIOA (*Non Uniform Input/Output Access*), nous avons cherché à exploiter les résultats précédents pour obtenir automatiquement les performances optimales de communication. Pour cela, à partir des quelques informations de topologie dont nous

disposons, nous avons mis en place un placement adaptatif dans la bibliothèque de communication rapide pour les grappes NewMadeleine [10].

3.2.1 Principe

Ce placement est basé sur un fonctionnement simple. Une fois que l'on est arrivé à obtenir la position physique des périphériques réseaux, il faut indiquer la décision de lier les threads et la mémoire associée sur le nœud choisi.

Cette opération doit être faite suffisamment tôt dans l'exécution, pour éviter que les threads ne soient déjà créés, ou aient déjà alloué de la mémoire sur un nœud. Cela exigerait de faire migrer tâches et mémoire, action coûteuse et potentiellement problématique. En effet,le système peut être amené à verrouiller des pages en mémoire physique, pour garantir des correspondances d'adresses, nécessité pour des transferts zéro-copie et DMA, engendrés par exemple pour des communications RDMA.

Il est également important d'attendre pour fixer une tâche d'avoir évalué toutes les possibilités relatives au placement sur chaque nœud doté d'un contrôleur d'entrée/sorties. Cela implique d'avoir obtenue toutes les indications disponibles sur la position des cartes, pour ne pas choisir un nœud NUMA, alors qu'une information de placement plus intéressante n'a pas encore été soumise. Il faut donc commencer par interroger tous les pilotes, pour récupérer les nœuds des périphériques réseaux lorsque l'information est disponible, et ensuite, faire un choix parmi les différentes possibilités, s'il y a lieu. Il est alors enfin possible de lier la mémoire et les threads, puis d'initialiser les pilotes.

3.2.2 Mise en œuvre dans NewMadeleine

Nous avons implémenté le placement automatique dans la bibliothèque NewMadeleine. Les résultats sont concluants : on obtient bien toujours les meilleures performances, même quand on effectue un placement préalable "à la main", sur le mauvais nœud, le placement est automatiquement rectifié par la bibliothèque de manière transparente.

Pour les envois de message sur les réseaux Myri-10G et InfiniBand, NewMadeleine obtient le nœud le plus proche de l'interface réseau, les threads sont automatiquement placés sur celui-ci. Pour les autres réseaux, qui ne fournissent pas d'information de placement, NewMadeleine considère qu'il n'y a pas de nœud plus proche du périphérique et n'intègre pas d'information pour le choix de placement.

Dans le cas ou aucune information n'est connue, aucun placement n'est imposé. Par contre, si des informations conflictuelles sont relevées, une décision de placement nécessaire. Celle-ci est faite à partir d'une pondération de la latence et du débit des réseaux, déterminée en fonction leur latence/débit théoriques. Pour l'instant, le domaine prépondérant (latence ou bande passante) est choisi par une intervention extérieur, mais il s'agit d'une méthode temporaire. A terme, notre objectif est d'utiliser des indices donnés par l'application pour influencer le choix du réseaux à favoriser.

Il faudra également en venir à tenir compte des performances réseau, mais aussi des besoins de l'application, notamment de la prépondérance des calculs ou des communications.

3.3 Découverte de la topologie générale

3.3.1 Intérêts

La section 2.5.3 met en évidence les problèmes qui surviennent sur les effets NUIOA avec la présence d'une charge de travail sur la machine. Les performances de latence et de débit

des communications, mais également les performances de calcul générales sont diminuées par l'éloignement des tâches de communication par rapport aux interfaces réseau.

