Analyse et modélisation des communications concurrentes dans les réseaux haute performance

Maxime Martinasso

UNIVERSITÉ JOSEPH FOURIER DE GRENOBLE

N° attribué par la bibliothèque

|_|_|_|_|_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER DE GRENOBLE

Spécialité : "Informatique : Systèmes et Communications"

PRÉPARÉE AU LABORATOIRE D'INFORMATIQUE DE GRENOBLE
DANS LE CADRE DE ***l'École Doctorale "Mathématiques, Sciences et Technologies de l'Information, Informatique"***

PRÉSENTÉE ET SOUTENUE PUBLIQUEMENT

PAR

MAXIME MARTINASSO

LE 25 MAI 2007

ANALYSE ET MODÉLISATION DES COMMUNICATIONS CONCURRENTES DANS LES RÉSEAUX HAUTE PERFORMANCE

Directeurs de thèse :
M Jean-François Méhaut
Mme Pascale Rossé-Laurent

JURY

M JEAN-CLAUDE FERNANDEZ	Univ. Joseph Fourier, Grenoble	Président
M FRÉDÉRIC DESPREZ	ENS Lyon / INRIA	Rapporteur
M RAYMOND NAMYST	Univ. Bordeaux 1	Rapporteur
MME PASCALE ROSSÉ-LAURENT	Bull SAS	Examinateur
M ARNAUD LEGRAND	CNRS, Grenoble	Examinateur
M JEAN-FRANÇOIS MÉHAUT	Univ. Joseph Fourier, Grenoble	Examinateur (directeur de thèse)

Il n'y a rien de constant, si ce n'est le changement.
Bouddha

Remerciements

Une thèse consacre trois années de travail et de recherche, travaux qui sont réalisés grâce à la générosité, à l'amitié et au savoir de plusieurs personnes.

Je tiens à remercier la société BULL SAS pour avoir soutenu les travaux de recherche de cette thèse.

Je voudrais remercier les membres du jury pour avoir accepté d'évaluer ces travaux. Je tiens à remercier Jean-Claude Fernandez, professeur à l'université Joseph Fourier, pour m'avoir fait l'honneur de présider ce jury de thèse. Je voudrais, tout naturellement, exprimer mes remerciements à Frédéric Desprez, directeur de recherche INRIA, et à Raymond Namyst, professeur à l'université Bordeaux 1, pour leurs rapports détaillés témoignant de l'intérêt porté à ces travaux. Ayant été plusieurs fois en contact avec Raymond Namyst durant ces trois années, je voudrais, tout particulièrement, le remercier pour son enthousiasme et sa grande sympathie. Merci aussi à Arnaud Legrand, chargé de recherche CNRS, d'avoir accepté de participer à ce jury de thèse, mais également pour sa disponibilité et pour partager sa passion de la recherche.

Un grand merci à Mme Pascale Rossé-Laurent, ingénieur de recherche dans la société BULL et co-encadrante de ces travaux, pour son amitié et ses qualités humaines, pour avoir partagé ses connaissances technologiques et m'avoir apporté un regard extérieur du monde académique. Je tiens aussi à te remercier pour ta préoccupation constante du devenir de tes thésards et des différents échanges qui m'ont permis de traverser le long désert que peut parfois être la recherche scientifique.

Finalement, la seule personne sans laquelle ces travaux n'auraient jamais abouti est M Jean-François Méhaut, professeur à l'université Joseph Fourier et directeur de mes travaux de thèse. Je te remercie très chaleureusement pour m'avoir fait confiance, pour cette expérience grenobloise qui au départ était faite de tant d'inconnues. Je voudrais également te dire combien j'ai apprécié travailler et enseigner avec toi, je te remercie d'avoir parfaitement dirigé cette thèse, tellement bien qu'à quelques jours de ma soutenance je ne stresse même pas ! J'ai également énormément apprécié les valeurs morales dont tu as fait preuve tout le long de ces trois années, ta rigueur scientifique, et te serais toujours grè de toutes les connaissances que tu m'as apportées tout aussi bien sur le plan scientifique que humain (et même orthographique !). J'espère dans le futur que l'on puisse collaborer à nouveau.

Ah.. mes ami(e)s comment tous vous remercier sans écrire 200 pages supplémentaires ! A vous amis qui ont suivi le même périple durant trois ans, je vous suis redevable de tant de moment magique. A toi mon Lolo pour de si nombreuses discussions autour d'autant de Guinness ! A toi Habou pour m'avoir fait partagé l'amour de ton pays et pour ta vision d'une vie stable et simple. A toi "ma petite Coco" pour m'avoir montré le chemin et pour tous ses souvenirs de fou rire qu'il me reste (Tao ! 1.. 2.. 3.. ! !). A mon petit Léo qui cache beaucoup de bonnes choses dans son casier ;). Finalement, à tous les membres du Labo ID-IMAG (..oui ..oui le LIG je sais !) qui apportent chaque jour une bonne ambiance et un plaisir le matin de venir travailler au labo ! En particulier,

Titoux qui possède l'art de faire disparaître les bières... Une petite pensée particulière pour Jé, en lui souhaitant une bonne thèse et bonne traversée du désert ;) ! A mes amis de l'INRIA, Nico et Jé, pour toutes les heures de pratique des arts martiaux.

Je tiens également à remercier tous mes ami(e)s de la bande RIUP et leur nouvelle génération !

A tous mes amis suédois, merci pour toutes les fiestas parisiennes, une oasis dans le désert ! pour m'avoir hébergé tant de fois, et pour toutes les fois à venir ;)

Enfin, un grand merci à ma famille pour leur soutien et encouragements ! pour ce lien très fort qui nous unit !

Table des matières

Table des figures

Liste des tableaux

Introduction 1

La société *Intel* annonçait récemment la fabrication d'un processeur intégrant 80 cœurs (unité de traitement) dont la consommation énergétique n'excédait pas 62 Watts. Cependant, si au niveau matériel la fabrication d'un tel composant est réalisable, de nombreux problèmes se posent pour son utilisation dans une architecture de machine. Le fait d'avoir un aussi grand nombre de cœurs provoque un partage des autres composants de la machine entre les différentes requêtes des cœurs. A titre d'exemple, il est possible de citer le problème des accès concurrents à la mémoire, avec pour de tels processeurs, même des architectures à mémoire hiérarchique (*NUMA*) auront très rapidement des goulots d'étranglement. En ce qui concerne les communications réseaux, les cartes d'accès et les liens physiques devront être également partagés entre les différents cœurs.

Les applications parallèles provenant du calcul scientifique s'efforcent d'exploiter au maximum les ressources d'une architecture parallèle. Pour ce faire, elles utilisent la totalité des unités de traitement disponibles afin de rendre leur exécution la plus rapide possible. Une exécution simultanée de plusieurs processus sur une machine génère des comportements particuliers causés par les accès concurrents aux ressources. Pour évaluer les performances d'une application, il devient donc nécessaire d'identifier et de comprendre les effets du partage de ressources.

Ces nouveaux comportements de partage de ressources, ainsi produits, sont difficiles à interpréter et à prédire. Dans cette thèse nous étudions le problème du partage du réseau. Les grappes de calcul utilisent des réseaux dédiés à haute performance tels que *Gigabit Ethernet*, *Myrinet* ou *Quadrics*. Même si ces réseaux sont rapides, ils restent un des composants les plus lents comparés à la vitesse de calcul des processeurs. Le partage de ressource provoque des pertes d'efficacités globales qu'il convient d'analyser.

Pour entreprendre cette étude, il est possible d'évaluer les performances et les interactions des différents composants d'une grappe. Dans le cas du partage de la ressource réseau, les effets du contrôle de flux ou de la gestion des accès aux cartes réseaux doivent être examinés de manière approfondie par des observations et des mesures. Cette phase d'observation et d'analyse du comportement est essentielle pour l'élaboration d'un modèle prédictif de performance permettant de proposer une caractérisation réaliste des accès réseaux concurrents.

Cette modélisation se base sur l'évaluation et/ou la connaissance du comportement des technologies les plus couramment utilisées. Elle consiste bien souvent à la simplification des systèmes observés grâce à une représentation mathématique (équations, processus stochastiques,...) ou par une description algorithmique du système (simulation). La qualité du modèle se situe, par conséquent, dans sa capacité à représenter avec précision le comportement d'un système complexe.

Pour quantifier la qualité des modèles de partage de ressource, une phase d'évaluation des mo-

dèles est nécessaire. Cette évaluation peut être menée face à des exécutions réelles d'applications parallèles afin de montrer l'aspect pratique des modèles et leur utilité dans un contexte industriel.

1.1 Contexte scientifique et industriel

Cette thèse s'inscrit dans le cadre du projet de collaboration *LIPS* entre *BULL*, l'*INRIA* Rhône-Alpes et le laboratoire *LIG* (*ID-IMAG*).

Bull est un constructeur de machines parallèles et de grappes de calcul. Les secteurs d'activité de *Bull* comprennent : les systèmes de stockage, les services et le support, la sécurité informatique, et le calcul haute performance. *Bull* a conçu la solution *Tera10* retenue par le Commissariat à l'Énergie Atomique qui s'est classée en cinquième position dans le TOP500.

L'Institut National de Recherche en Informatique et en Automatique (*INRIA*) est un organisme public de recherche français. Son ambition est de mettre en réseau les compétences de la recherche française dans le domaine des sciences et technologies de l'information.

A travers le projet *LIPS*, *BULL* et l'*INRIA* joignent leurs efforts afin d'expérimenter des solutions adaptées aux grandes grappes de calcul. Les principaux thèmes abordés par le projet *LIPS* concernent l'amélioration des flux de calcul/données dans les grappes de calcul et l'élaboration d'un environnement logiciel complet (administration, exploitation, programmation) sur ces grappes. Ces solutions contribuent à l'élaboration de l'offre des serveurs de calcul développés au sein de *BULL*. Ces recherches sur les grappes et grilles sont au cœur des thématiques du laboratoire *LIG* (*ID-IMAG*), en particulier sur des aspects comme l'évaluation de performance et sur la gestion de ressources dynamiques.

1.2 Problématiques et objectifs

La définition de modèles prédictifs a plusieurs objectifs. D'une part, elle permet une meilleure compréhension des phénomènes de concurrence et, si on applique les modèles à une application scientifique, elle permet d'identifier les périodes de pertes de performances de l'application. D'autre part, la définition, pour chaque réseau, d'un modèle permet de facilement comparer les performances d'une application. Ces deux points peuvent être des éléments importants pour aider un constructeur à proposer une solution réseau pour un ensemble d'applications.

L'observation et la compréhension des mécanismes de communication permettent de relier les pertes de performance à certains détails des mécanismes réseaux. Cette analyse fine des comportements réseaux révèle, par exemple, le gain obtenu en utilisant un lien full-duplex, ou à l'inverse, le coût en temps que provoque la gestion de la concurrence par le contrôle de flux. Il devient donc possible d'associer à ces mécanismes l'influence qu'ils génèrent sur les performances d'applications parallèles.

1.3 Contributions

Les contributions de cette thèse s'articulent en fonction des deux objectifs énoncés dans la section précédente.

Expériences et analyse des communications concurrentes

Cette thèse met en évidence au travers d'expériences les effets provoqués par la congestion dans les réseaux haute performance. Les travaux proposés établissent une relation entre la perte globale de performance dûe à la concurrence et le contexte d'exécution des communications. Pour atteindre cet objectif, nous faisons varier un contexte expérimental basé sur une association entre un programme synthétique et un placement des tâches pour créer des contextes de concurrences particuliers. Ces expériences sont basées sur les grappes du projet Grid'5000[1] ainsi que sur la grappe *Teratec* fournie par la société *Bull*. Pour mettre en avant les phénomènes concurrents, le nombre d'expériences réalisées nécessaires est conséquent (quelques milliers), chaque expérience correspondant à plusieurs milliers de tests. En se basant sur ces expériences, nous avons défini les notions de conflits réseaux et de pénalités.

De manière plus pragmatique, les comportements concurrents sont étudiés sur trois architectures réseaux : *Quadrics*, *Myrinet* et *Gigabit Ethernet*. En outre, cette étude révèle et approfondit également l'influence spécifique de plusieurs composants réseaux mis en jeu lors d'une communication.

Modélisation des communications concurrentes

En se basant sur l'étude précédente, plusieurs modèles de communication concurrente ont été définis. Ces modèles décrivent le partage de la bande passante réseau entre communications concurrentes et les pertes de performance qu'occasionnent ce processus de partage. Ils sont basés soit sur une compréhension des mécanismes de contrôle de flux, soit sur une description des effets du partage de la bande passante. Ils modélisent deux architectures réseaux : *Myrinet* et *Gigabit Ethernet*. Ils sont intégrés à l'intérieur d'un simulateur qui calcule les effets de la concurrence sur des applications scientifiques. Les modèles couplés au simulateur prédisent avec une erreur faible (en moyenne 10%) les temps des communications par rapport à des valeurs mesurées d'exécutions réelles. Le simulateur développé contient environ 4200 lignes de code.

Cette thèse a fait l'objet de deux communications en conférences internationales [59][52], deux communications en conférences nationales [57][58], d'une présentation dans un atelier international [60] et d'un rapport de recherche [61].

1.4 Organisation du manuscrit

Ce manuscrit s'articule autour de trois parties chacune divisée en deux chapitres.

La première partie introduit le contexte relatif au calcul haute performance. Une description des principaux composants d'une grappe de calcul est présentée dans le premier chapitre. Cette description regroupe à la fois les éléments logiciels et matériels mis en jeu lors de l'exécution d'une application. Cette présentation met en valeur les influences de ces composants sur les performances des applications. Le deuxième chapitre propose une étude des travaux de recherche relatifs à la modélisation des communications réseaux. Ce chapitre permet de positionner notre

[1] `www.grid5000.org`

approche par rapport aux modèles existants et aux particularités des réseaux haute performance.

La deuxième partie traite de manière approfondie les communications concurrentes et leurs effets sur les performances. Le chapitre 4 étudie les effets des communications concurrentes. Ce chapitre expose le protocole expérimental choisi et propose une analyse des résultats obtenus par plusieurs expériences. Ces expériences décrivent l'influence de la concurrence pour les trois architectures réseaux étudiées. En se basant sur cette analyse, les modèles de concurrence sont définis dans le chapitre 5. Ce chapitre inclut également des éléments formels permettant la réalisation d'un simulateur. Ce simulateur est utilisé pour appliquer les modèles dans le cadre d'applications réelles.

La troisième et dernière partie s'articule également sur deux chapitres. Le chapitre 6 étudie l'impact de certains composants réseaux sur les effets engendrés par la concurrence. Il est constitué de plusieurs expérimentations dont les protocoles sont ajustés pour caractériser l'influence d'un composant en particulier. L'influence de ces composants sur les modèles est également abordée. Le dernier chapitre de ce document évalue la précision des modèles dans leurs prédictions. Cette précision est évaluée par comparaison entre des mesures observées et prédites grâce à l'utilisation de schémas de communications synthétiques et de schémas de communication provenant de l'application *HPL*. De plus, la recherche de modèles a mené à la découverte de certains facteurs permettant l'optimisation des coûts réseaux. Les effets directs de ces optimisations sur *HPL* sont également présentés dans ce chapitre.

I

Communications dans les grappes

Cette première partie présente le contexte technologique du calcul à haute performance. Cette présentation est effectuée du point de vue des performances. Les performances d'un système représentent sa capacité à fournir des ressources aux utilisateurs et applications. Dans le cas d'une grappe de calcul, les acteurs sont les applications et les critères de performance sont, par exemple, le temps d'exécution ou le taux d'utilisation d'une ressource : mémoire, processeur ou réseau. Cependant, l'influence de l'architecture matérielle et de la pile logicielle (système, support d'exécution) est prépondérante pour établir les différents critères de performance. Certains critères théoriques, par exemple ceux proposés par les constructeurs, ne sont que très rarement atteints et, par conséquent, ne constituent pas des références reconnues pour prédire les performances des applications. Ces différents problèmes sont abordés dans le premier chapitre qui montre sur des exemples l'impact des principaux choix de conception des composants d'une grappe. Il est également tenu compte de l'influence de différentes techniques logicielles sur les performances des applications. Le second chapitre est consacré à l'étude de modèles prédisant les temps de communications réseaux. Un problème important dans l'étude des communications réseaux sont les phénomènes de congestion. Nous verrons comment certains modèles tentent de comprendre et prédire l'impact de ces phénomènes.

Grappes de calcul 2

2.1 Introduction

Les nouvelles architectures informatiques offrent aux chercheurs et ingénieurs la possibilité d'analyser, de simuler et de prédire le comportement de systèmes complexes réels de grande taille. De tels systèmes, comme par exemple celui de la propagation des ondes sismiques sur la surface du globe ou des interactions entre particules à hautes énergies, sont simulés en utilisant des modèles sophistiqués tentant au maximum de reproduire la réalité du phénomène physique. Ces simulations exigent de très grandes ressources de calcul et d'espace mémoire. Cette demande en ressource informatique, sans cesse croissante, ne peut être satisfaite par les architectures séquentielles traditionnelles. C'est ce qui a principalement motivé les recherches et développement autour du calcul à haute performance et l'émergence des architectures parallèles et même massivement parallèles. Malheureusement, les coûts de recherche et de développement étaient très importants par rapport à la taille du marché, ce qui a entraîné la disparition de nombreuses sociétés spécialisées dans les architectures parallèles. Les grappes de calcul, avec un rapport très agressif coût/performance, se sont rapidement positionnées comme des concurrentes directes des architectures parallèles.

Les grappes de calcul (ou *cluster*) sont constituées d'un ensemble homogène de machines (appelées nœuds) connectées par un réseau dédié. Les premières grappes sont apparues dans le milieu des années 90 et étaient basées sur des nœuds monoprocesseur possédant à peu près les mêmes caractéristiques que les machines de bureau ou stations de travail (*cluster of workstation*). Le projet *BEOWULF*[1] [90] a été un pionnier pour la mise en œuvre de grappes de calcul construites avec des composants du marché. La grappe *Icluster1* a démontré, en 2001, qu'il était possible de recycler 225 PC de bureau (HP Vectra) en nœuds d'une grappe de calcul qui s'est retrouvée classée à la 375ème place d'un classement du TOP500. Plus récemment, le projet RNTL IGGI[2] propose d'agréger temporairement les ressources (PC de bureau) non utilisées (nuit ou week-end) d'un intranet pour en faire des nœuds d'une grappe de calcul virtuelle.

De nos jours, les évolutions technologiques et la diminution des coûts de production des processeurs ont permis l'émergence des machines multiprocesseurs. Les nœuds des grappes de calcul sont ainsi rapidement devenus multiprocesseurs et même multicœurs. A l'intérieur des nœuds de telles architectures, l'un des problèmes cruciaux est celui du partage de ressources (mémoire, réseau, stockage) par les différents processeurs.