On peut donc envisager un placement "palliatif", lorsque la machine présente une charge de calcul à laquelle viennent s'ajouter plusieurs tâches de communications qui entrent en concurrence pour l'accès aux contrôleurs d'entrée/sortie, tandis que la répartition de la charge de travail sur la machine induit leur dispersion sur plusieurs nœuds. Ce placement favoriserait le rapprochement des tâches de communication des interfaces réseau, en les plaçant sur *les* nœuds les plus proches de l'interface. Sur des machines hiérarchiques complexes, on placerait ainsi une tâche de communication sur le nœud de l'interface réseau, puis les suivantes sur les nœuds séparés de celui-ci par un seul lien, et ainsi de suite pour obtenir un placement hiérarchique.

Dans le cas d'applications MPI classiques, les contentions sont inévitables dans la mesure où l'on répartit l'ensemble des processus sur l'ensemble des cœurs pour équilibrer la charge de travail. Les effets NUIOA importent en fait uniquement si les communications sont irrégulières entre les différents processus, ou si les processus ne peuvent pas exploiter tous les cœurs simultanément (cas peu courants).

Cependant, dans des applications plus complexes où certaines tâches sont dédiées au calcul tandis que d'autres sont dédiées aux communications vers les autres nœuds de la grappe, par exemple en cas d'utilisation conjointe d'OpenMP [12] et MPI [13], une connaissance avancée de la topologie matérielle permet d'optimiser le placement. En effet, l'ensemble des cœurs est exploité lors du déploiement des sections parallèles OpenMP au cours du calcul. Mais les communications externes MPI sont confiées à un nombre réduit de tâches, en particulier entre les sections parallèles, qui doivent être placées à proximité des périphériques réseaux. Il est dans ce cas extrêmement avantageux de placer ces derniers proches des cartes réseau. Cela permet notamment d'éviter de créer des contentions supplémentaires si les données avaient à transiter au travers des cœurs effectuant des calculs avant d'être envoyées sur le réseau.

Pour mettre en place un placement palliatif, il est nécessaire de posséder des informations sur la topologie physique de la machine, comme par exemple la distances entre les nœuds, ou encore les liens physiques existants. Ces informations permettraient notamment de pouvoir trier les nœuds NUMA en fonction de la distance par rapport aux interfaces de chaque réseau. Nous avons donc travaillé sur des moyens d'obtenir de telles informations.

3.3.2 Découverte de topologie

Découvrir la topologie d'une machine est un problème difficile. Le matériel exporte peu d'informations et le système d'exploitation ne donne que des indications limitées. On peut ainsi obtenir le nombre de nœuds de la machine à partir du système d'exploitation, mais la distance entre les nœuds NUMA est inconnue. Il n'existe donc pas de moyens simples d'obtenir des renseignements pertinent.

Comme nous avons expliqué précédemment, pour obtenir des informations de topologie, les mesures de latence sont plus facilement exploitables que celles de débit. Nous avons donc mis en place des tests basés sur des mesures de latence. Nous avons ainsi mesuré le temps d'accès mémoire (contournant le cache), entre tous les processeurs et les bancs mémoire de la machine *Hagrid*, dont la topologie est rappelée sur la figure 2.2.

Les différences (en nanosecondes), de temps d'accès à chaque banc mémoire avec le temps d'accès à la mémoire locale sont présentées dans le tableau 3.1

On remarque que les différences observées sont pour la plupart des multiples de environ 34 ns. Cette valeur correspond à peu près au coût estimé pour traverser un lien HyperTransport en section 2.3. Chaque différence nous permet ainsi de compter le nombre de liens traversés. Les latences les plus faibles ($\simeq 34ns$), présentées en gras sur le tableau représentent ainsi les liens

	M0	M1	M2	M3	M4	M5	M6	M7
C0	0	**37**	**35**	67	67	*133*	102	*100*
C1	**37**	0	67	**35**	*134*	66	*97*	97
C2	**35**	69	0	**36**	**35**	*101*	69	*67*
C3	69	**34**	**35**	0	*102*	**33**	*66*	68
C4	67	*101*	**35**	*67*	0	69	**38**	**37**
C5	*100*	65	*66*	**34**	71	0	**35**	**37**
C6	102	*115*	69	*83*	**37**	**35**	0	**35**
C7	*115*	99	*83*	67	**37**	**37**	**35**	0