[1] Projet Beowulf, `http://www.beowulf.org`
[2] Projet RNTL IGGI `http://iggi.imag.fr/`

La puissance de calcul des grappes ne suffisant pas, une étape a été franchie avec l'apparition des grilles de calcul qu'on peut rapidement définir comme l'agrégation des grappes de calcul. Nous citerons bien évidemment la grille de calcul française Grid'5000[3] et les grappes de calcul installées dans 9 villes (Orsay, Lille, Nancy, Lyon, Bordeaux, Grenoble, Nice, Toulouse, Rennes). Certaines de ces grappes ont été utilisées dans les développements présentés dans ce chapitre. Il convient de rappeler que la problématique de recherche se situe au niveau de l'étude du comportement d'une grappe de calcul et non pas du comportement au niveau global d'une grille.

Les performances d'une application parallèle s'exécutant sur les ressources fournies par une grappe, dépendent, à la fois, de la complexité algorithmique de l'application, du langage de programmation (Fortran, C, Java), du processus de compilation ou d'interprétation, du système d'exploitation et bien évidemment des composants matériels de la grappe. Ainsi, du point de vue des performances, différentes approches et technologies permettent aux applications d'optimiser certains critères de performances, tels que le temps d'exécution, les taux d'utilisation des processeurs ou de la mémoire, ou le coût des communications. Le TOP'500[4] classe les 500 architectures les plus puissantes au monde à partir d'une application de calcul numérique (*HPL*). Cette application est jusqu'à aujourd'hui considérée comme l'application de référence pour quantifier la puissance d'une machine. Dans la suite de ce chapitre, plusieurs références à ce classement seront utilisées.

Notre analyse des performances d'une grappe débutera avec l'étude des différents composants des nœuds d'une grappe. Nous aborderons ainsi les processeurs, la hiérarchie mémoire des nœuds et les réseaux de communication. Nous nous concentrerons sur l'architecture générale de ces composants et donnerons quelques-uns des éléments caractérisant leurs performances.

2.2 Nœuds de calcul multiprocesseurs

La technologie des nœuds est un élément clé dans la puissance globale d'une grappe. Plusieurs paramètres, comme le nombre de processeurs, la structuration de l'espace mémoire (cache et centrale), l'interconnexion avec les éléments périphériques, auront un impact déterminant sur les performances globales d'une grappe de calcul. C'est ce que nous allons voir dans les sections suivantes.

2.2.1 Exploitation du processeur

Le processeur d'un nœud est l'un des composants essentiels fournissant la puissance nécessaire aux applications. Le processeur, composé d'une ou plusieurs unités fonctionnelles, exécute la suite d'instructions issues des applications et générées par le compilateur.

La complexité de la structure des processeurs ne cesse de croître, mais leur programmation par l'utilisateur reste simple et souvent efficace grâce au travail effectué par le compilateur. Dans cette partie, nous choisissons de présenter l'ensemble des caractéristiques conditionnant les performances du processeur. En suivant cette logique, nous présenterons deux technologies actuelles de processeur.

[3] `http://wwww.grid5000.org`

[4] Les références au classement TOP500 correspondent au classement de juillet 2006, `http://www.top500.org`

2.2.1.1 Structure du processeur

La fréquence d'un processeur se mesure en Hertz (Hz) et indique le nombre de cycles par seconde. Cette fréquence constitue un premier indicateur pour classer les processeurs d'une même famille (deux processeurs Itanium par exemple). La fréquence ne permet pas de comparer des processeurs de familles différentes (par exemple un Itanium et un Opteron). Aujourd'hui, on constate que les fréquences des processeurs ont tendance à stagner. Les efforts des constructeurs se portent plus sur le développement des technologies de type multicœurs (plusieurs unités de traitement sur un même chip).

Le format du mot machine et de l'adressage est un autre paramètre déterminant dans le processeur. La tendance aujourd'hui est d'aller vers des processeurs manipulant des données sur 64 bits. Cela permet avec une seule instruction d'exécuter des opérations sur des entiers et flottants de plus grande taille, ce qui peut être très intéressant pour des applications de calcul où le besoin de précision est crucial. Le format des adresses tend aussi à aller vers du 64 bits. Cela ouvre la possibilité d'un espace d'adressage extrêmement grand et allant largement au-delà des capacités des mémoires actuelles.

Le modèle von Neumann[38] a pour conséquence une exécution séquentielle des instructions d'un programme. Pour améliorer les performances des processeurs, les recherches des constructeurs ont abouti à des solutions permettant à plusieurs instructions de s'exécuter simultanément. Cela a commencé avec l'exécution pipelinée des instructions qui, à partir de l'exécution des instructions en plusieurs phases, a permis d'alimenter un pipeline avec plusieurs instructions. Les technologies de type superscalaire permettent aussi d'exécuter des instructions arithmétiques sur différentes unités fonctionnelles. Les technologies de type ILP (Instruction Level Parallelism) [11] ou SMT (Simultaneous MultiThreading) [24] permettent de s'affranchir de la contrainte limitant le processeur a exécuté qu'une seule instruction à un instant donné.

2.2.1.2 Mémoire cache

En plus du petit nombre de registres contenant des mots machine, le processeur dispose de plusieurs niveaux de mémoire cache. La capacité de ces mémoires caches est supérieure à l'espace mémoire des registres, mais reste évidemment largement inférieure à la capacité de la mémoire centrale. Les temps d'accès à ces niveaux de caches sont très inférieurs aux temps d'accès à la mémoire centrale. Le principe de base dans l'utilisation de la mémoire cache est de stocker les données dont on estime probable les accès futurs. La structure de la mémoire cache est décrite de façon très détaillée dans [42]. Elle se divise en plusieurs sections de taille différente : le cache L1, le plus petit, est utilisé pour contenir une partition des données et des instructions du programme, le cache L2 et parfois L3 sont généralement uniquement utilisés pour les données du programme. Chaque accès à un niveau de cache implique un temps d'accès mesuré en cycles processeurs, plus ou moins grand suivant la proximité et la taille du cache. Dans la nouvelle génération de processeurs multicœurs contenant plusieurs processeurs (cœurs) par chip, la mémoire cache est souvent séparée entre chaque cœur.

Compilateur	ICC version 7.1	ICC version 8
Performance SPEC2000	92	109

TAB. 2.1 – Influence du compilateur, SPEC2000, Bull Novascale 8 processeurs Itanium 2

2.2.1.3 Compteurs de performances

La plupart des processeurs disposent de compteurs matériels qui comptabilisent les événements liés au fonctionnement du processeur. Nous citerons, à titre d'exemple, le nombre de cycle processeurs (*CPU_CYCLES*) ou le nombre d'accès aux différents niveaux de caches (*L1D_READ_MISS*, *L2_HITS*, etc). Les valeurs de ces compteurs donnent des informations très précises, mais de bas niveau, sur les performances d'une application ou d'un programme. Ces compteurs peuvent expliquer certaines pertes de performance des programmes. Par exemple, un programme pourrait avoir de très mauvaises performances parce que les données accédées ne se retrouveraient quasiment jamais dans les caches. Les compteurs *L2_MISS* ou *L3_MISS* auraient ainsi des valeurs très grandes. Ces compteurs permettent aussi de comptabiliser le nombre de tops d'horloge du processeur et ainsi d'avoir des outils de mesures avec une unité en cycles d'horloge.

2.2.1.4 Compilateurs

Finalement, le potentiel d'un processeur à fournir des ressources de calcul dépend des programmes qui lui sont soumis. Nous ne parlerons pas ici de l'algorithmique des programmes, mais plutôt du processus de traduction et de génération de code qui est assuré par le compilateur. Cette traduction peut être générique à une structure commune de processeurs (*i386*, *x86_64*) ou spécifique à un processeur particulier (*alpha*). Il en résulte que les performances d'une application sur un nœud de calcul reposent aussi sur l'intelligence du compilateur, à savoir, l'utilisation des caractéristiques fines du processeur. Dans un futur proche, il est raisonnable d'envisager que l'extension des compilateurs englobera aussi la compréhension et l'optimisation des applications pour une architecture mémoire donnée.

Pour illustrer ces propos, nous proposons dans le tableau 2.1 de comparer deux versions d'une même application compilée avec un compilateur de génération différente et exécutée sur une même architecture. L'application est le *benchmark SPEC2000*[5] et l'architecture est une machine *Bull Novascale* composée de 8 processeurs Itanium 2 (1.5GHz, 6Mo en cache L3). Tous deux sont décrits plus amplement dans la sous-section suivante 2.2.2. Le compilateur utilisé est le compilateur *Intel ICC*[6]. Les valeurs proposées sont les moyennes géométriques de 12 tests réalisant des calculs entiers intensifs. Cette moyenne est à comparer aux résultats sur une machine particulière obtenant un indice de 100. Ainsi, plus la valeur obtenue par le *benchmark SPEC2000* est grande plus les performances sont bonnes. Dans ce cas précis, une évolution du compilateur permet un gain de performances de presque 20% ce qui démontre le rôle prépondérant du compilateur.

Comme nous l'avons vu dans les paragraphes précédents, les processeurs 64 bits sont les plus fréquemment utilisés dans le contexte du calcul haute performance où les exigences en calcul flot-

[5] *SPEC2000*, `http://www.spec.org`

[6] *Intel Compilers*, `http://www.intel.com/cd/software/products/asmo-na/eng/compilers/`

tant sont grandes. Lors des différentes expérimentations présentées dans la suite de ce document, nous avons utilisé deux technologies de processeur 64 bits. Dans les deux sous-sections suivantes, nous détaillons ces deux technologies.

2.2.1.5 Processeur AMD Opteron

Advanced Micro Devices (*AMD*) est un constructeur américain de processeurs. Cette société s'est positionnée sur le même marché que celui de la société *Intel* leader du secteur. *AMD* a choisi une solution basée sur une évolution de l'architecture des processeurs 32 bits vers une architecture 64 bits. Le processeur Opteron est un processeur *RISC* 64 bits compétitif offrant un rapport coût/performance de tout premier plan. C'est une des raisons qui explique pourquoi on le retrouve dans les nœuds de nombreuses grappes de calcul.

Structure du processeur Opteron

Les figures 2.1 et 2.2 présentent les structures des processeurs Opteron monocœur et double cœur où nous remarquons que chacun des cœurs dispose de deux niveaux de cache indépendants. D'autre part, il est intéressant de noter que les données et instructions arrivent, soit par les liens *Direct Connect*, soit par les liens *Hyper Transport* et transitent à travers un crossbar[7] puis sont redirigées via le bus de requêtes vers le cœur concerné. Nous pouvons aussi noter que le processeur double cœur n'apporte aucune modification significative au sein du processeur. Ainsi, la queue des requêtes, le crossbar et les contrôleurs mémoires sont partagés entre les deux cœurs. La fréquence d'horloge de l'Opteron varie actuellement entre 1.8 et 2.2 GHz.

L'Opteron est constitué d'une technologie *Direct Connect* reliant directement au processeur la mémoire et les entrées/sorties (E/S). Il est équipé d'un contrôleur mémoire *DDR* qui gère les accès au bus mémoire maximisant sa bande passante en fonction du nombre de processeurs.

L'*Hyper Transport* est le nom donné par *AMD* au réseau reliant les processeurs entre eux ainsi que les autres microcontrôleurs de la carte mère. Chaque processeur Opteron comporte trois liens *Hyper Transport*.

Mémoire cache

Un des points faibles des processeurs *AMD* est la taille réduite de leur cache. Ils ne comportent que deux niveaux de cache dont la taille maximale est 2 Mo. Cependant, l'ajout d'un contrôleur mémoire *DDR* directement dans la structure du processeur améliore grandement les temps d'accès en mémoire centrale et efface quelque peu les effets d'un cache de taille réduite.

Compteur de performances

Le processeur Opteron dispose des compteurs de performances tels que le compteur de cycles ou le nombre de défauts de cache. Il n'offre pas de compteurs de performances spécifiques ni de technologies innovantes.

[7] Le terme anglais *crossbar* sera utilisé pour référencer un commutateur utilisant une topologie en croisillon.

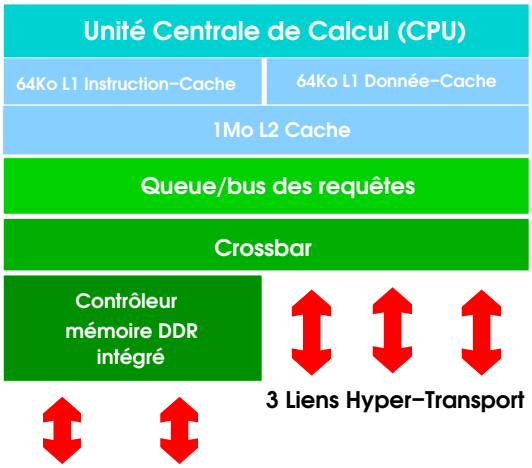


FIG. 2.1 – Processeur Opteron monocœur

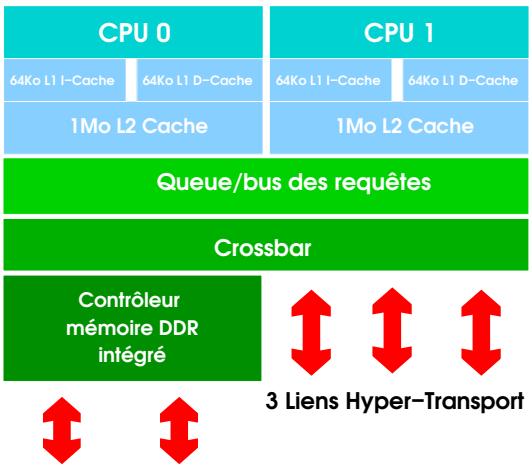


FIG. 2.2 – Processeur Opteron double cœur

Compilateur

La société *AMD* ne développe pas de compilateur spécifique pour ses processeurs. Le compilateur généralement utilisé est le compilateur *GNUcc*[8]. Nous pouvons, aussi citer d'autres compilateurs tels que les compilateurs *PGI*[9] qui proposent des optimisations pour le processeur Opteron. Ces compilateurs fournissent également le support *OpenMP* ou bien encore le compilateur *Fortran 90* non disponibles avec l'environnement *GNU*.

2.2.1.6 Processeur Intel Itanium 2

Le processeur Itanium provient d'un partenariat entre le constructeur *Hewlett-Packard-Compaq* (*HP-Compaq*) et *Intel*. L'objectif de ce consortium est de concevoir et réaliser un nouveau processeur 64 bits. Ce choix de conception d'un nouveau processeur est tout à fait en opposition à celui effectué par la société *AMD* pour son processeur Opteron 64 bits qui s'appuie sur l'architecture 32 bits.

Structure de l'unité de calcul

La structure de l'Itanium 2 est basée sur la technologie *Explicitily Parallel Instruction Computing* (*EPIC*) qui incorpore un fort taux de parallélisme dans l'exécution des instructions.

Les figures 2.3 et 2.4 présentent la structure d'un processeur Itanium 2 monocœur et double cœur. A la différence du processeur Opteron double cœur, l'Itanium 2 double cœur incorpore une hiérarchie avec deux nouveaux niveaux (Synchroniseur et Arbitre) par rapport à la version monocœur. Cette hiérarchie à deux niveaux permet de mieux équilibrer la charge entre les deux cœurs. La fréquence d'horloge de l'Itanium 2 varie entre 0.9 et 1.66 GHz.

[8] *GNU Compiler Collection*, `http://www.gnu.org/software/gcc/`
[9] *PGI High-Performance Compiler*, `http://www.pgroup.com/`

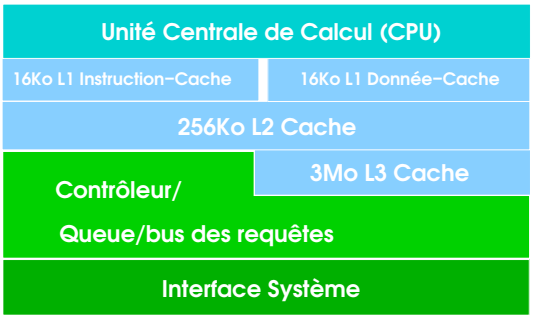


FIG. 2.3 – Processeur Itanium monocœur

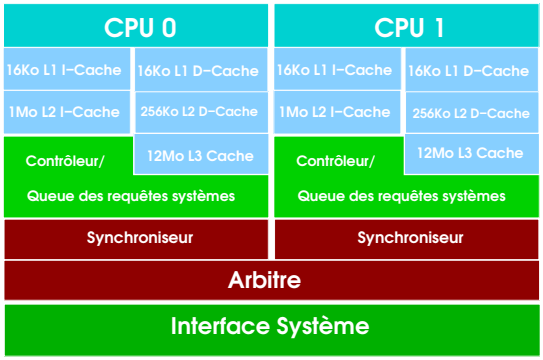


FIG. 2.4 – Processeur Itanium double cœur

Mémoire cache

Un autre choix technologique de l'Itanium 2 est de s'appuyer sur une mémoire cache de grande taille. L'effet bénéfique des caches de grande taille n'est pas direct, car le temps de latence des caches augmente avec leur taille. La version double cœur incorpore un cache L3 de 12 Mo par cœurs, soit au total 24 Mo de cache par processeur.

Compteur de performances

Une particularité du processeur Itanium 2 est de pouvoir agréger plusieurs compteurs. De plus, ces compteurs sont structurés de façon à permettre un accès facile et uniforme. En particulier, il est possible d'extraire les valeurs des compteurs pour une région d'un code applicatif.

Compilateur

Intel développe son propre compilateur pour ses différentes familles de processeur. Le compilateur *ICC* comporte une version pour l'Itanium 2. Le développement du compilateur *ICC* est très actif pour faire face aux demandes des utilisateurs. D'une version à l'autre du compilateur *ICC*, les gains sur des codes peuvent être assez importants.

Conclusion

La société *Intel* propose une mesure permettant d'évaluer la puissance théorique d'un processeur. Cette mesure, *Composite Theoretical Performance* (*CTP*), se calcule à partir d'une formule prenant en compte le nombre de cœurs du processeur, sa fréquence ainsi que la longueur du mot machine. Cette mesure ne prend pas en compte la taille des caches ni le débit du bus reliant le processeur aux autres composants. Son unité est le million d'opérations théoriques par secondes (*Mtops*). Le tableau 2.2 présente les valeurs *CTP* des processeurs Opteron et Itanium monocœur et double cœur.

De ce point de vue, l'Itanium 2 est plus puissant que l'Opteron pour les calculs à virgule

Itanium 2		Opteron	
monocœur 1.4 Ghz	double cœur 1.4 Ghz	monocœur 1.8 Ghz	double cœur 1.8 Ghz
9100	17500	5550	10500

TAB. 2.2 – *CTP* des processeurs Opteron et Itanium

	Itanium 2 (ICC 9.0)			Opteron (GCC 4.1)		
	0.9 Ghz L3 : 3Mo	1.5 Ghz L3 : 6Mo	1.6 Ghz L3 : 9Mo	1.0 Ghz L2 : 1Mo	1.8 Ghz L2 : 1Mo	2.2 Ghz L2 : 1Mo
FFT (s)	15.9	7.9	9.0	16.7	9.7	8.2
LU (s)	216.5	130	122.3	180	112	99

TAB. 2.3 – Comparaison des temps d'exécution de LU et FFT sur un processeur Itanium 2 et un processeur Opteron.

flottante. Cette constatation correspond bien à la particularité de l'Itanium 2 d'être principalement dédié aux calculs en virgule flottante. Le coût d'ajout d'un cœur ne double pas exactement la valeur *CTP*. Il en résulte que d'un point de vue théorique, l'addition d'un cœur ajoute un surcoût pénalisant les performances. Nous pouvons aisément supposer que ce même phénomène, quantifié théoriquement, affecte effectivement les performances réelles.

Du point de vue des performances observées, le tableau 2.3 reprend les temps d'exécution pour deux applications avec différentes générations de processeur Itanium 2 et Opteron. Les applications considérées font partie des applications fréquemment utilisées dans le calcul scientifique. Il s'agit d'une factorisation de matrice (LU) et du calcul d'une transformée de Fourier (FFT). Chacune de ces applications, appartenant à l'ensemble applicatif *Splash 2*[10][81], utilise une importante part des ressources calcul et de mémoire d'une machine. L'application FFT demande plus de ressource de calcul alors que l'application LU est plus sensible à la ressource mémoire. Le tableau 2.3 montre qu'au niveau de l'application FFT, et donc, au niveau de la capacité de calcul, les processeurs Opteron et Itanium 2 proposent des résultats assez similaires. Cependant, notons que la fréquence des processeurs Itanium 2 est plus faible que celle des processeurs Opteron indiquant, qu'à fréquence égale, les processeurs Itanium 2 sont probablement plus performants. Nous pouvons observer cet effet en comparant le résultat de FFT sur l'Itanium 2 à 0.9 Ghz et l'Opteron à 1.0 Ghz. Au niveau de l'application LU sensible à la ressource mémoire, nous remarquons que les processeurs Opteron sont nettement plus performants. Ainsi, la grande taille de cache des processeurs Itanium 2 ne favorise pas l'application LU. Une explication possible de la différence des temps entre Itanium 2 et Opteron tiendrait en ce que l'Itanium 2 propose des performances d'accès à la mémoire centrale plus longs que celles de l'Opteron. La comparaison proposée reste limitée et sujette aux performances des compilateurs respectifs.

D'autres éléments de comparaison sont les temps de latence d'accès aux différents niveaux de cache pour chacun des types de processeur. Le tableau 2.4 résume le temps de latence pour chaque niveau de cache ainsi que la taille des lignes de cache. Ce tableau montre que le processeur Itanium 2 utilise moins de cycle pour accéder aux données dans le cache. Néanmoins, la valeur en temps d'un cycle est différente entre les processeurs Itanium 2 et Opteron, il dépend de la fréquence du

[10] *Splash 2*, `http://www-flash.stanford.edu/apps/SPLASH/`

Processeur	L1	L2	L3
Itanium 2	1c / 64o	5-6c / 128o	12c / 128o
Opteron	3c / 64o	8-11c / 64o	-

TAB. 2.4 – Comparaison des temps d'accès aux caches (nombre de cycles CPU) et taille des lignes de caches.

processeur qui est souvent plus élevée pour les Opterons.

En se référençant au TOP500, l'Opteron détient 16% du nombre total de processeurs des 500 architectures les plus performantes au monde. Parallèlement, l'Itanium 2 représente 7,4% en terme de processeurs au TOP500 avec, entre autre, les positions 4 et 5 du classement sur deux architectures différentes de nœuds.

Il en résulte que techniquement l'Itanium 2 par sa technologie *EPIC* et les capacités de ses caches apporte des axes innovateurs. Mais, ceux-ci ne confèrent pas à l'Itanium 2, une suprématie dans la famille des processeurs 64 bits. Effectivement, d'une part la maturité de ces nouvelles technologies n'a pas encore été atteinte par *Intel*, et d'autre part, l'impact bénéfique sur les performances n'a pas été prouvé. A l'inverse, l'Opteron se base sur une technologie mature, mais, mise à part la technologie *Hyper Transport*, il ne propose aucun axe de développement pour les processeurs de génération future. Un autre aspect important est celui de la consommation énergétique. Le processeur Itanium 2 se révèle beaucoup plus gourmand en électricité avec pour conséquence des grands besoins en capacité de refroidissement.

La performance d'un nœud de calcul dépend aussi, pour une grande part, de l'architecture mémoire et plus particulièrement des bus et de leur bande passante pour l'accès aux données.

2.2.2 Architecture mémoire

L'augmentation du nombre de processeurs et de cœurs dans les architectures multiprocesseurs accroît le nombre d'accès simultanés à la mémoire centrale. L'ajout de cache sur chaque processeur réduit évidemment le nombre d'accès mémoire, mais la cohérence des données entre ces caches pose de nouvelles problématiques. Dans les nœuds de calcul, il convient de considérer la mémoire comme la seconde ressource à prendre en compte.

Le passage d'une architecture monoprocesseur à une architecture multiprocesseur peut se faire tout simplement en connectant de nouveaux processeurs sur le bus donnant accès à la mémoire. C'est le principe de base des architectures *UMA*.

UMA

L'architecture à accès uniforme en mémoire *Uniform Memory Acces* (*UMA*) est une solution simple et économique proposée par les constructeurs d'ordinateurs. Avec cette architecture, chaque processeur accède à la mémoire centrale de façon uniforme c'est-à-dire en partageant le même bus d'accès.

La problématique de cohérence entre les caches des processeurs se pose avec les architectures multiprocesseur. En effet, en supposant que deux processeurs détiennent un même bloc de données dans leurs caches, si le premier processeur effectue une modification sur ce bloc avant que le

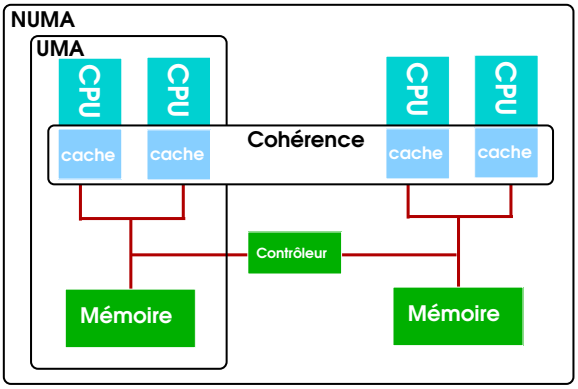


FIG. 2.5 – Architecture *UMA*, *NUMA* et cohérence de cache.

second n'en n'effectue une lecture, le bloc de données dans le cache du second processeur doit être invalidé. Cette invalidation provoque l'accès, par le second processeur, à la dernière mise à jour du bloc de données se trouvant dans le cache du premier processeur ou en mémoire centrale.

Dans [23] les auteurs présentent un résumé des différentes techniques logicielles et matérielles de cohérence de caches. Les performances des techniques de cohérence ont été discutées dans [86] et [83]. Pour une architecture *UMA*, la technique d'invalidation de lignes de cache la plus utilisée repose sur une scrutation du bus mémoire : *snoofing* et *snarfing*. Dans la technique de scrutation de bus, lors de la mise à jour d'une ligne de cache (écriture), le contrôleur de cache diffuse soit uniquement l'adresse (*snoofing*) soit l'adresse et la donnée (*snarfing*) de la ligne de cache modifiée aux autres contrôleurs de cache qui, le cas échéant, invalident leurs données dans le cache. Le maintien de la cohérence entre les caches provoque un trafic important sur le bus mémoire dégradant d'autant plus les performances des nœuds de calcul et les accès mémoire.

Ainsi dans une architecture *UMA*, le bus mémoire est partagé entre chaque processeur ce qui génère des goulots d'étranglements et de forts ralentissements des applications lorsqu'on augmente le nombre de processeurs. Pour éviter cette perte de performance, il existe des solutions augmentant la capacité du bus ou bien grossissant la taille des caches des processeurs. Ces techniques restent minimales et finalement peu évolutives. Néanmoins, l'uniformité des accès présente l'avantage de rendre plus simple l'évaluation des performances et la réalisation de modèles prédictifs. Pour palier au partage exclusif du bus mémoire par un grand nombre de processeurs, des architectures mémoire hiérarchiques et distribuées ont été proposées.

NUMA

L'objectif des architectures NUMA est de construire des machines à mémoire partagée autour d'un grand nombre de processeurs. Il fallait résoudre le problème de contention sur le bus mémoire qu'on pouvait observer sur les architectures UMA. Le principe de base est de former de petits groupes de processeurs disposant d'un bus et d'un espace mémoire. Les groupes de processeurs sont ensuite interconnectés par des réseaux d'interconnexion plus ou moins sophistiqués (hypercube pour *Origin 2000* de *SGI*, grille torique pour *HP/Marvel*) pour des technologies en anneau ou par commutateurs. Ces architectures sont appelées architectures aux accès mémoire

non uniformes *Non-Uniform Memory Acces* (*NUMA*), car le temps d'accès pour un processeur peut varier en fonction de la localisation des données accédées. L'introduction du facteur *NUMA* permet d'identifier le nombre de latences mémoire différentes au sein de la machine.

La cohérence de cache intervient également dans les architectures *NUMA*. Cependant, la technologie utilisée est différente de celle des architectures *UMA*. Cette nouvelle technologie consiste à la création d'un répertoire de cache. Ce répertoire correspond à un annuaire des pages qui ont été mises en cache par chaque processeur. Si un processeur effectue une lecture sur une donnée qu'il détient dans son cache, il va interroger le répertoire pour connaître l'emplacement de la dernière mise à jour de cette donnée. Cette donnée peut être soit contenue en mémoire centrale soit détenue par le cache d'un autre processeur. Cette technique logicielle d'invalidation de lignes de cache est appelée, technique par répertoire ou *directory*.

Finalement, la figure 2.5 synthétise les différentes structures mémoire de machine multiprocesseur.

Il n'existe pas de standard d'architecture *NUMA*, chaque fabriquant propose sa propre topologie. Les figures 2.6, 2.7 et 2.8 décrivent les architectures mémoires développées par différents constructeurs.

Architecture SGI

L'architecture *SGI Altix 3700*[11], figure 2.6, est basée sur des blocs de 2 processeurs. L'architecture *SGI* peut contenir au maximum 512 processeurs pour un facteur *NUMA* maximal de 4. Les bandes passantes théoriques par processeur sont identiques et valent 3.2 Go/s.

Architecture Bull

L'architecture *Bull Novascale* comporte des blocs à 4 processeurs et un facteur *NUMA* de 3 pour un maximum de 32 processeurs. Une telle architecture présente l'inconvénient d'avoir une bande passante mémoire commune à 4 processeurs, la réduisant à 1.6 Go/s par processeur.

Architecture *AMD* Opteron

Le processeur Opteron comporte trois liens *Hyper Transport*. Les liens *Hyper Transport* sont les liens reliant les processeurs entre eux. *AMD* propose des architectures *NUMA* utilisant les liens *Hyper Transport*, figure 2.8. Chaque processeur est relié à une mémoire différente par un lien *Direct Connect* de 6.4 Go/s. Le facteur *NUMA* dépend directement du nombre de processeurs.

Effet *NUMA*

De telles architectures ouvrent de nouvelles problématiques. Ces nouvelles problématiques résultent tant de l'utilisation de ces technologies par des applications que de l'évaluation du gain de performance obtenu. Ces architectures sont aussi à l'origine du développement de bibliothèques spécialisées dans l'allocation et le contrôle de zones mémoire. Pour montrer l'influence sur les

[11] Dans la suite du document ces architectures seront référencées par le nom de leur constructeur.

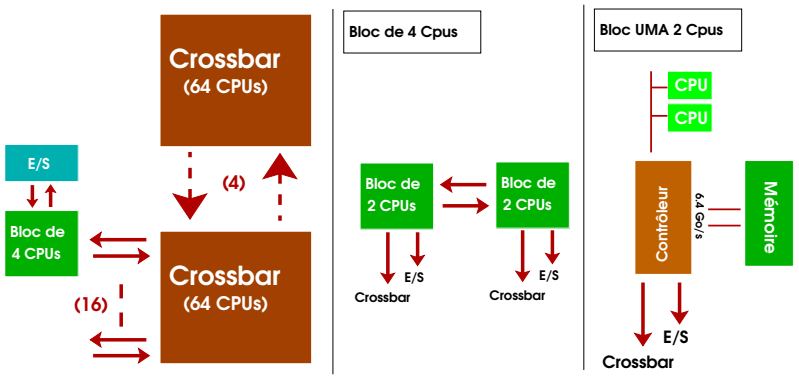


FIG. 2.6 – Architecture NUMA : *SGI Altix 3700*, 2-512 CPUs

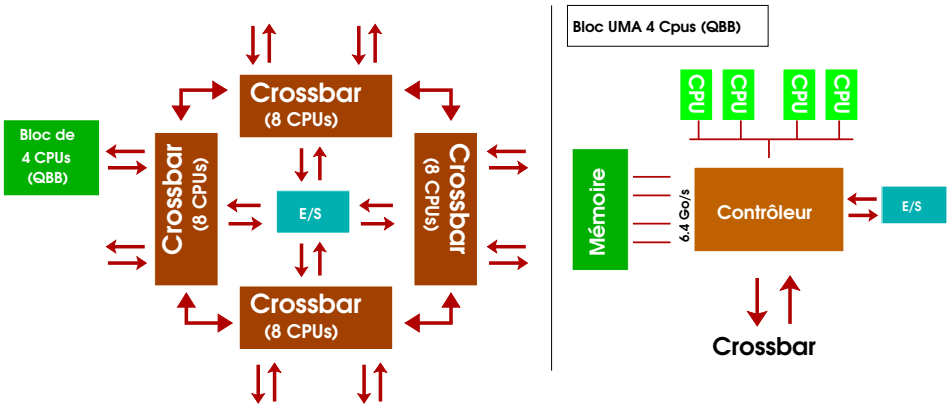


FIG. 2.7 – Architecture NUMA : *Bull Novascale*, 2-32 CPUs

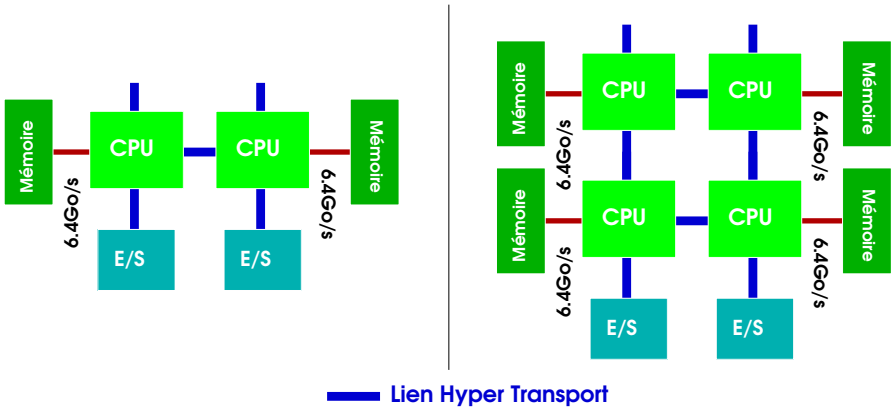


FIG. 2.8 – Architecture NUMA : *AMD Opteron*, 2 et 4 CPUs

Nombre de CPUs	CINT2000		CFP2000	
	Bull	SGI	Bull	SGI
4	55.7	57.2	64.1	82.2
8	108	114	151	164
16	198	227	213	327
32	387	447	447[13]	644
64	-	854	-	1257
128	-	1721	-	2410[13]

TAB. 2.5 – Comparaison de l'effet *NUMA*

performances que proposent les architectures NUMA, le tableau 2.5 présente les valeurs obtenues par le *benchmark* SPEC2000 sur les architectures *Bull* et *SGI*. SPEC2000 est un *benchmark* inspiré d'applications réelles faisant intervenir à la fois l'efficacité des processeurs et la structure mémoire. Il propose deux mesures, l'une dédiée au calcul entier intensif (CINT2000) et l'autre dédiée au calcul à virgule flottante intensif (CFP2000). Pour ces mesures[12], ces deux architectures disposent de la même version de processeur Itanium 2 (1.5 GHz, 6Mo de cache) et le *benchmark* est compilé avec le même compilateur *ICC* (génération 8.0 pour CINT2000 et génération 7.1 pour CFP2000). De plus, la capacité des bus mémoires est identique. Les valeurs présentées montrent de meilleures performances pour l'architecture *SGI*. Cela peut facilement être expliqué par la réduction par 4 pour l'architecture *BULL* contre 2 pour l'architecture *SGI* de la bande passante du bus mémoire. De plus, lorsque le nombre de processeurs est doublé, l'architecture *SGI* double ses performances. Parallèlement, l'architecture *BULL* est plus irrégulière (notons par exemple, le passage de 4 à 8 processeurs pour CFP2000, où l'architecture *BULL* multiplie par 2.35 ses performances, alors que pour la même mesure, le passage de 8 à 16 multiplie par 1.41 ses performances).

Le bus mémoire est aussi utilisé lorsque les processeurs ont à dialoguer avec les périphériques, par exemple pour les transferts sur le réseau ou bien vers les opérations disques. Une requête du processeur vers un périphérique (ou vice-versa) transite par le bus mémoire, mais aussi à travers un bus qui connecte les dispositifs périphériques.

2.2.3 Interactions avec les périphériques

Finalement, le dernier composant d'un nœud de calcul, qui influence les performances, est le bus connectant les processeurs aux cartes d'extension telles que les cartes réseaux. Le bus *PCI* est un bus unidirectionnel à 32 bits. La génération suivante des bus *PCI*, dénommée *PCI-X*, améliore la capacité du bus en permettant, par exemple, des échanges sur 64 bits. Cette génération de bus *PCI-X* propose une fréquence allant de 66 Mhz à 533 Mhz et une bande passante allant de 528 Mo/s à 4,2 Go/s.

Dernièrement, le développement du bus *PCI-Express* ou *PCIe* étend les capacités en terme de rapidité du traditionnel bus *PCI*. En effet, le bus *PCIe* n'est plus basé sur un simple bus, mais sur une architecture en étoile autour d'un commutateur. Le bus *PCIe* se diversifie en fonction de la taille des connecteurs. Une carte *PCIe* $1\times$ (c'est-à-dire carte à une voie, soit une liaison en

[12] Ces résultats sont consultables sur le site `http://www.spec.org`

[12] *Compilée avec ICC 8.0*

réception et une liaison en émission) permet d'atteindre un débit unidirectionnel de 250Mo/s, alors qu'une carte 32 voies (*PCIe* 32×) obtient une bande passante de 8 Go/s.

Dans le cadre des grappes de calcul, le bus dédié aux périphériques est l'élément qui connecte le réseau aux processeurs et à la mémoire. Il est alors envisageable de considérer la connectivité sur les périphériques comme un des composants traversés par une communication entre deux nœuds d'une grappe de calcul.

2.3 Réseaux haute performance

Les réseaux haute performance (ou haut débit) proposent des technologies efficaces de communication pour le transfert des messages. Dans un contexte de grappe, ces réseaux permettent de réduire les temps des phases de communication, et par conséquent, de limiter l'attente des processeurs. Les caractéristiques d'un réseau sont très fréquemment évaluées par deux paramètres : la latence et la bande passante. La latence correspond au temps nécessaire à l'envoi d'un message de taille minimale (un ou quelques octets). La bande passante correspond, pour des messages de grande taille à l'inverse du taux de transfert des données sur le réseau. Plusieurs techniques, à la fois logicielles et matérielles, permettent de réduire la latence et d'augmenter la bande passante[9].