TAB. 3.1 – Surcoût des accès à chaque banc mémoire (en nanosecondes) par rapport à l'accès à la mémoire locale, depuis tous les processeurs de la machine 8 nœuds *Hagrid*.

directs entre les nœuds. On observe bien une symétrie : si la latence est minime pour l'accès à la mémoire d'un nœud x depuis un nœud x, traduisant un lien direct entre x et y, elle est minime pour l'accès sur y depuis x. Dans notre expérience, on observe 22 latences minimums qui correspondent bien aux 11 liens de la topologie de la machine *Hagrid*. Notons que chaque processeur possède trois liens HYPERTRANSPORT, soit au total 24 liens pour les huit nœuds de *Hagrid*, dont 22 sont détectés dans notre tableau, il en reste donc deux sur lesquels peuvent être placés des contrôleur d'entrée/sortie.

Pour les latences plus élevées (plusieurs liens traversés), on remarque des dissymétries en italiques et quelques "demi-multiples". La table de routage HYPERTRANSPORT de la machine nous a permis d'expliquer ces valeurs. En effet, celle-ci relate des choix de route différents pour l'accès à un nœud x depuis un nœud y, et l'accès au nœud y depuis le nœud x. Ces différences sont à l'origine des dissymétries. Par exemple, le chemin emprunté pour accéder du nœud 0 au nœud 5 (quatre liens traversés), n'est pas celui utilisé pour l'écriture du nœud 5 sur le nœud 0 (trois liens traversés). Les demi-multiples sont dûs à des variations de chemin à l'intérieur même des requêtes : la "demande d'écriture" ne passe pas par le même chemin que la "notification d'écriture".

Notons que les informations de routage constituant la table, sont composées de valeurs appartenant à une série de registres très spécifiques au matériel [14], et complexes à traduire en informations exploitables par l'application. Il est donc impossible de les utiliser de manière générique pour de acquérir des informations de topologie. L'étude de cette table à été faite dans le but d'expliquer et de valider nos résultats.

Bien que le matériel et le système d'exploitation ne donnent que peu d'informations à propos de la topologie interne, des tests nous ont permis de retrouver tous les liens HYPERTRANSPORT reliant les nœud NUMA de la machine *Hagrid*. Il semble ainsi possible d'envisager un placement qui tiendrait compte de l'éloignement progressif des nœuds par rapport à la carte réseau. Cela permettrait de minimiser les contentions sur les liens, par plusieurs tâches de communication concurrentes (qui doivent rester près du réseau) et les flots de calcul (dont le placement importe peu).

Conclusion

Bilan

Dans le domaine du calcul haute performance, les grappes de calcul prédominent mais leurs nœuds deviennent de plus en plus complexes. La généralisation des architectures SMP munies de processeurs multi-cœurs conduit à la nécessité de distribuer la mémoire pour éviter le goulet d'étranglement des bus centralisés, favorisant ainsi la démocratisation des architectures NUMA. Ces architectures complexes peuvent entraîner une iniquité d'accès aux contrôleur d'entrées-sorties, et les performances des communications se voir ainsi amoindries.

Nous avons étudié dans ce mémoire ces effets dans le but d'évaluer leur impact et d'élaborer des solutions pour préserver l'efficacité des communications, qui sont un facteur majeur dans l'obtention de hautes performances pour le calcul scientifique sur grappe. Nous avons présenté une évaluation de l'impact du placement des tâches et des données dans les machines NUMA sur les performances des communications sur réseaux rapides.

Nous avons notamment relevé une influence NUIOA sur la latence, se traduisant par une légère augmentation lorsque la distance de l'interface réseau au processeur augmente, en particulier lorsque des liens HyperTransport additionnels doivent être traversés. Le surcoût de quelques dizaines de nanosecondes par lien traversé peut constituer un pourcentage conséquent de la latence d'un réseau très rapide, mais il reste de l'ordre d'un défaut de cache, ce qui représente une perte acceptable dans la plupart des cas.