2.3.1 Aspects technologiques

2.3.1.1 Technologies matérielles

Le réseau comprend un ensemble de composants que nous allons décrire dans les paragraphes suivants. Nous commencerons par décrire les fonctions réalisées par les cartes d'interface réseau (*Network Interface Controller, NIC*) qui connectent les nœuds de la grappe au commutateur que nous détaillerons ensuite. Dans le contexte des réseaux à hautes performances, nous noterons que ces réseaux sont d'une couverture géographique restreinte (une salle ou un bâtiment) et que les erreurs de transmission sont rares, ce qui permet d'alléger considérablement les procédures de contrôle et donc d'améliorer l'efficacité des communications.

Carte réseau

Les cartes réseaux sont des composants intra-nœud réalisant l'émission et la réception des messages provenant du réseau. Une des caractéristiques que l'on retrouve sur de nombreuses cartes est la présence d'un processeur spécifique dont le rôle est d'exécuter une partie du protocole de communication. Cela permet ainsi d'effectuer des traitements qui auraient été à la charge du processeur central. Ces processeurs contribuent à un meilleur recouvrement des communications. En plus du processeur, les cartes disposent également d'une capacité mémoire relativement étendue leur permettant de stocker des messages (de taille raisonnable) en attente de transfert effectif.

Le processeur de la carte réseau dispose aussi d'un ou plusieurs composants *Direct Memory Acces* (*DMA*). Typiquement, il peut y avoir deux *DMA*, un pour le lien sortant et un autre pour le lien entrant. Ces deux *DMA* peuvent fonctionner simultanément, de manière virtuellement autonome, en émission et en réception de messages. Ces *DMA* permettent de libérer le processeur de la carte d'interface des interactions avec le réseau physique. Ils contribuent aussi au recouvrement

des communications. Les *DMA* vont donc lire ou écrire des données directement dans la mémoire centrale. Cette technique sans copie intermédiaire des données est appelée *zero-copy*. Un problème important est que ces données peuvent se trouver dans des pages, qui potentiellement, peuvent être swappées sur le disque. Pour éviter ce problème, les interfaces de communication utilisent des mécanismes d'ancrage permettant de contraindre des pages à rester en mémoire. Lorsqu'une opération de communication impliquant un *DMA* est terminée, l'application et le processeur en sont informés, soit par interruption, soit par scrutation périodique effectuée par l'application.

Commutateur

Le commutateur du réseau constitue un autre composant du réseau qui a un impact déterminant sur les performances. Les commutateurs établissent les chemins que doivent emprunter les messages pour atteindre le nœud destinataire. Le commutateur dispose d'un ensemble de ports qui indiquera le nombre de machines pouvant être connectées. Plus le nombre de ports est grand, plus le nombre de commutateurs nécessaires pour connecter les nœuds de la grappe sera petit et plus la distance en termes de composants (commutateurs + câbles) entre les nœuds sera faible. Une topologie performante a pour objectif de réduire le nombre de parties communes entre les chemins possibles [5]. La plupart des commutateurs, dédiés au réseau haute performance, utilisent une topologie interne en forme de réseau Clos (*Clos Network*)[47]. Cette topologie autorise plusieurs routes possibles entre le port source et le port destination du commutateur. Elle est basée sur une hiérarchie de crossbars, figure 2.9. La route suivie par un paquet dans le commutateur est calculée par un arbre couvrant.

Dans la plupart des réseaux haute performance, le routage, au sein du commutateur, est de type *cut-through*. Le commutateur analyse seulement l'en-tête de chaque message pour en déterminer le port de sortie. Le temps de commutation d'un paquet (*per-hop delay*) pour de tels réseaux est de l'ordre de quelques dizaines de nanoseconde. Le routage est fixé à la source par le contrôleur de la carte réseau. Cette commutation directe est en opposition à celle de type *store-and-forward*, qui stocke, dans les tampons du commutateur, les paquets pour les traiter (recherche d'erreur) et les ordonnancer vers le port destinataire.

Routage

Le routage classique d'un réseau consiste à transférer un paquet de composants réseaux à composants réseaux en totalité et, à chaque étape, d'en contrôler la validité jusqu'à la destination. Cependant, ce type de routage, appelé *store-and-forward*, ne présente pas des performances acceptables pour les réseaux haute performance. Ainsi, une nouvelle technologie de routage a été créée pour les réseaux haute performance. Ce routage se base sur la fiabilité des composants réseaux. Cette fiabilité permet de réduire le nombre de contrôles de validité des paquets (*checksum*), ou de proposer des stratégies réduisant cette contrainte. Le contrôle de validité est généralement effectué à la réception globale du paquet, ce qui peut entraîner une réémission du paquet dans le cas de paquets corrompus. Cependant, ce surplus de trafic généré reste négligeable. Ce routage est composé en deux parties indépendantes :

– La commutation de type *cut-through* [72] exploite parfaitement la fiabilité des composants

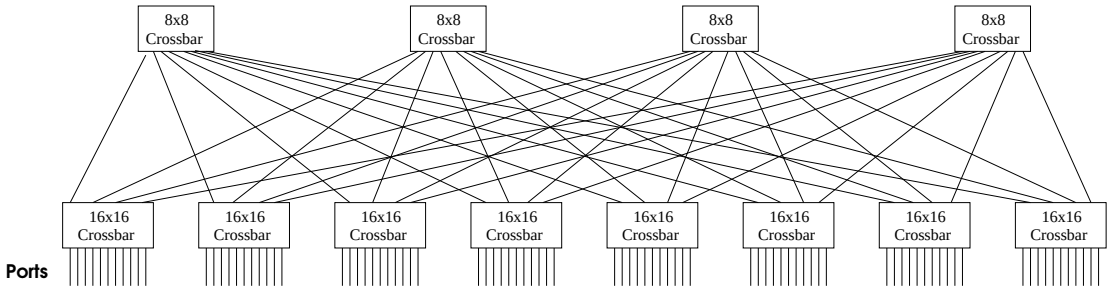


FIG. 2.9 – Réseau Clos 8×16 : topologie interne des commutateurs

réseaux. Cette commutation consiste à transférer un paquet avant que celui-ci soit totalement reçu. Par exemple, dans un commutateur, un paquet commence à être commuté sans nécessairement attendre que le commutateur le reçoive entièrement.

– Le routage *wormhole* [100] décompose un paquet en fragments (*flow control digits*, *flits*) plus petits qui sont transmis en pipeline. De plus, le routage est effectué à la source. Ainsi, le commutateur doit seulement analyser le premier fragment pour commuter, et, grâce à la commutation *cut-through*, transmet directement les fragments du message.

Il en ressort que l'utilisation d'une ou de ces deux technologies ensemble permet une forte réduction de la latence.

2.3.1.2 Technologies logicielles

Les constructeurs de réseaux fournissent des logiciels spécifiques permettant d'exploiter efficacement les caractéristiques du matériel. Les performances en latence et bande passante doivent être accessibles aux programmes développés par les utilisateurs.

Communication intelligente

Comme indiqué précédemment pour la technique de *zero-copy*, les contrôleurs *DMA* et *Remote Direct Memory Access* (*RDMA*) souffrent d'un manque de réactivité au niveau de l'application créant une latence additionnelle. La technique par interruption est responsable de ce surcoût. Pour éviter ce coût additionnel, des techniques logicielles par scrutation ont été développées. Ces techniques par attente active permettent de récupérer plus rapidement un événement réseau, mais en contre partie consomment du temps processeur. Il convient alors d'utiliser la technique par scrutation pour des événements à fréquence d'apparition élevée et d'utiliser les interruptions pour des événements à fréquence basse [68].

Le chemin critique généré par les appels au système d'exploitation est une autre source de perte de performances au niveau de la latence. Permettre l'accès directement des applicatifs aux cartes réseaux en évitant le système d'exploitation procure des latences dépendantes uniquement des technologies matérielles réseaux. Cette technique est communément appelée en anglais *OS-Bypass*.

Dans des grappes de calcul aux nœuds multiprocesseurs, une communication réseau peut-être réalisée au sein d'un seul nœud. De telles communications intra-nœud peuvent correspondre à

de simple copie de mémoire à mémoire. La bibliothèque de communications doit être à même de différencier de telles communications afin d'éviter le coût de descente et de remontée des données vers l'interface réseau. Afin de pleinement exploiter l'intelligence logicielle ajoutée aux communications, un routage efficace est nécessaire.

2.3.2 Principaux réseaux haute performance

Dans cette sous-section, nous présentons trois réseaux haut débit que nous avons étudié plus amplement dans cette thése. Il s'agit de *Gigabit Ethernet*, *Myrinet* et *Quadrics*. Nous invitons le lecteur à se référer à [36] pour des informations techniques sur le réseau *InfiniBand* et [21] pour le réseau *Dolphin SCI*.

2.3.2.1 Gigabit Ethernet

Depuis le début des années 1970, *Ethernet*[16] est devenu le standard des réseaux locaux, en proposant, jusqu'à aujourd'hui, différentes améliorations augmentant les débits de transmission. Les différentes générations d'*Ethernet* sont définies dans le standard *IEEE 802.3*[14]. Principalement, les versions d'*Ethernet* diffèrent par la qualité du câble utilisé (le nom même d'*Ethernet* réfère au câble : l'ether). *Fast Ethernet* a permis le passage d'une bande passante de 10 à 100 Mbits par seconde en partie grâce à la réduction de la longueur des câbles. La version actuelle d'*Ethernet* a une bande passante de 1 Gbits par seconde. Le passage de 100 Mbits vers 1000 Mbits (ou 1 Gbits) a été réalisé, à la fois par une modification la pile logiciel et par l'ajout d'émission de paquets en rafale (*Packet Bursting*). Cette version dénommée *Gigabit Ethernet* propose deux versions : une version sur fibre optique et une version sur câble en cuivre. La version 1000Base-T utilise la totalité des quatre paires croisées du câble en cuivre. Toutefois, *Gigabit Ethernet* n'aura probablement pas la même longévité que *Fast Ethernet* car une nouvelle version d'*Ethernet*, 10 *Gigabit Ethernet* propose un débit dix fois supérieur au réseau *Gigabit Ethernet* actuel. Ce nouveau débit est atteint par l'utilisation unique de liens full-duplex ce qui permet de retirer l'utilisation de technique couteuse pour la détection de collisions entre paquets (*CSMA/CD*).

Les composants des réseaux *Ethernet* ont été produits par plusieurs constructeurs grâce à la définition des standards *Ethernet*. Généralement les performances en termes de bande passante restent très proches du maximum théorique (valant 125Mo/s soit 1 Gbits/s pour *Gigabit Ethernet*). La latence dépend essentiellement des couches logicielles.

Carte réseau

Au niveau des cartes réseaux, les compagnies *Broadcom*[15] et *Intel* sont devenues les deux principaux constructeurs de part le rapport qualité/prix des cartes. A l'inverse, les constructeurs de commutateurs compatibles *Gigabit Ethernet* sont au nombre d'une dizaine. Chaque constructeur introduit des particularités, telles que, par exemple, la taille des tampons, rendant propres les performances à chaque fabricant. A titre de référence les cartes *Gigabit Ethernet* de *Broadcom* pro-

[14] Institute of Electrical and Electronics Engineers

[15] `http://www.broadcom.com`

posent un tampon mémoire de 96 Ko, alors que les cartes *Intel* ont un tampon mémoire de 64 Ko.

Commutateur

La topologie interne des commutateurs *Gigabit Ethernet* dépend essentiellement des choix du constructeur. Cette topologie peut-être simple (tore ou crossbar) ou bien du type réseau Clos[79]. Le réseau *Gigabit Ethernet* peut être un réseau dédié au calcul haute performance, comme celui d'un simple réseau d'entreprise. De ce fait, les constructeurs de commutateur proposent fréquemment deux gammes de commutateur : des commutateurs optimisés pour le calcul haute performance et de simples commutateurs. Les commutateurs optimisés intègrent plusieurs fonctionnalités supplémentaires telles que : la répartition de charge, la protection contre la perte de paquets, ou encore des techniques optimisant les phases de commutation.

Routage

La latence est nettement moins performante et plus hétérogène avec des temps de l'ordre de quelques dizaines de microsecondes. Cette faible performance est due à l'utilisation des protocoles classiques de l'Internet, tels que TCP[46] ou UDP [45], qui sont peu performants en terme de latence. La conception de TCP axée sur la fiabilité en fait un protocole peu efficace sur des réseaux haute performance qui sont par nature formés de composants fiables.

Le mécanisme de contrôle de flux est défini dans l'*IEEE 802.3x*. Lorsqu'un nœud en réception devient congestionné, il envoie une trame *pause frame* à la source pour stopper l'émission. La source attend un délai spécifié dans la trame *pause frame* avant de transmettre à nouveau. Aucune spécification n'est précisée dans le standard sur la durée de la *pause frame* ni sur le taux de congestion déclenchant le contrôle de flux.

La forte croissante des capacités en bande passante du réseau *Ethernet* en fait un réseau incontournable dans les grappes de calcul (plus de 50% des grappes du TOP500). *10 Gigabit Ethernet* se distingue de *Gigabit Ethernet* par l'utilisation exclusive de fibre optique et de lien full-duplex (*Gigabit Ethernet* permet une utilisation half-duplex). La conséquence directe est l'inutilité du protocole de collision d'*Ethernet* (protocole *CSMA/CD*). En outre, *10 Gigabit Ethernet* reste entièrement compatible avec *Ethernet*. Avec une telle bande passante, on peut se demander si les bus *PCI* classique ne deviendront pas bientôt le composant pénalisant les communications.

2.3.2.2 Myrinet

Myrinet [65] est un réseau haute performance développé par la société américaine *Myricom*[16]. La dernière version du réseau *Myrinet* s'appelle *Myrinet-2000*, mais une version hybride compatible avec le réseau *10 Gigabit Ethernet* existe sous le nom de *Myri-10G*.

Myricom développe et construit les commutateurs et cartes réseaux *Myrinet*. Il existe différentes références dépendantes de la taille des tampons ou du nombre de ports pour les cartes et du nombre de nœuds connectables pour les commutateurs. *Myricom* fournit des commutateurs

[16] http ://www.myri.com

et des cartes réseaux mono ou double ports connectées par des liens full-duplex sur fibres spécifiques pouvant délivrer une bande passante maximale de 500 Mo/s dans chaque direction pour la technologie *Myrinet*-2000 et une bande passante maximale de 1200 Mo/s pour *Myrinet*-10G.

Carte réseau

Une carte Myrinet contient un processeur *RISC* programmable (*LANAI*) qui implante une partie du protocole de communication. De plus les cartes réseaux *Myrinet* accèdent à la mémoire centrale grâce à des composants *RDMA* comportant plusieurs moteurs *DMA*. De part ces choix technologiques, *Myrinet* propose une latence faible de l'ordre de 2 μs. La taille des tampons mémoires est soit de 2 Mo soit de 4 Mo. Au-delà des caractéristiques performantes des réseaux *Myrinet*, la possibilité de reprogrammation des cartes et les spécifications libres du matériel ont contribué au développement des réseaux *Myrinet*.

Commutateur

Les commutateurs proposent une topologie sous forme de réseau Clos [47], figure 2.9. Les commutateurs peuvent connecter jusqu'à 512 nœuds. La plus grande grappe comportant un réseau *Myrinet* possède environ 2400 nœuds. Les communications utilisent de manière efficace le réseau *Myrinet* grâce à une commutation de type *cut-through* et à un contrôle d'erreur au niveau matériel.

Routage

Le routage des paquets utilisent un routage à la source basé sur une commutation *cut-through*. Le contrôle de flux utilisé est le protocole *Stop & Go*. Dans le cas de communications concurrentes, la carte congestionnée envoie un symbole *Stop* ou *Go* pour informer la carte émettrice d'arrêter ou de relancer une émission de paquets.

Le réseau *Myrinet* poursuit son évolution vers la réalisation de débit plus important franchissant le giga-octets par seconde de bande passante. Pour ce faire, *Myricom* a choisi de fabriquer une version du réseau *Myrinet* compatible avec le réseau *10 Gigabit Ethernet*. Ce nouveau réseau, *Myri-10G*, est à la fois compatible avec un réseau Myrinet classique et le réseau *10 Gigabit Ethernet* permettant une interconnexion des deux réseaux. L'avantage de cette solution est d'apporter au réseau *10 Gigabit Ethernet* les technologies logicielles réduisant la latence réseau.

2.3.2.3 Quadrics

Le réseau *Quadrics* [31][30] est probablement le réseau ayant la plus petite latence, mais aussi le plus onéreux. Il intègre des liens full-duplex de bande passante de 900 Mo/s et propose une latence de l'ordre de la microseconde.

Carte réseau

Les cartes *Quadrics*, nommées *ELAN*, sont pilotées par un processeur programmable. Force

est de constater que la non-divulgation des spécifications matérielles n'a pas permis un fort développement du réseau *Quadrics* par des tiers. Une innovation des cartes *Quadrics* est l'intégration d'un moteur *Memory Management Unit* (*MMU*). Ce moteur permet la translation directe entre adresse mémoire virtuelle et adresse mémoire physique. Il divise l'espace d'adressage en introduisant la notion de pages et il inclut un cache de translation (*TLB*) entre pages physiques et pages virtuelles. Un tel composant rend inutile les enregistrements mémoires de la mémoire de la carte au près du *MMU* de la mémoire centrale. La latence réseau devient alors très faible. En revanche, l'utilisation d'une telle technologie oblige la modification du système d'exploitation. Toutefois, *Quadrics* propose des pilotes logiciels évitant cette modification, mais au prix de performances moindres. *Quadrics* comporte des moteurs *RDMA* qui, couplés avec les unités *MMU*, autorisent l'adressage de la mémoire du nœud à distance. Il en résulte que le réseau *Quadrics* peut être, en principe, considéré comme un système à mémoire virtuelle partagée. La taille des tampons mémoires intégrés sont de 64 Mo. Les cartes *Quadrics* intègrent d'autres fonctionnalités telles que le *multicast* (diffusion d'un donnée vers un ensemble de cartes).

Commutateur

Les commutateurs *Quadrics*, nommés *Elite*, utilisent une topologie en arbre quaternaire gras (*quaternary fat tree*)[15]. Cette topologie est similaire au réseau Clos. La taille maximale des commutateurs, en nombre de nœuds, est de 4096. Les commutateurs gèrent le contrôle d'erreur des paquets et intègrent deux niveaux de priorité pour aider à la répartition équitable des paquets. La commutation est de type *cut-through* couplé à un routage *wormhole*.

Routage

Le contrôle de flux de *Quadrics* est défini par le routage *wormhole*[29]. Son but est de réduire au maximum le stockage d'un paquet dans le commutateur. Dans un routage *wormhole*, un paquet est divisé en plusieurs petits fragments *flits* diffusés en pipeline. Lorsque le *flits* de tête rencontre un lien déjà utilisé, il est stoppé (ainsi que les *flits* suivants) jusqu'à ce que le lien redevienne disponible. Ainsi, les *flits* sont stockés par plusieurs composants réseaux, ce qui réduit, pour un composant, la taille de stockage nécessaire. La latence *Quadrics* est de l'ordre de 1.5 μs. Comme dans le cas de *Myrinet*, *Quadrics* développe une version de son réseau compatible avec le réseau *10 Gigabit Ethernet*.

2.3.2.4 Bilan des principaux réseaux haute performance

Quadrics et *Myrinet* sont deux réseaux haute performance développés vers le même objectif : l'efficacité en terme de latence et bande passante. La principale différence intervient dans l'échelle de performance/prix. *Quadrics* est plus performant que *Myrinet*, mais aussi nettement plus coûteux. A l'inverse *Gigabit Ethernet* est né de l'amélioration du réseau *Ethernet* qui n'est pas destiné en priorité à être un réseau de grappe. De ce constat, *Gigabit Ethernet* ne comporte pas d'optimisation logicielle ou matérielle spécifique au réseau haute performance. Pourtant, par son prix abordable et ses bonnes performances, *Gigabit Ethernet* est un réseau prédominant dans les

Caractéristiques	*Gigabit Ethernet*	*Myrinet*	*Quadrics*
Technologie	*full-duplex*, optimisation	*full-duplex, RDMA, LANAI*	*full-duplex, RDMA, MMU*
Mémoire sur la carte	< 1Mo	4Mo	64Mo
Routage	Variable (*store-and-forward*)	*cut-through*	*cut-through, wormhole*
Contrôle de flux	trame *pause frame* (*IEEE 802.3x*)	*Stop & Go*	*wormhole* (*flits*)
Max nœuds connectables / commutateur	64-576	512	128
Topologie du commutateur	Variable (crossbar, arbre, tore)	Clos	Arbre gras
Bande passante (Mo/s)	125+125	500+500	900+900
Latence (μs)	$\simeq 10 - 20$	2	1.5

TAB. 2.6 – Résumé des caractéristiques des réseaux haute performance.

grappes de calcul (plus de 50% des grappes du TOP500 comportent un réseau *Gigabit Ethernet*). Le tableau 2.6 résume les différentes propriétés de ces trois réseaux.

Avec l'avènement du *10 Gigabit Ethernet* qui surclasse en bande passante *Quadrics* ou *Myrinet*, on peut s'interroger sur l'avenir de la concurrence entre fabriquant de réseaux haute performance. Bien que *10 Gigabit Ethernet* ne possède pas une faible latence (mais pour combien de temps ?) en raison d'un protocole inadapté aux réseaux haute performance, la suprématie de *10 Gigabit Ethernet* commence à être visible de par l'annonce par *Quadrics* et *Myrinet* de nouvelles versions de leur réseau compatible avec *10 Gigabit Ethernet*. De ce fait, la capacité de bande passante proposée par les réseaux haute performance ne sera plus un critère de sélection.

2.4 Interface de communications pour les applications

Les propriétés caractéristiques des réseaux haute performance sont pilotées par la couche logicielle. Le choix d'implantation de certaines techniques logicielles améliore ou réduit les performances générales du réseau. Lors de l'exécution d'une application utilisant un réseau haute performance, par souci de portabilité, les appels aux routines de communication sont décomposés en deux parties. La première partie concerne la couche logicielle accédant directement au matériel. Cette couche est implantée dans des méthodes et fonctions réalisant le transfert des données sur le support physique. Ces méthodes sont réunies à travers des librairies bas niveau (2.4.1). La deuxième partie correspond à une interface standard de programmation (2.4.2). Ce standard donne lieu à des librairies pour des applications qui accèdent aux fonctions de communications des librairies bas niveau. Ces librairies destinées aux applications assurent la portabilité des opérations de communication.

2.4.1 Librairies bas niveau

2.4.1.1 Socket

La librairie *Socket* est fournie comme librairie de communication standard dans les environnements *Linux* et *Unix*. Elle est très largement utilisée par différentes librairies haut niveau. Une *Socket* est un point de communication qui peut-être adressé par le réseau. De cette manière, deux processus créant chacun une *Socket* peuvent échanger des données. Pour augmenter les performances de la librairie *Socket*, une nouvelle librairie appelée *Fast Socket* a été développée [82]. La principale innovation de cette librairie est l'utilisation de message actif [91]. De manière similaire aux appels de méthodes distantes (*RPC*), un message actif contient les données nécessaires à l'exécution d'une méthode sur la machine destination. De plus, la librairie *Fast Socket* utilise des techniques efficaces de gestion des tampons logiciels en limitant, par exemple, le nombre de copies. Les librairies *Socket* et *Fast Socket* offrent la possibilité d'utiliser deux des protocoles réseaux les plus connus : *UDP*[45] et *TCP*[46].

Unreliable Datagram Protocol (*UDP*) est un protocole de type "sans connexion" qui découpe un message en plusieurs paquets. Chaque paquet est envoyé indépendemment vers le nœud destinataire. Dès lors, les paquets peuvent arriver dans le désordre, plusieurs fois ou être perdus. Il s'agit donc au programmeur utilisant *UDP* de détecter et de traiter ces problèmes.

Transmission Control Protocol (*TCP*), à l'inverse de *UDP*, est un protocole de type "connexion active". En effet, avant l'envoi d'un message, *TCP* établit une connexion entre le nœud émetteur et le nœud receveur. Cette connexion restera active (ouverte) le temps de la communication. En outre, *TCP* inclut un mécanisme de contrôle de flux ou contrôle de congestion[56][93]. Il en résulte, d'une part, une plus grande fiabilité, mais, d'autre part, un coût logiciel beaucoup plus important. Par conséquent, la latence *TCP* est importante et elle est due au choix du protocole (contrôle d'erreur, copie mémoire, tampons). Au niveau des réseaux haute performance qui, par définition, ont des composants matériels fiables, le manque d'intérêt dans l'utilisation d'un protocole tel que *TCP*[95] apparaît évident.

2.4.1.2 Librairie Myrinet Express

Les spécifications matérielles ouvertes de *Myrinet* ont conduit à l'implantation de plusieurs protocoles. La conception du protocole *Basic Interface of Parallelism* (*BIP*) [12] par une équipe académique (*INRIA*) a été l'un des plus abouti. Il utilisait pleinement les capacités de faible latence et de bande passante de *Myrinet*. D'autres travaux académiques ont conduit à des protocoles opérationnels comme *Fast Message* (*FM*) [84] ou *Virtual Memory-Mapped Communication* (*VMMC*) [14]. Aujourd'hui, beaucoup de ces travaux se sont arrêtés, mais les idées et concepts ont été repris dans les protocoles proposés par la société *Myricom*.

Le premier protocole, proposé par *Myricom*, s'appelle *GM* [64]. Ce protocole *GM* bien qu'à la fois robuste et efficace reste moins performant que *BIP*. Pourtant, il fournit des services de plus haut niveau pour détecter la perte des paquets ou les paquets corrompus.

Pour répondre aux faiblesses de *GM*, un nouveau protocole est défini par *Myricom*. Ce protocole, appelé *Myrinet Express* (*MX*)[63], propose de faire converger *BIP* et *GM*. Il se caractérise d'abord par l'utilisation de communications *RDMA*, d'une meilleure gestion des terminaisons des communications, et ensuite par l'incorporation de primitives réalisant les opérations collectives de

diffusion et barrière. Le protocole *MX* a la particularité de classer les messages en trois catégories suivant leur taille : petits messages (inférieure à 128 o), moyens messages (compris entre 128 o et 32 Ko) et de grands messages (supérieur à 32 Ko). En outre, il optimise de manière différente les accès en mémoire suivant chaque catégorie de messages.

MX, comme *BIP*, propose une interface de programmation logicielle par passage de message. Son implantation intègre la technologie de *OS-bypass* et comporte une interface de notification de terminaison des requêtes.

2.4.1.3 Librairie Quadrics Elan4

ELANLIB est la librairie de communication bas niveau proposée par *Quadrics*. Elle permet un adressage virtuel de l'espace mémoire des nœuds d'une grappe. Cet adressage ne concerne pas uniquement la mémoire des cartes d'interface réseau, mais également la mémoire centrale du nœud. Ainsi, *ELANLIB* donne la possibilité de voir la grappe comme un système global à mémoire partagée. La technologie de communication *Quadrics* est basée sur un système *RDMA* composé d'opérations *put* et *get* sur la mémoire de la carte. Ces opérations peuvent être étendues aux mémoires centrales des nœuds.

En outre, *ELANLIB* intègre une technologie *OS-bypass* et permet au niveau utilisateur l'adressage de la mémoire sur la carte, sans recourir à des copies de données. Une particularité de *Quadrics*, est d'inclure des threads de notification d'événements s'exécutant sur le processeur de la carte et résidant sur la mémoire de la carte. Il en résulte que la librairie *ELANLIB* obtient un contrôle performant des événements réseaux.

ELANLIB inclut une librairie de communication point à point par passage de message (*Tports*). Cette librairie est à la base des implantations *MPI* sur *Quadrics*.

Les spécifications du réseau *Quadrics* ne sont pas du domaine public. En conséquence, seule la compagnie *Quadrics* fournit des librairies et protocoles de bas niveau.

2.4.2 Interface de communication *MPI*

La programmation parallèle sur grappe de calcul s'effectue principalement autour d'un modèle de tâches communicantes. Le modèle est directement dérivé du modèle introduit par C.A.R. Hoare [19]. Un programme parallèle va donc être constitué d'un ensemble de tâches séquentielles qui vont communiquer par des envois de messages. *Parallel Virtual Machine* (*PVM*) a été l'un des environnements pionniers implantant ce modèle sur des grappes de calcul. L'interface de communication *MPI* est née de la volonté de proposer une interface standard pour la programmation parallèle et distribuée. Un consortium avec des partenaires académiques et industriels s'est donc créé pour définir l'interface *MPI*.

Message-Passing Interface (*MPI*) [62] propose une interface de programmation définissant un ensemble de fonctions de communication ainsi que des recommandations pour leurs implantations. Plus précisément, *MPI* propose plusieurs méthodes de communication point à point telles que *MPI_Send* (envoi) et *MPI_Recv* (réception) ainsi que des communications collectives comme, par exemple, *MPI_Barrier* (barrière) et *MPI_Broadcast* (diffusion). En outre, afin d'accroître la portabilité des applications, *MPI* propose des types de données de base et donne la possibilité de créer des types complexes et structurés.

D'autre part, *MPI* recommande d'utiliser un protocole par *Rendez-vous* visant à gérer la synchronisation et l'allocation mémoire entre les processus émetteur et récepteur. De manière plus précise, lorsqu'un processus initie une communication de grande taille, il envoie un premier message au processus destinataire afin qu'il puisse en préparer la réception. Le standard *MPI* ne précise pas la limite entre une communication de grande taille et de petite taille. Cette limite est spécifique à chaque implantation.

MPI est un standard très largement utilisé dans le calcul parallèle, générant plusieurs implantations propriétaires et libres. Dans la suite de ce document, une présentation de deux implantations *MPI*, issues du logiciel libre, est proposée. Par suite, les performances de ces deux implantations seront évaluées sur une même architecture réseau.

2.4.2.1 Implantation *Mpich*

Mpich [98] est une implantation de *MPI* portable grâce à une interface de pilote abstraite (*Abstract Device Interface, ADI*). L'*ADI* est une couche intermédiaire qui décrit les fonctions de communication utilisées par l'implantation *MPI*. Ces fonctions reposent sur les librairies bas niveau précédemment citées. Il en résulte que seules les fonctions de communication décrites dans l'*ADI* ont besoin d'être réécrites pour fournir une nouvelle implantation de *Mpich* spécifique à une architecture réseau. De plus, les opérations de communications collectives utilisent des appels aux fonctions de communication point à point. Cependant, l'*ADI* permet d'écrire ses propres fonctions de communication collectives pour utiliser, par exemple, des fonctions matérielles (barrière). Cela permet à *Mpich* de facilement proposer une compatibilité avec plusieurs réseaux haute performance. La conception portable de *Mpich* est également à la base de plusieurs implantations *MPI* propriétaires.

La nouvelle version de *Mpich* : *Mpich2 genesis* [96] intègre de nouvelles optimisations qui ont comme principal but de réduire le surcoût logiciel et de ce fait la latence réseau. *Mpich2* intègre également des améliorations algorithmiques, issues de la recherche internationale, pour les communications collectives.

L'implantation *Mpich* est développée au laboratoire Argonne (*Argonne National Laboratory*).

2.4.2.2 Implantation *Lam/MPI*

Lam/MPI [32] est une autre implantation libre de *MPI*. La particularité de *Lam/MPI* est de lancer un démon sur chaque nœud préalablement au déploiement des processus applicatifs. Ce démon permet la gestion des rangs (*ranks*) *MPI* ainsi que diverses options : système de traçage et de déboguage. Cependant, son principal but est de permettre l'utilisation de *Lam/MPI* sur des nœuds aux architectures hétérogènes. Un avantage de la gestion des processus par le démon est de faciliter le placement précis des processus sur les nœuds.

Lam/MPI comporte un système de couche abstraite (*RPI*) similaire à l'*ADI* de *Mpich*, mais moins aboutie. Il en ressort que *Lam/MPI* n'est que très peu utilisée comme souche pour le développement de versions industrielles et propriétaires. Par conséquent, *Lam/MPI* propose une couche logicielle spécifique à certains réseaux. A titre d'exemple, *Lam/MPI* fournit une couche logicielle pour le protocole *GM* de *Myrinet*, alors qu'aucun développement ne sera réalisé pour le nouveau

	Lam/MPI	*Mpich*	*Mpich* optimisé en bande passante
Latence [μs]	45.8	47.5	47.7

TAB. 2.7 – Comparaison des latences.

protocole *MX*. En effet, le développement de *Lam/MPI* a été stoppé pour donner naissance à son successeur : *OpenMPI* [27].

OpenMPI est la fusion de quatre implantations *MPI* créées par quatre laboratoires de recherche. *OpenMPI* intègre les implantations *LA-MPI* (*LANL*[17]) et *FT-MPI* (*ICL*[18]) qui permettent la tolérance aux fautes. Le but d'*OpenMPI* est de devenir la meilleure implantation *MPI* aussi bien au niveau des performances que des fonctionnalités proposées.

2.4.2.3 Comparaison des performances

Dans cette sous-section, une étude sur l'impact de l'implantation *MPI* est présentée. Cette étude se base sur le programme de test *NetPIPE* [75]. *NetPIPE* peut-être utilisé en combinaison des interfaces *MPI* présentées précédemment.

En utilisant *NetPIPE*, nous proposons des mesures de bande passante et de latence. *NetPIPE* propose la mesure d'une communication aller-retour (*ping-pong*) pour une taille de message variable. Le nombre de tests réalisés dépend de la taille du message. En effet, plus le message est de petite taille, plus le temps de communication est sensible à de faibles variations des états initiaux de l'expérience (réseau, système d'exploitation). Partant de cette constatation, *NetPIPE* effectue un plus grand nombre de tests (5000) pour des petits messages et moins d'une dizaine pour des messages de grande taille. De plus, la taille des messages varie suivant une puissance de 2 et une valeur proche de cette puissance de 2 ($2^n - 3$). Ceci afin d'éviter, dans les mesures, un possible biais dû aux uniques tests sur des puissances de 2. Dans ces expériences, nous prenons l'architecture réseau *Gigabit Ethernet* (grappe de Nancy, cf annexe A), dont la couche logicielle *MPI* est généralement développée en première dans des implantations généralistes. Les performances *MPI* pour *Myrinet* et *Quadrics* ont été présentées dans [43].

Le tableau 2.7 et la figure 2.10 montrent que *Lam/MPI* a une meilleure latence et bande passante que *Mpich*. Il est important de noter que la différence entre ces deux implantations révèle l'importance, du point de vue des performances, des choix d'implantation du standard *MPI*.

De plus, certaines implantations peuvent être configurées afin d'obtenir de meilleures performances. C'est le cas de *Mpich*, qui sur l'architecture *Gigabit Ethernet*, autorise un paramétrage de la taille des tampons utilisés par la librairie *Socket*. Le choix de ces options a un grand effet sur les performances comme l'ont démontré les auteurs de [26]. Dans les tests que nous effectuons, nous évaluerons *Mpich* avec et sans le paramétrage des tailles de tampons (*P4_SOCKBUFSIZE*). Le paramètre par défaut est de 32 Ko que nous avons augmenté à une valeur de 4Mo.

Le tableau 2.7 et la figure 2.11 montrent que la bande passante est fortement influencée par cette option, alors que la latence reste inchangée.

[17] Los-Alamos National Laboratory
[18] Innovative Computing Laboratory

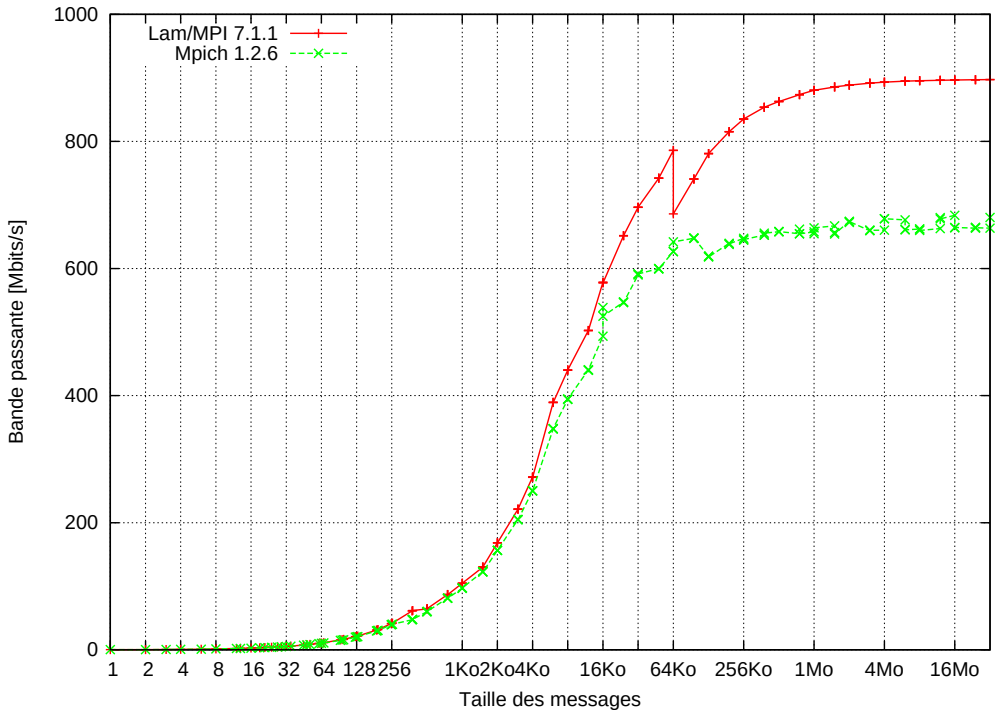


FIG. 2.10 – Comparaison des implantations *Lam/MPI* et *Mpich* sur *Gigabit Ethernet*

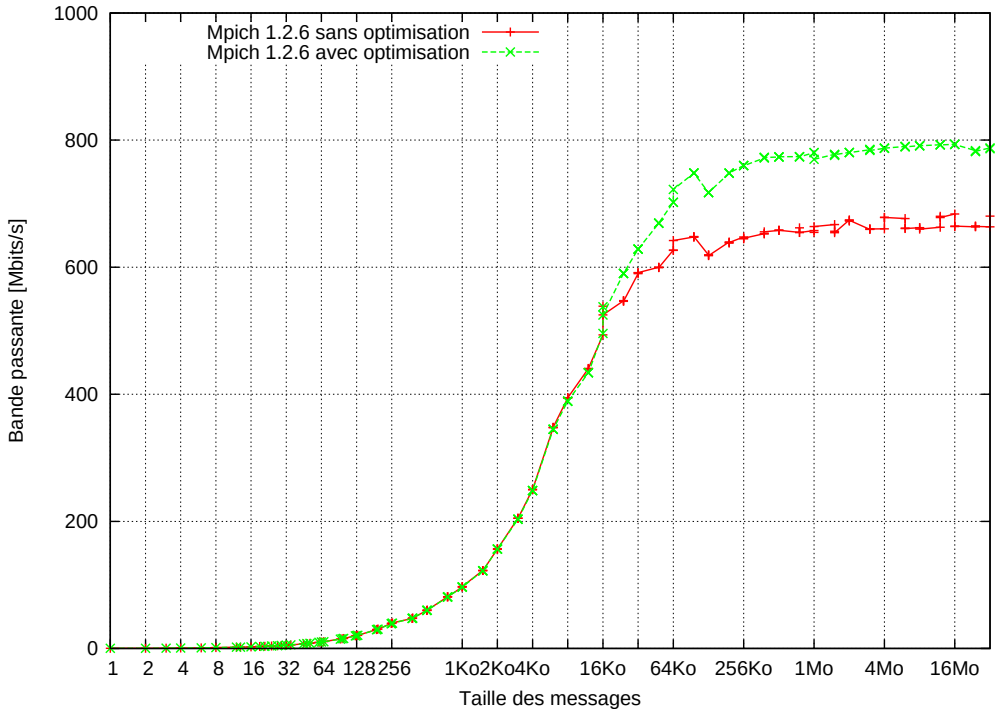


FIG. 2.11 – Gain par optimisation de *Mpich* sur *Gigabit Ethernet*

2.5 Conclusion

Une grappe de calcul se construit autour d'un ensemble de nœuds de calcul interconnectés par un réseau à haute performance.