Nous avons par contre observé des effets importants sur le débit lorsqu'il devient suffisamment important (plus d'1 Go/s). La perte liée au mauvais placement augmente avec le débit observé et peut prendre des proportions considérables, avec notamment une chute de près de 40 % pour un débit attendu de 2 Go/s. Nous avons conclu de nos résultats que la dégradation de performances est probablement liée à une saturation des liens mémoire.

Les variations de placement ont par ailleurs mis en évidence des répercussions dissymétriques puisque les dégradations du débit se sont avérées plus importantes pour les transferts depuis la carte réseau vers l'hôte, que pour ceux dans le sens inverse. Cela traduit le caractère critique du choix des zones mémoires où les données sont déposées. Bien que ce résultat semble d'importance, il doit être relativisé car semble lié à la configuration intrinsèque du matériel qui pourrait être améliorée dans les futures générations de processeurs et bus mémoire.

Nous avons enfin remarqué que l'accroissement de la distance par rapport à la carte réseau ne continuait pas à dégrader les performances en l'absence de congestion mémoire. Par contre, une machine chargée perturbe beaucoup plus des communications très mal placées en raison de contentions mémoire.

Ces observations ont mis en évidence la nécessité de prendre soin du placement des ressources de communication, pour garantir des transmissions de messages rapides et efficaces, qualités indispensables dans le domaine des grappes de calcul. Elles nous ont amenés à envisager la nécessité de proposer un placement automatique de ces ressources, pour pallier les difficultés de

placement manuel (jusqu'à présent laissé aux utilisateurs) et garantir la portabilité immédiate des performances.

Nous avons mis en évidence les obstacles liés à l'acquisition d'informations concernant la position physique des périphériques réseau dans la machine, puis avons mis en œuvre une stratégie basée sur ces indications permettant un choix de placement et son automatisation, dans la bibliothèque de communication pour grappes de calcul NewMadeleine.

Nous avons enfin présenté des expériences destinées à obtenir des informations détaillées sur la topologie interne de la machine dans le but d'élaborer une solution de placement adaptatif pour les applications complexes mélangeant de nombreuses tâches de calcul et quelques tâches dédiées aux communications.

Le travail présenté dans ce mémoire à donné lieu a la soumission d'un article pour publication dans une conférence internationale [15] et nous amène à envisager de nombreuses perspectives.

Perspectives

Pour valider nos résultats sur l'ensemble des machines NUMA, nous projetons d'approfondir nos travaux par l'étude des impacts sur d'autres architectures. Nous souhaiterions notamment travailler sur des machines à base de processeur Itanium 2 qui présentent des bus mémoire hiérarchiques et plusieurs bus d'entrées-sorties, comme par exemple les serveurs NovaScale de Bull. Par ailleurs, on s'attend à une généralisation des architectures distribuées de type AMD HyperTransport puisque Intel a annoncé l'arrivée prochaine d'une technologie équivalente (CSI, *Common System Interface*) qu'il nous faudra étudier.

Nous prévoyons également d'observer les effets NUIOA sur les communications via d'autres technologies réseau, en particulier Ethernet pour lequel le placement des interruptions est connu pour être critique, ou encore Infinipath dont la connexion directe sur HyperTransport pourrait engendrer des effets NUIOA singuliers.

Nous envisageons par ailleurs un affinement de nos recherches en étudiant précisément les effets du placement sur différents modes de communications (PIO, copie et DMA) qui diffèrent notamment par le taux d'implication des différentes ressources (processeur en PIO, mémoire en DMA) et dont les contraintes de placement peuvent ainsi varier. Il pourrait également être intéressant d'étudier plus en détails les effets de la congestion mémoire sur les performances des communications.