Les nœuds de calcul se divisent en trois éléments majeurs : l'unité de calcul ou processeur, l'architecture mémoire et le bus reliant les périphériques. Ces trois éléments influencent les performances. Au niveau des processeurs, nous avons vu que les performances varient en fonction de l'architecture du processeur et la structure des caches. Il est aussi important de noter que les performances théoriques annoncées par les constructeurs ne permettent pas de déterminer les performances effectives des programmes. Par exemple, le compilateur a une influence prépondérante puisqu'il va exploiter au mieux les spécificités de l'architecture. Il en résulte une différence entre les performances théoriques, fournies par les constructeurs, et les performances réelles obtenues sur de vraies applications. L'architecture mémoire, en particulier l'architecture *NUMA*, propose des structures mémoires hiérarchiques qui ont un très fort impact sur les performances des applications. Avec de telles structures hiérarchiques, il est difficile de prédire, lors de l'accès à une donnée, son emplacement soit en cache (cohérence de cache), soit en mémoire centrale et l'état des bus pour y accéder. Finalement, le bus reliant les périphériques est probablement l'élément le moins complexe, et donc, le plus prévisible en termes de performances.

Les réseaux haute performance se distinguent des réseaux classiques par des optimisations matérielles et logicielles. L'objectif des réseaux haute performance est de réduire la latence et d'augmenter la bande passante. Nous avons décrit plus particulièrement trois catégories de réseaux haute performance : *Gigabit Ethernet*, *Myrinet* et *Quadrics*. *Myrinet* et *Quadrics* proposent des technologies matérielles et logicielles principalement dédiées pour le calcul à haute performance. On peut citer, par exemple, les accès *RDMA*, le routage *wormhole* ou bien les techniques de *OS-Bypass*. *Gigabit Ethernet*, à l'inverse, est un réseau plus classique de type *Ethernet* qui est largement utilisé pour les réseaux locaux d'entreprise (Intranet) avec des performances le situant dans la catégorie des réseaux haute performance.

Les librairies de communications bas niveau ainsi que les interfaces de communications haut niveau permettent de tirer partie de ces technologies. En particulier, l'interface de programmation par passage de message, *MPI* est très largement utilisée dans le calcul scientifique sur grappe. Les performances de *MPI* ont été amplement étudiées et améliorées.

La capacité de calcul d'une grappe dépend du nombre de nœuds de calcul et du nombre d'unités de calcul par nœud. Cette capacité mesurée en nombre d'opérations à virgule flottante par seconde (*flops*) est un des premiers critères de performance retenu qui est souvent utilisé pour évaluer la puissance globale d'une grappe. Cette mesure ne peut pas être considérée comme le seul critère de comparaison, mais il reste, néanmoins, le plus populaire. Le programme *High-Performance Linpack* (*HPL*) [19] permet d'obtenir une mesure du nombre d'opérations flottantes par seconde d'une grappe en résolvant une série d'équations d'un système linéaire. Ce programme ou *benchmark* est utilisé par le site TOP500. *HPL* propose deux mesures : *Rmax* et *Rpeak*. *Rmax* représente la performance maximale obtenue lors de l'exécution de *HPL* alors que *Rpeak* est le maximum théorique de la grappe (nombre d'opérations théoriques multiplié par la fréquence du processeur et par le nombre de processeurs de la grappe). La différence entre ces deux mesures

[19] *High-Performance Linpack*, `http://www.netlib.org/benchmark/hpl/`

peut-être importante. En effet, la valeur théorique est une borne supérieure, car elle ne prend pas en compte la dégradation induite par les autres composants de la grappe. La valeur pratique (par *HPL*) donne une valeur plus proche de la réalité. Cependant, elle ne peut pas à elle seule évaluer la réelle puissance d'une grappe. Actuellement, la machine la plus puissante du classement TOP500 délivre environ un *Rmax* de 280 Teraflops et un *Rpeak* de 367 Teraflops pour 131072 processeurs alors que la moins puissante délivre un *Rmax* d'environ 2 Teraflops et *Rpeak* d'environ 5,7 Teraflops pour 1028 processeurs.

Il en ressort que la compréhension du comportement applicatif sur des grappes de calcul est un problème complexe. Cette complexité provient du grand nombre de paramètres provenant du matériel et du logiciel dont l'impact sur le système est prépondérant. Par exemple, le débit d'un bus ou la taille d'un tampon mémoire peuvent ralentir l'accès à une ressource particulière. Ainsi, la modification d'un seul paramètre peut améliorer ou réduire significativement les performances d'un programme parallèle. D'autre part, l'identification des paramètres dominants est une tâche complexe. Il est souvent difficile d'en isoler le seul effet. Par exemple, lors d'une communication réseau, il est difficile de distinguer l'influence de la taille du tampon physique de la carte réseau par rapport au choix de la taille du tampon logiciel sur le temps de communication. On peut imaginer que dans le cas de communications multiples partant d'une même source, une telle distinction augmente en complexité.

L'un des composants le plus pénalisant dans l'exécution d'une application distribuée est le réseau. D'une part, l'ordre de grandeur du temps d'une communication est largement supérieur à celui d'un accès mémoire ou d'un accès disque. D'autre part, au niveau du développement de l'application, l'ajout d'une communication inclut fréquemment le besoin d'une synchronisation et donc d'un temps d'attente. De ce fait, nous concentrons le reste de ces travaux de thèse en l'étude des communications entre tâches d'une application sur différents réseaux.

L'analyse des performances va mettre en évidence les possibilités de prédiction du comportement de systèmes ou d'applications. Cette prédiction se basera sur des modèles décrivant leurs comportements. Avec un modèle approprié, il sera ainsi possible de prédire le comportement d'applications s'exécutant sur des grilles de calcul. De tels modèles sont souvent complexes à formuler. Dans le chapitre suivant, nous présentons un état de l'art des différents modèles permettant de comprendre et de prédire le comportement logiciel et matériel des grappes de calcul. Cette présentation se focalise sur les modèles de communication.

Dans la suite de ce document, les plates-formes utilisées pour les expérimentations seront décrites et référencées dans l'annexe A.

Communications dans les grappes : approches de modélisation 3

3.1 Introduction

La modélisation du comportement des systèmes informatiques a plusieurs objectifs. Elle permet, par exemple, de mieux comprendre et appréhender certains phénomènes propres au système. Elle donne aussi la possibilité de prédire le comportement du système dans différents contextes. Une modélisation se base principalement sur la description des mécanismes utilisés par le système.

Cependant, si nous regardons le problème global des performances d'applications sur les grappes afin de les modéliser, nous nous confrontons à un problème difficile. En effet, la description du contexte matériel et logiciel des grappes montre une grande disparité qui est à l'origine de phénomènes influant sur les applications et leurs performances. Certains composants ou choix de logiciels peuvent faire varier significativement les temps ou les besoins en ressources des applications. L'impact et l'interaction de ces facteurs logiciels ou matériels montrent une grande difficulté dans la définition de modèles.

Selon nous il convient de mettre en avant le pragmatisme à respecter dans la recherche de modèles de grappe. Le modèle a pour objectif de permettre des prédictions précises sur le comportement des applications. Les modèles à retenir sont ceux dont la mise en œuvre est réalisable et efficace. Par exemple, la détermination des paramètres nécessaires aux calculs des prédictions et la complexité de ces calculs doivent permettre l'obtention des prédictions.

Ainsi, lorsque l'on souhaite proposer un modèle d'un système complexe, il convient d'associer au modèle deux critères fondamentaux : la simplicité de sa mise en œuvre et la précision du modèle dans la reproduction du système. Bien souvent, on imagine qu'un modèle complexe implique des précisions fiables et un modèle simpliste des précisions médiocres. Cependant, cela n'est pas toujours vrai. En effet, généralement, un modèle complexe ne fournit pas de meilleures précisions qu'un modèle simple, mais au contraire, sa complexité (grand nombre de paramètres difficiles à établir) limite à la fois son utilité et son efficacité. Par exemple [41], une simulation fine peut donner des résultats moins efficaces qu'une simulation restreinte aux caractéristiques principales du système. Un modèle doit être caractérisé par un rapport complexité/précision.

Un des systèmes informatiques qui a été amplement étudié sont les réseaux de communication. Notamment, la recherche de modèles de communication est apparue comme un axe principal. De ce fait, face à l'étendue des travaux réalisés, nous proposons de réduire ce chapitre à la présentation de modèles susceptibles d'appréhender le comportement des réseaux haute performance.

Dans ce chapitre, nous explorons la problématique de la modélisation des réseaux. Dans une première partie, nous nous focaliserons cette étude sur la modélisation des communications point à point. Dans une seconde partie, nous abordons la modélisation des phénomènes de congestion réseau. Pour chaque modèle de réseau présenté, nous essaierons d'examiner sa capacité à modéliser les réseaux haute performance.

3.2 Problématique

Une communication réseau consiste en un transfert de données entre une entité émettrice et une entité réceptrice. Ce transfert est soumis à différents éléments caractérisant le réseau. Le protocole utilisé [71], les caractéristiques techniques du réseau [43], le nombre de composants réseaux traversés sont autant de facteurs exerçant une influence sur la transmission des données.

Il est important de spécifier pour chaque modèle les caractéristiques du réseau modélisé. A titre d'exemple, un modèle du réseau Internet ou d'un réseau *Wide Aera Network* (*WAN*) ne convient probablement pas pour la modélisation d'un réseau à haute performance et dédié à une grappe. Le comportement des réseaux de grappe, par leur aspect dédié, est plus apte à être capturé par certaines techniques de modélisation.

Ces réseaux présentent plusieurs propriétés utiles pour la construction de modèles. Ces propriétés sont :

- la fiabilité des composants qui implique des résultats reproductibles comportant une faible variance ;
- les systèmes de transmission des données (routage) ou de contrôle de flux présentant un caractère simple.

Finalement, ces deux propriétés mettent en valeur l'aspect simple et reproductible des communications réseaux. Partant de ce constat, une approche déterministe semble être plus adéquate pour modéliser les réseaux haute performance.

3.3 Modèle de communication point à point

Une communication point à point consiste d'en l'envoi de données entre deux nœuds de la grappe. Cette transmission ne comporte aucun effet de congestion dû à d'autres communications.

3.3.1 Modèle de Hockney

Le modèle de Hockney [78] est historiquement un des premiers modèles de communication, ce qui en fait l'un des modèles les plus utilisés. Pour calculer le temps d'une communication, ce modèle introduit deux paramètres : la latence et la bande passante. La latence correspond au temps minimal de traversée du réseau alors que l'inverse de la bande passante correspond au taux de service maximum qu'offre le réseau. Ces deux paramètres ont été largement utilisés dans d'autres modèles. Le modèle de Hockney combine ces deux paramètres à travers une équation affine :

$$t(m) = L + m/B$$

Ainsi, le temps d'une communication $t(m)$ qui envoie une quantité m de données est fonction de la latence L et de la bande passante B.

D'une manière plus formelle, le calcul de la latence L correspond à la durée d'envoi d'une quantité de message nulle. Elle se mesure en seconde. A l'inverse, la bande passante est le rapport entre une taille de message "tendant vers l'infini" et sa durée de transfert. Elle correspond au débit et se mesure en octet par seconde. Pour obtenir les valeurs de ce deux paramètres il existe différents outils de mesure dont entre autre [66][75].

Un modèle similaire au modèle Hockney a été proposé dans [35]. Il combine ces paramètres à travers une équation hyperbolique. Ce modèle ne propose pas de meilleures performances que le modèle de Hockney. Toutefois, il a ouvert la voie au concept de temps de latence proportionnelle à la taille de message. Ce concept sera repris dans les modèles de la famille LogP, en introduisant les concept de surcoût logiciel (*overhead*).

3.3.2 Famille de modèle LogP

Le modèle de Hockney calcule les temps de communications en fonction d'une simple équation affine et de deux paramètres. La famille des modèles LogP est apparue afin d'exprimer de manière détaillée les mécanismes intervenant dans une communication. De ce fait, ces modèles sont plus complexes et augmentent le nombre de paramètres utilisés.

Dans le cas de communication point à point, sans congestion, la famille LogP comporte trois modèles : le modèle LogP [22], le modèle LogGP [1] et le modèle pLogP [88].

3.3.2.1 Modèle LogP

Le modèle LogP [22] est le modèle d'origine dont découlent les autres modèles de la famille LogP. Le modèle LogP définit quatre paramètres comme suit :

- L (*Latency*) : qui correspond à la latence réseau ;
- o (*overhead*) : le coût logiciel induit par le mécanisme de communication, c'est-à-dire la traversée des couches logicielles et préparation du message ;
- g (*gap*) : le temps intrinsèque entre deux envois ou réceptions de paquets ;
- P (*Processors*) : le nombre de processus mis en jeu.

Trois de ces paramètres sont combinés par l'équation suivante :

$$2 \times o + L + \lceil \frac{k}{w} \rceil \times max(g, o)$$

Cette équation calcule le temps nécessaire à l'envoi de k octets divisés en w paquets. A la différence du modèle de Hockney, pour caractériser le taux fixe de transfert, le modèle LogP ajoute à la latence les coûts logiciels induits par l'émission et par la réception. Ces coûts correspondent à la création de paquets, l'encapsulation, etc. Un taux variable est associé aux différents paquets. Lorsque le modèle de Hockney précise une bande passante commune pour chaque taille de message, sans introduire la notion de paquets, le modèle LogP ajoute une contrainte entre paquets. Cette contrainte stipule que deux paquets consécutifs ne peuvent pas être transmis en moins de g unités de temps. Ce paramètre g représente la durée pendant laquelle le réseau transmet un paquet. Le modèle LogP donne des résultats efficaces pour des messages de petite taille avec peu de paquets. Néanmoins, pour des messages de grande taille les résultats sont moins précis. En effet, les

valeurs de g et o sont identiques quel que soit la taille des messages, et ce alors qu'il paraît logique que le surcoût logiciel est plus important pour de grands messages que pour de petits messages. Pour remédier à cette perte de prédiction, le modèle LogGP a été introduit.

3.3.2.2 Modèle LogGP

Le modèle LogGP [1] est très similaire au modèle LogP. Ce modèle ajoute le paramètre G qui représente une valeur du gap non plus fix mais fonction de la taille des messages. En d'autres termes, le gap G représente l'inverse de la bande passante. Avec ce nouveau paramètre l'équation du modèle précédent se réécrit par :

$$2 \times o + L + (k-1) \times G$$

Nous noterons la grande similitude de cette équation avec celle du modèle de Hockney. Le modèle LogGP ne propose pas de grandes avancées à la fois dans la modélisation et dans la compréhension des mécanismes de communication par rapport aux modèles précédents.

3.3.2.3 Modèle pLogP

Le modèle pLogP (*parameterized LogP*) [88] introduit une nouvelle façon de considérer les paramètres du modèle LogP. Ce modèle a pour paramètre le surcoût logiciel et le gap en fonction de la taille des messages. En outre, il distingue le surcoût logiciel en émission et en réception. Ainsi, le paramètre o devient $o_s(m)$ et $o_r(m)$, et le paramètre g s'écrit $g(m)$.

L'utilisation de communications bloquantes réduit l'impact et l'utilité des paramètres $o_s(m)$ et $o_r(m)$. Cette remarque, identifiée par les auteurs du modèle, a conduit à l'écriture de l'équation sous la forme :

$$L + g(m)$$

Cependant, pour des communications non-bloquantes il convient de réintroduire les paramètres $o_s(m)$ et $o_r(m)$.

Les auteurs vont plus loin que la simple expression du modèle. En effet, ils fournissent sur leur site Internet[1] un programme qui permet d'évaluer chaque paramètre en fonction de la taille des messages. Ce programme propose plusieurs innovations. Par exemple et de manière simplifiée, le gap pour de petites tailles de messages est calculé en divisant le gap obtenu pour des messages de grande taille par le temps d'aller-retour d'un message de taille nulle (RTT(0)). D'une manière similaire, les auteurs présentent une méthode détaillée pour la mesure des paramètres $o_s(m)$ et $o_r(m)$.

Ce modèle a deux principaux avantages : la flexibilité des paramètres o et g, fonction de la taille des messages, et l'existence d'un programme évaluant, pour un réseau donné, ces paramètres. Toutefois, il apparaît que ce modèle consiste principalement à obtenir des mesures depuis un programme qui teste le réseau. Il ne propose pas une équation pour déterminer $g(m)$. Pour répondre à ce problème, les auteurs de ce modèle proposent une correspondance entre les paramètres $o_s(m)$, $o_r(m)$ et $g(m)$ et les paramètres du modèle LogGP (o, g et G).

[1] `http://www.cs.vu.nl/albatross`

La famille de modèles LogP a été largement utilisée, modifiée et adaptée à différents problèmes. Certains auteurs [33] ont par exemple introduit le nombre de processeurs comme paramètre dans des équations linéaires basées sur le modèle LogGP.

Dans la sous-section suivante, nous nous intéressons à l'utilité de ces modèles pour modéliser les communications *MPI*. Nous présentons des travaux de recherche en étudiant leur pertinence sur des réseaux haute performance.

3.3.3 Modèle de communication et *MPI*

L'utilisation du modèle LogGP a été introduit dans le cas de programmes parallèles simples tel que le tri [3]. De manière similaire, ce modèle a été appliqué à des problèmes tels que le calcul parallèle redondant [101], ou par exemple la distribution de données [94].

Avec l'apparition du standard *MPI* et d'implantations libres, l'utilisation du modèle LogGP s'est très rapidement amplifiée. Dans [25], les auteurs utilisent conjointement le modèle LogP et *MPI* pour analyser le comportement d'une application scientifique. Ils introduisent une modification du modèle LogGP en proposant un modèle composé de deux équations linéaires. Cette modification est causée par le standard *MPI* qui introduit un protocole de rendez-vous pour gérer les messages de grande taille. En simplifiant, ils ajoutent une latence supplémentaire et un gap différent pour modéliser les effets de ce protocole. Cette technique n'est pas sans rappeler le modèle pLogP comportant un gap et un surcoût en fonction de la taille des données envoyées. Néanmoins, même si les auteurs présentent une modélisation efficace, il est intéressant de décrire leur protocole expérimental. En effet, les auteurs utilisent des nœuds SMP (4 processeurs) mais ne placent qu'une tâche *MPI* par nœud. Il en résulte que des phénomènes tels que le partage de la carte réseau ou le bus PCI ne sont pas pris en compte. Nous verrons dans les chapitres 4 et 6 l'influence significative des partages des ressources d'un nœud.

Un dernier cas d'application des modèles de communication point à point a été réalisé pour l'analyse des opérations collectives incluses dans *MPI*. Les auteurs de [28] proposent d'étudier les différents modèles précédents en appliquant leurs prédictions sur l'opération collective de diffusion *MPI_Broadcast*. Leurs conclusions sur l'utilisation des modèles démontrent que le modèle pLogP est le plus précis dans ces prédictions. Les auteurs remarquent aussi que les modèles de Hockney et LogGP sont très similaires. Ces deux modèles proposent une modélisation trop pessimiste dans le cas de messages de petites et moyennes tailles et trop optimiste pour les messages de grande taille.

Afin d'avoir un aperçu de la précision de ces modèles sur les réseaux et implantations *MPI* actuels, nous évaluons leurs prédictions dans le paragraphe suivant.

Comparaison et mesure : pour évaluer leurs prédictions, nous utilisons le programme *NetPIPE* [75] et le réseau *Myrinet* de la grappe de Sophia (confère annexe A). Dans le cas du modèle de Hockney, les expériences consistent à déterminer les paramètre L et B de l'équation affine par *NetPIPE*. En outre, nous utilisons le modèle pLogP pour évaluer la pertinence de la famille LogP. Ce modèle met à disposition un programme permettant d'évaluer les paramètres nécessaires aux prédictions ($g(m)$ et L). De plus, il propose une correspondance entre ces paramètres mesurés et le modèle LogGP, ce qui nous permet d'ajouter le modèle LogGP à notre comparaison. Cette

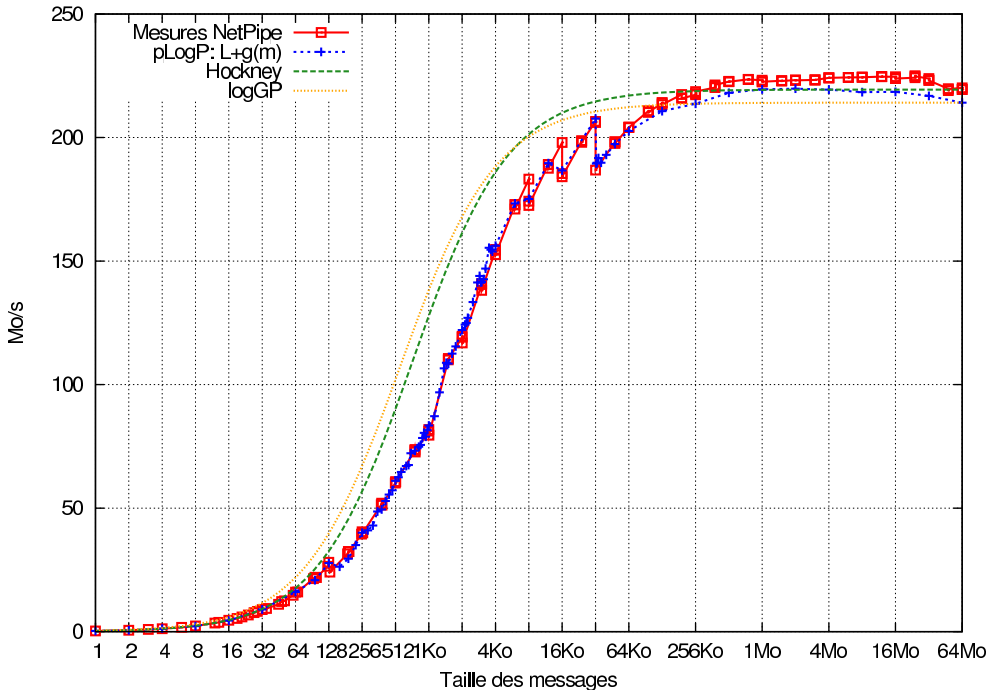


FIG. 3.1 – Comparaison des modèles, grappe de Sophia, réseau *Myrinet*

comparaison confronte les valeurs obtenues par *NetPIPE* et celles prédites par ces trois modèles pour toutes les tailles de messages.

Les comparaisons des prédictions et des mesures réelles sont présentées dans la figure 3.1. Il convient de relever deux points quant à l'analyse de cette figure :

- le modèle de Hockney et le modèle LogGP proposent des valeurs proches de celles mesurées par *NetPIPE* pour des tailles de messages petites (< 64o) et pour des messages de grandes tailles (> 256Ko). Pour des messages de taille intermédiaire ces modèles montrent une déviation des prédictions supérieures aux mesures. Ces deux modèles surestiment le temps des communications de taille intermédiaire.
- le modèle pLogP est plus proche des valeurs mesurées par *NetPIPE*. Cependant, nous pouvons nous demander quel est l'apport de ce modèle par rapport à *NetPIPE* ? En effet, un tel modèle propose une méthode pour mesurer $g(m)$ ce que fait également *NetPIPE*.

3.3.4 Conclusion

En conclusion, le modèle de Hockney reste un modèle simple et pertinent, mais peu descriptif quant aux différents mécanismes utilisés lors d'une communication. Les modèles LogP et LogGP introduisent une plus grande description mais restent très similaires au modèle de Hockney au niveau des prédictions. Le modèle pLogP propose des paramètres dépendants de la taille des messages. Cependant, le calcul du gap pour chaque taille de message est nettement pénalisant. De plus, nous verrons dans le chapitre suivant que les valeurs du gap sont aussi fonction du contexte des communications. Nous définissons par contexte le nombre de communications sortantes ou

entrantes sur chaque nœud de la grappe. Il est difficile de déterminer suivant chaque contexte de communication l'ensemble des valeurs du gap. Il est préférable de déterminer une expression de la répartition du gap suivant le contexte.

Pour conclure nous retenons que l'application du modèle pLogP a atteint une limite, et que les prédictions du modèle LogGP sont équivalentes à celles du modèle de Hockney. De ce fait, le modèle de Hockney reste un modèle simple et efficace.

Après avoir traité des modèles de communications sans congestion, il est intéressant de décrire les modèles de congestion ou de concurrence entre communications.

3.4 Modèle de concurrence

Lorsque plusieurs communications interagissent, se créent des phénomènes de congestion [57]. Ces phénomènes se caractérisent fréquemment par un surcoût du temps de communication. L'impact de la congestion réseau sur les temps d'exécution d'applications parallèles a été étudié dans [76]. Le coût supplémentaire ajouté par la congestion est difficile à modéliser par des modèles simples. Une extension à ces modèles consiste à modéliser l'interaction entre communications.

Il a été proposé plusieurs techniques de modélisation de ces interactions. Certaines de ces techniques consistent à ajouter des propriétés aux modèles linéaires, alors que d'autres introduisent des notions mathématiques plus évoluées telles que les processus stochastiques. Toutefois, la modélisation de la congestion est une étude difficile. En outre, l'applicabilité de certains modèles au contexte de réseaux haute performance est limitée.

Nous divisons cette section suivant les différents axes de recherche relatifs à la modélisation de la congestion réseau. Pour chaque axe, nous commenterons leur capacité à appréhender les mécanismes de congestion des réseaux haute performance.

3.4.1 Modèles linéaires augmentés

Un des premiers axes de recherche est l'ajout de paramètres, représentant la congestion, à un modèle linéaire de communication point à point. Les auteurs de [13] ont repris le modèle LogGP en y rajoutant des paramètres modélisant la congestion. Le nouveau modèle obtenu a été nommé modèle loGPC, C pour *Contention*. Ce modèle associe un modèle de congestion simple dérivé de la théorie des files d'attente avec le modèle LogGP. En outre, ce modèle a été développé dans l'optique des réseaux performants.

Toutefois, sa mise en pratique est plus délicate. En effet, ce modèle introduit énormément de paramètres supplémentaires au modèle LogGP. Ces paramètres représentent certaines caractéristiques du réseau, comme par exemple, le taux d'injection de message incluant les délais dûs à la congestion ou encore la distance moyenne traversée par un message. Bien que les auteurs proposent des équations affines suivant ces paramètres, leur modèle se limite à une applicabilité réduite. La détermination des valeurs des paramètres est le principal problème, d'autant plus qu'elle se limite en pratique à certaines topologies réseaux.

Un autre modèle introduit dans [80] traite de la concurrence entre communication sur le réseau *Myrinet*. Les auteurs proposent un modèle qui multiplie une équation linéaire proche du modèle LogGP par le nombre de communications en concurrences sur le chemin réseau. La validation du modèle est réalisée en comparant des valeurs mesurées et prédites. Les taux d'erreurs obtenus sont

encourageant, dans le pire des cas ils avoisinent 30%. Toutefois, les expériences sont réalisées sur des nœuds mono-processeur. On peut se demander son efficacité sur des nœuds multi-processeurs (SMP).

D'autres approches similaires utilisant des modèles linéaires ont été introduites pour la prédiction des temps de communication collective [52].

3.4.2 Modélisation probabiliste

Les processus stochastiques peuvent être utilisés comme support pour la modélisation de la congestion réseau. En particulier, les approches Markoviennes et plus précisément les files d'attente [54] ont fait l'objet de beaucoup de recherches.

Les files d'attente déterminent des valeurs moyennes sur certain critères d'un système, tels que le temps d'attente ou le nombre de clients dans la file. Elles peuvent être agrégées entre elles, et dans certains cas, une résolution directe est possible. Dans le domaine des communications réseaux, les files d'attente sont principalement utilisées dans la commutation de paquet et pour des réseaux à forte granularité telle que les *Wide Area Network* (WAN) ou Internet. Par conséquent, elles sont généralement appliquées à la modélisation de TCP[44].

En effet, les files d'attente ont besoin de connaître les fonctions de distribution des inter-arrivées des paquets, ainsi que les taux de service (fréquemment équivalent à la bande passante). Ces distributions représentent le trafic de manière générale sur le réseau. Elles sont liées à certaines contraintes mathématiques pour être utilisables par les processus stochastiques sous-jacent aux files d'attente. La distribution par *self-similarity*, découverte récemment, semble être la plus apte à représenter le trafic sur des réseaux haute performance [51]. Cette distribution est une distribution dont les variables stochastiques sont fortement couplées [102]. En d'autres termes, cette distribution modélise aussi bien un trafic de quelques millisecondes que de plusieurs minutes. Nous recommandons aux lecteurs intéressés l'article [4].

Finalement, les files d'attente modélisent les performances globales d'un système et prédisent des valeurs statistiques sur l'état d'un système. Elles sont souvent utilisées comme outil pour déterminer des critères de qualité de service des réseaux. Elles modélisent les réseaux et la congestion réseau d'un point de vue macroscopique.

Dans un contexte de calcul haute performance, les files d'attente sont peu utilisées pour modéliser ces réseaux dédiés. Il existe plusieurs raisons à cela. D'une part, les réseaux haute performance, mise à part *Gigabit Ethernet*, n'utilisent pas de systèmes par commutation de paquets (*wormhole routing* ou *cut-throught routing*). Ils affectent des chemins à travers le réseau aux communications. D'autre part, il est commun d'exécuter une seule application à la fois sur une grappe. Or l'extraction à partir de l'application de la distribution du trafic et de son paramétrage est difficile.

3.4.3 Partage de ressource et équité

Un aspect plus théorique de la congestion peut-être abordé. La congestion réseau peut-être considérée comme une forme de partage de la bande passante entre communications [55]. D'un point de vue théorique, ce partage de ressource entraîne la notion d'équité.

L'étude de l'équité n'entraîne pas directement une modélisation ayant pour objectif la prédiction de temps de communications. L'intérêt de ces modèles de partage de ressources est d'appréhender certains systèmes face à différentes politiques d'équité [87]. Pour les réseaux, les modèles de communications appliqués au principe d'équité sont très fréquemment des modèles basés sur la théorie des fluides. Une communication est envisagée comme un liquide passant à travers des tuyaux et la bande passante correspond au débit du fluide.

Cette modélisation est utilisée dans le contexte de grille de calcul. Plus particulièrement, elle est intégrée aux simulateurs de grille comme modèle réseau [34].

Comme pour le cas des files d'attente, il s'agit d'une modélisation au niveau macroscopique.

3.4.4 Modèle de garantie pour des flux réseaux

Une nouvelle analyse des performances réseaux a été proposée dans [20][49]. Le *Network Calculus* est un outil qui permet d'obtenir des garanties de performances déterministes [48]. Cet outil est basé sur la notion de flux d'entrée et de sortie. Il permet la caractérisation d'un flux de sortie en fonction d'une courbe d'entrée (flux d'entrée) et d'une courbe de service garantie, comme le montre la figure 3.2. Il fait appel à l'outil algébrique "max-plus", qui implique une analyse des performances des cas les moins favorables.

Le *Network Calculus* est une approche intéressante car il propose une analyse déterministe de systèmes basés sur des files d'attente en utilisant une algèbre linéaire. Bien que le *Network Calculus* n'a pas été directement adapté à la problématique des réseaux haute performance tels que *Myrinet* ou *Quadrics*, son utilité dans ce domaine paraît probable s'agissant du calcul des délais maximaux des communications.

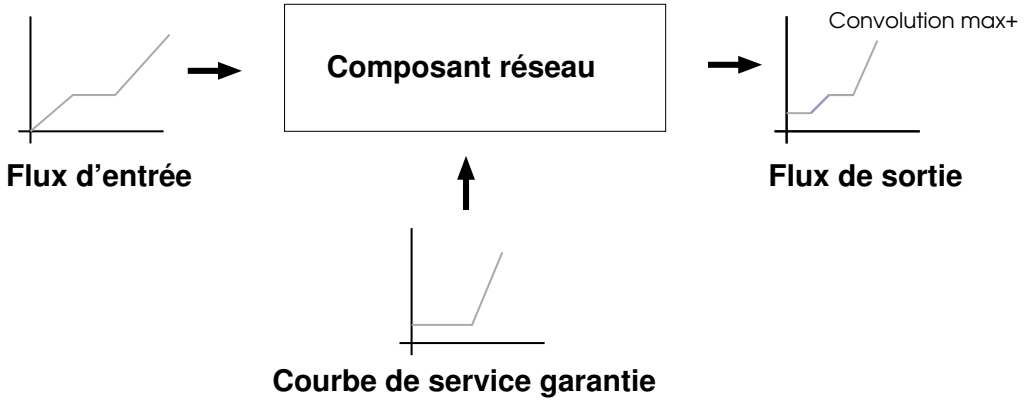


FIG. 3.2 – Modélisation par garantie pour des flux réseaux

3.4.5 Simulation

La simulation est une autre technique pour étudier et modéliser la congestion réseau. Elle consiste à reproduire les mécanismes réseaux mis en jeu mais de manière simplifiée. Cette simplification se base principalement sur des modèles du comportement réseau. Ces simulations sont fréquemment basées sur l'enchaînement d'événements, on parle alors de simulation à événements

discrets [53]. Par exemple, une simulation peut intégrer des modèles basés sur des files d'attente ou provenant de la théorie des fluides.

Le *Network Simulator* (*NS*) [92] est un simulateur à événements discrets largement répandu dans le monde de la recherche. Il permet de simuler des réseaux utilisant TCP et plus particulièrement des réseaux de grande taille. A titre d'exemple, NS simplifie TCP en ne simulant pas l'établissement des connections. Son application permet de comparer et valider différentes stratégies et/ou modèles de communication. Bien que conçu pour simuler des réseaux de grande taille, il peut-être utilisé pour simuler des réseaux dédiés.

Une simulation du réseau *Myrinet* a également été décrite dans [2]. Cette simulation à événements discrets décrit finement les différents mécanismes mis en œuvre par *Myrinet*. De plus, les auteurs proposent des modèles analytiques, mais utilisant des notions probabilistes sur certains critères tels que le taux d'occupation d'un commutateur. L'évaluation de leurs modèles se base principalement sur une comparaison entre la simulation et le modèle analytique. Les auteurs proposent également une comparaison entre une simulation et un programme simple de multiplication de matrice. Bien que les résultats obtenus soient intéressants, on peut être déçu par la méthode d'évaluation qui ne caractérise pas suffisamment les performances de cette approche. Cependant, il s'agit d'une approche intéressante et prometteuse.

Le réseau *Quadrics* a aussi fait l'objet d'une simulation dans [50]. Cette simulation à événements discrets reprend de manière détaillée les mécanismes du réseau *Quadrics*. Elle propose une évaluation sur une application réelle et prédit avec une bonne qualité le temps de l'application.

La simulation à événements discrets est un outil efficace dans la recherche de modèles pour les réseaux haute performance. Toutefois, sa mise en pratique est difficile car elle nécessite une connaissance fine des mécanismes du réseau sous-jacent.

3.4.6 Conclusion

Pour résumer, nous dirons que les modèles linéaires augmentés et la simulation sont deux outils qui permettent de prédire avec efficacité les temps de communications. Dans le cas de réseaux dédiés, ces deux techniques sont avantageuses :

- les modèles analytiques sont rapides à calculer, lorsque le nombre de paramètres utilisés est raisonnable, et ils sont précis grâce à la fiabilité de tels réseaux ;
- une simulation à événements discrets (suite d'événements déterministes) permet de comparer des prédictions obtenues à partir de trafic réseau provenant d'applications. Ceci est un avantage certain en comparaison avec le trafic venant de distribution probabiliste (comme par exemple celles utilisées dans les files d'attente).

Néanmoins, tous deux sont limités :

- les modèles linéaires ont une représentation trop large pour appréhender en détail la congestion réseau ;
- les simulations à événements discrets, à l'inverse, mettent en œuvre des mécanismes complexes et parfois coûteux pour obtenir des prédictions.