L'implémentation du placement automatique dans la bibliothèque de communication NewMadeleine et la découverte de topologie constituent un premier pas dans la conception d'un placement adaptatif intelligent. À terme, ces informations de placement et de topologie devraient permettre de gérer le placement des tâches dans des applications complexes, disposant de plusieurs tâches de communication et de flots de calculs, par exemple en cas d'utilisation conjointe de OpenMP et MPI. Des conseils fournis par l'application devraient pouvoir assister les décisions de placement en indiquant par exemple que l'on souhaite favoriser la latence, le débit ou la disponibilité du processeur. Un de nos principaux objectifs sera donc d'intégrer des informations de placement NUIOA dans un ordonnanceur, tel que Marcel/BubblesSched [11], pour pouvoir guider les threads de communications vers les interfaces réseau, adaptant ainsi l'ordonnancement à la topologie des systèmes d'entrées-sorties.

Bibliographie

[1] Top500 Supercomputing Sites. `http://top500.org/`.

[2] HyperTransport I/O Link Specification. `http://www.hypertransport.org/`.

[3] Joseph Antony, Pete P. Janes, and Alistair P. Rendell. Exploring Thread and Memory Placement on NUMA Architectures : Solaris and Linux, UltraSPARC/FirePlane and Opteron/HyperTransport. In *Proceedings of the International Conference on High Performance Computing (HiPC)*, Bangalore, India, December 2006.

[4] Timothy Brecht. On the Importance of Parallel Application Placement in NUMA Multiprocessors. In *Proceedings of the Fourth Symposium on Experiences with Distributed and Multiprocessor Systems (SEDMS IV)*, San Diego, CA, Sept 93.

[5] Martin Steckermeier and Frank Bellosa. Using locality information in userlevel scheduling. Technical Report TR-95-14, University of Erlangen-Nürnberg – Computer Science Department – Operating Systems – IMMD IV, Martensstraße 1, 91058 Erlangen, Germany, December 1995.

[6] Rohit Chandra, Scott Devine, Ben Verghese, Anoop Gupta, and Mendel Rosenblum. Scheduling and page migration for multiprocessor compute servers. In *Proceedings of the sixth international conference on Architectural support for programming languages and operating systems table of contents*, pages 12–24, San Jose, CA, 1994.

[7] Sadaf R. Alam, Richard F. Barrett, Jeffery A. Kuehn, Philip C. Roth, and Jeffrey S. Vetter. Characterization of Scientific Workloads on Systems with Multi-Core Processors. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*, San Jose, CA, 2006.

[8] Andreas Kleen. A NUMA API for LINUX, April 2005.

[9] Leon Arber and Scott Pakin. The impact of message-buffer alignment on communication performance. *Parallel Processing Letters*, 15(1) :49–65, March 2005.

[10] Olivier Aumage, Elisabeth Brunet, Guillaume Mercier, and Raymond Namyst. High-Performance Multi-Rail Support with the NewMadeleine Communication Library. In *Proceedings of the Sixteenth International Heterogeneity in Computing Workshop (HCW 2007), held in conjunction with IPDPS 2007*, Long Beach, CA, March 2007.

[11] Samuel Thibault. A Flexible Thread Scheduler for Hierarchical Multiprocessor Machines. In *Proceedings of the Second International Workshop on Operating Systems, Programming Environments and Management Tools for High-Performance Computing on Clusters (COSET-2)*, Cambridge, MA, June 2005.

[12] OpenMP : Simple, Portable, Scalable SMP Programming. `http://openmp.org/`.

[13] Message Passing Interface Forum. MPI : A message-passing interface standard. Technical Report UT-CS-94-230, 1994.

[14] Advanced Micro Devices, Inc. BIOS and Kernel Developer's Guide for AMD Athlon 64 and AMD Opteron Processors, February 2006.

[15] Stéphanie Moreaud and Brice Goglin. Impact of NUMA Effects on High-Speed Networking with Multi-Opteron Machines. Submitted to the *IEEE Cluster 2007 Conference*, Austin, TX, September 2007.