L'application des *Network Calculus* dans le domaine des réseaux haute performance reste une thématique ouverte pour la recherche de délais maximaux.

3.5 Discussion

Une approche couplant ces différentes techniques peut-être envisageable dans la recherche de modèles prédictifs de congestion.

Les modèles linéaires sont les modèles les plus efficaces pour déterminer le temps de communication point à point. Néanmoins, pour représenter les phénomènes de congestion, ils peuvent être fusionnés avec des modèles de répartition de bande passante. Ces modèles de répartition proviennent soit du principe d'équité (max/min, proportionelle, etc) ou d'une analyse fine des comportements réseaux. Pour pouvoir obtenir des prédictions d'applications s'exécutant sur une grappe, il est aussi possible de coupler ces deux techniques avec une simulation à événements discrets. Cependant, au lieu de simuler en détail les mécanismes du réseau, une telle simulation utilisera le couplage entre le modèle linéaire et le modèle de répartition de la bande passante pour calculer les temps de communications.

Dans le chapitre 5 nous utiliserons la combinaison de ces trois techniques pour proposer un modèle général. En outre, nous nous baserons sur un modèle de répartition de la bande passante obtenu par une analyse détaillée des communications concurrentes. Cette analyse est présentée dans le chapitre 4.

II

Modélisation des communications concurrentes

Lors de l'exécution d'une ou plusieurs applications sur une grappe de calcul, des accès simultanés sont créés sur les ressources des nœuds et du réseau. Plus particulièrement, une application comprenant plusieurs processus génère des communications concurrentes soit entrantes, soit sortantes d'un nœud. De la même manière, une application crée des accès concurrents en écriture ou lecture sur la mémoire. Par exemple, pour obtenir de bonnes performances, il conviendrait de ne pas regrouper sur un nœud, qui ne contient qu'une seule interface réseau, des processus provoquant beaucoup d'accès réseau. Une autre solution pour obtenir de bonnes performances serait de proposer une grappe avec un réseau plus performant ou des nœuds avec plusieurs cartes réseaux. Ainsi, déterminer le comportement de processus en concurrence en termes de coûts permet d'appréhender les performances réseaux d'une application sur une grappe de calcul.

Cette seconde partie est dédiée à l'étude du comportement réseau de plusieurs processus en concurrence. Nous examinerons dans un premier chapitre ce comportement grâce à un ensemble d'expériences. Ces expériences ont pour objectif de faire apparaître une hiérarchie dans les coûts ou pénalités sur les communications créées par des accès concurrents. En se basant sur ces observations, nous déterminerons dans un second chapitre des modèles permettant de prédire ces pénalités en fonction de la grappe de calcul sous-jacente. Ces modèles décrivent le comportement de processus accédant simultanément à la ressource réseau. Ils sont séparés en deux parties : une partie modélisant les comportements induit par la concurrence réseau, et une partie reliant cette modélisation aux besoins des applications.

Expériences et analyse des communications concurrentes 4

4.1 Introduction

Une application scientifique parallèle, s'exécutant sur une grappe de calcul, effectue des phases de calcul et de communication. Ces phases sont souvent bien distinctes dans le sens ou un processus transite d'une phase de calcul vers une phase de communication et vice versa. Les phases de communications peuvent bloquer le processus dans le cas d'une communication synchrone ou bien être non-bloquante dans le cas d'une communication asynchrone. Ainsi, chaque tâche de l'application oscille entre des phases de calcul et des phases de communications bloquantes ou non bloquantes.

Plusieurs tâches s'exécutant sur un nœud multiprocesseur ont la possibilité lors de phases de communication simultanées d'utiliser la ressource réseau. A titre d'exemple, une tâche peut envoyer des données pendant qu'une autre tâche sur le même nœud est en phase de réception. Il est alors intéressant d'étudier comment est partagée la ressource réseau entre les tâches. Plus particulièrement, quel est l'impact du partage des ressources, par exemple au niveau de la carte réseaux ou du commutateur, sur les performances des communications, et, plus généralement, sur les applications ?

Une première approche pour analyser le coût de la concurrence est l'observation du comportement des communications lors d'expérimentations réelles. Une telle approche nécessite l'étude d'indices de performance. Le temps ajouté à chaque communication par l'effet du partage de ressource semble être un indice de performance significatif et facile à obtenir. En effet, l'étude de l'utilisation des ressources réseaux matérielles ou le traçage logiciel des mécanismes de communication sont difficiles et impliquent un surcoût important sur la mesure obtenue, bien que ces techniques peuvent-être utilisées en complément.

Le partage de ressource peut prendre différentes formes suivant les paramètres expérimentaux choisis. Il devient intéressant de faire varier le contexte expérimental afin d'analyser l'évolution du coût induit par la concurrence. Par exemple, nous pouvons faire varier le nombre de communications en concurrence, les différents nœuds sources ou nœuds destinations, la date de départ ou bien encore la taille des messages. Ces diverses expérimentations engendreront ou n'engendreront pas de différence sur les temps de communication et sur le coût causé par la concurrence.

Cependant, préalablement aux expérimentations, il convient de décrire le mécanisme de com-

munication élémentaire et les potentiels points de partage lors de communications en concurrence. Cette présentation est réalisée dans la première section 4.2 de ce chapitre. La seconde section 4.3 décrit les expérimentations réalisées et les résultats qui y sont associés. Enfin, la dernière section 4.4 résume les propriétés observées sur le comportement de la concurrence entre communication.

4.2 Communications concurrentes

La première partie de cette section présente les différents mécanismes mis en œuvre lors d'une communication point à point utilisant *MPI*. Cette présentation a comme objectif d'introduire les différents composants à la fois logiciels et matériels qui interviennent lors de communications concurrentes.

4.2.1 Communications point à point

Une communication point à point correspond au sens large à un transfert de données entre deux entités. Dans un modèle de programmation par passage de messages, une communication fait intervenir deux tâches d'une même application. La première tâche, dite tâche émettrice, initie la communication et contient les données à transférer. La seconde tâche, dite tâche réceptrice, doit être à l'écoute des demandes de transfert de données et, le cas échéant, en préparer la réception. Une fois la communication initiée à la source et préparée en réception, le transfert de données peut s'effectuer sur le support physique.

Communication point à point dans *MPI*

En se rapportant au cas d'applications parallèles utilisant l'interface *MPI*, une communication point à point est engendrée par deux tâches *MPI*. Une requête est émise par la tâche émettrice vers la tâche réceptrice, puis les données sont transférées. Cette requête permet d'initier la communication. Cependant, *MPI* ne précise pas en détail le comportement standard entre ces deux tâches pour effectuer une communication. Une interprétation largement retenue, retrouvée dans plusieurs implantations *MPI*, est de considérer les tâches de réception et d'émission indépendantes. Cette indépendance est considérée comme vrai tant que l'espace des tampons est suffisant pour réaliser la communication. Dans le cas où cet espace devient insuffisant, un protocole de synchronisation entre les tâches est nécessaire. Ce protocole est un protocole de type rendez-vous. Il consiste à envoyer un premier message de petite taille pour préparer la tâche réceptrice et lui permettre d'allouer la mémoire indispensable. De manière opposée, si la taille de données est suffisamment petite, elles sont envoyées en même temps que la requête initiant la demande de communication. Par conséquent, la taille des données à envoyer détermine l'utilisation du protocole de rendez-vous. Néanmoins, la valeur de la taille, provoquant le changement de protocole, n'est pas définie par le standard *MPI*, elle dépend des choix de l'implantation *MPI*. Ces implantations suivent les particularités propres des différents réseaux.

MPI propose différentes méthodes d'envoi des données. L'appel d'une méthode d'envoi provoque le transfert des données vers le support physique en exécutant les méthodes spécifiques aux librairies bas niveau du réseau sous-jacent. Cependant, suivant les méthodes d'envoi et de récep-

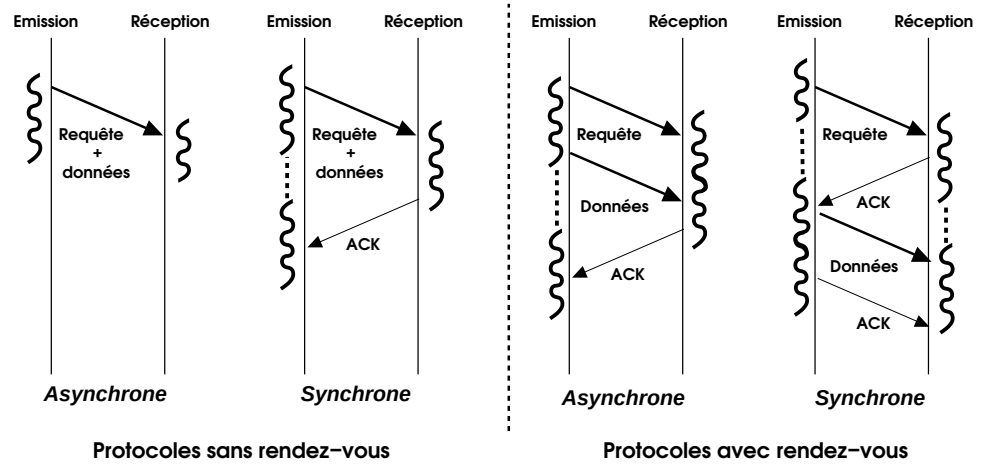


FIG. 4.1 – Protocoles de communication utilisés par *MPI*

tion utilisées (bloquantes ou non-bloquantes) le transfert des données sera immédiat ou retardé. En effet, un envoi non-bloquant (*MPI_Isend*) ne transfère pas immédiatement les données sur le support physique. Il redonne la main à la tâche *MPI* avant même la terminaison de la communication. Ces méthodes asynchrones permettent un recouvrement des communications par le calcul, en particulier, si l'implantation *MPI* contient un ou plusieurs threads dédiés aux communications. Toutefois, la tâche exécutant une méthode non-bloquante devra faire appel à une méthode d'attente (*MPI_Wait*) pour s'assurer de la terminaison de la communication.

A l'inverse, un envoi bloquant (*MPI_Send*) initie la communication au moment de l'appel de la méthode d'envoi. Il rend la main à la tâche uniquement après que la communication soit terminée. Plus exactement, dans le cas bloquant, la méthode d'envoi rend la main à la tâche à partir du moment où le tampon contenant les données d'envoi peut-être réutilisé. En outre, certaines implantations (comme *Mpich*) utilisent plusieurs méthodes d'envoi suivant la taille des messages à envoyer. Par exemple, pour des messages de petites tailles, un *MPI_Send* rend la main après une copie du tampon mémoire et non à la fin de la communication. Pour de grands messages, la même opération *MPI_Send* utilise un mode synchrone qui attend un message confirmant que les données peuvent être réceptionnées. Elle se termine lorsque toutes les données ont été transférées vers la librairie bas-niveau.

La figure 4.1 décrit différentes variantes du protocole de communication *MPI* avec et sans rendez-vous.

4.2.1.1 Communications intra-nœud

Une communication intra-nœud fait intervenir un seul nœud qui est à la fois la source et la destination des données communiquées. Ainsi, une communication intra-nœud copie des données d'une adresse mémoire vers une autre adresse mémoire du nœud. Un mécanisme traitant les communications intra-nœud comme des copies mémoires est fréquemment implanté dans la plupart des implantations *MPI*. D'une part, il devient évident, grâce à ce mécanisme à mémoire partagée, qu'une communication intra-nœud ne transfère aucune donnée sur le réseau physique. D'autre

part, la descente des données vers la carte réseau et leurs remontées vers la mémoire du nœud sont inutiles.

4.2.1.2 Communications inter-nœud

Une communication inter-nœud utilise la totalité des composants réseaux reliant deux nœuds. Nous appellerons chemin réseau la suite de composants traversés par les données lors d'une communication inter-nœud. Le chemin réseau d'une communication inter-nœud est constitué par la mémoire et plus particulièrement le bus mémoire, le bus *PCI*, la carte réseau, les liens réseaux, le commutateur, puis de nouveau les liens réseaux suivis de la carte réseau, du bus PCI et enfin du bus mémoire. La figure 4.2 représente un chemin réseau. Nous considérons uniquement des chemins réseaux avec un seul niveau de commutateur.

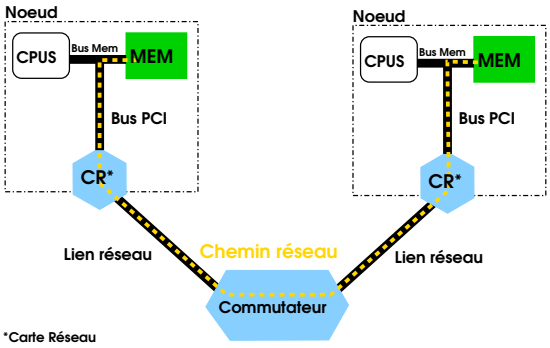


FIG. 4.2 – Chemin réseau.

Lorsque plusieurs communications intra-nœud ou inter-nœud utilisent simultanément une partie commune du chemin réseau, il se produit un partage de ressource exclusif (bus PCI) ou non-exclusif (lien full-duplex). De telles communications sont appelées des communications concurrentes.

4.2.2 Communications concurrences : identification des points chauds

La concurrence entre communication intervient à deux niveaux.

Premièrement, dans des nœuds multiprocesseur, plusieurs tâches *MPI* accèdent aux bus reliant la mémoire à la carte réseau et à la carte réseau elle-même. Au niveau de la carte réseau, le composant partagé le plus pénalisant est le tampon mémoire (*buffer*) physiquement implanté sur la carte. Par exemple, lors de concurrence entre plusieurs communications émettant depuis un même nœud, la taille du tampon mémoire est généralement divisée proportionnellement entre les messages émis. La plupart des cartes réseaux comporte un tampon mémoire pour l'émission de données et un tampon mémoire pour la réception des données.

Deuxièmement, la concurrence produite par plusieurs communications s'effectue sur les composants réseaux externes au nœud. Les liens réseaux constituent le premier élément concerné par la concurrence. Cependant, l'accès au lien est régi soit par la carte réseau (communications sortantes) soit par le commutateur (communications entrantes). Le commutateur devient donc l'élément principal pour la gestion de la concurrence à l'extérieur des nœuds. Toutefois, nous avons vu dans le

chapitre 2 que les commutateurs de réseaux haute performance, hors mis dans le cas de *Gigabit Ethernet*, ont des fonctionnalités très réduites. Ainsi, le routage, le contrôle de flux, sont contrôlés et traités en amont du commutateur, c'est-à-dire au niveau des cartes réseaux. A l'inverse, dans le cas du réseau *Gigabit Ethernet*, les fonctionnalités du commutateur ont une grande importance dans la gestion de la concurrence. En effet, le commutateur gère la répartition des accès aux liens réseaux.

En conclusion, dans le cas des réseaux haute performance, la carte réseau est le composant réseau principal qui gère la concurrence entre communications. Toutefois, pour le réseau *Gigabit Ethernet*, les commutateurs interviennent aussi de manière significative dans la gestion de la concurrence.

En se basant sur ces premières constatations techniques, nous proposons dans la section suivante d'étudier de manière expérimentale les effets de la concurrence entre différentes communications. Nous reviendrons dans le chapitre 6 sur l'importance des composants réseaux dans la gestion de la concurrence.

4.3 Expérimentations

Cette section a pour objectif d'étudier expérimentalement la concurrence entre communications. Les effets de la concurrence dépendent du contexte expérimental, c'est-à-dire du nombre de communications en concurrence, des caractéristiques des communications (taille de message, date de départ), du type de réseau ou encore du choix de l'implantation *MPI*. Devant ce nombre important de paramètres, nous choisissons une approche progressive dans laquelle les contextes expérimentaux augmentent progressivement en complexité (nombre de communications). De plus, nous analyserons, en parallèle, la concurrence entre communications pour trois implantations *MPI* (deux implantations libres et une implantation propriétaire) et trois réseaux.

Dans la première partie, nous établirons le protocole expérimental, en spécifiant l'objectif des expérimentations et les outils utilisés pour l'obtention des résultats de performance. Dans une seconde partie, nous commenterons les résultats d'expériences introductives. Nous préciserons la notion de conflits réseaux en décrivant différentes formes élémentaires d'expression de la concurrence entre communications. Dans cette partie, nous proposerons une simplification de la présentation des résultats expérimentaux, afin, dans une troisième partie, d'analyser plus aisément des expériences complexes. Finalement, dans une dernière partie, nous synthétiserons les différents résultats expérimentaux. A noter que nous présentons dans le cas du réseau *Gigabit Ethernet* uniquement des expériences réalisées avec l'implantation *Mpich*. Cependant, la même série d'expériences réalisées avec l'implantation *Lam/MPI* est accessible dans l'annexe B.

4.3.1 Protocole expérimental

Le protocole expérimental permet de décrire les objectifs expérimentaux et d'établir le contexte des expériences.

Objectifs expérimentaux

Le but principal des expériences réalisées est l'étude des effets de la concurrence sur les temps de communication. Il s'agit donc d'étudier la variabilité des temps de communication induite par la concurrence. Nous appelons pénalité le rapport entre le temps d'une communication soumise à aucune concurrence avec les temps de communications soumises, quant à elles, à un contexte concurrent. En outre, cette étude s'applique à toutes tailles de messages. Il s'agit donc d'étudier à la fois la latence et la bande passante. Les résultats obtenus sont présentés suivant deux supports. Le premier support, sous forme de graphe à l'échelle logarithmique et de tableaux de valeurs, décrit en détails les temps de communications de contextes concurrents simples. Dans le cas d'expériences complexes, faisant intervenir plusieurs nœuds et communications, un deuxième support de présentation, moins détaillé, est proposé. Ce support consiste à la représentation graphique de contextes concurrents accompagnés des pénalités associées à chaque communication.

Le contexte expérimental se base sur plusieurs paramètres.

Grappes de calcul

Les grappes de calcul, utilisées dans les expérimentations suivantes, comportent trois types de réseau et deux types de nœuds. Les grappes considérées sont : la grappe Rennes pour le réseau *Gigabit Ethernet*, la grappe Sophia pour le réseau *Myrinet* et enfin la grappe *Teratec* pour le réseau *Quadrics*. Les deux premières grappes sont constituées de nœuds biprocesseur Opteron. La grappe *Teratec* contient des nœuds à 16 processeurs Itanium 2, nœuds *Bull Novascale*, et un réseau *Quadrics*.

Le lecteur trouvera plus de précisions sur les grappes de calcul employées dans l'annexe A.

Implantation *MPI*

Suivant la grappe de calcul, nous utiliserons une ou plusieurs implantations *MPI*. Les implantations utilisées sur le réseau *Gigabit Ethernet*, c'est-à-dire la grappe Rennes, sont *Mpich* 1.2.6 et *Lam/MPI* 7.1.1. L'implantation employée sur la grappe Sophia, comportant un réseau *Myrinet*, est une implantation *Mpich* 1.2.6 associée à une version 1.1.3 du protocole *MX*. Cette implantation *MPI* est fournie par la société *Myricom*. Finalement, une implantation industrielle fournie par *Bull*, sera utilisée sur la grappe *Teratec*. La version de cette implantation est *MPI Bull* 1.6.4.

Programme test

Afin de générer des communications concurrentes, nous utiliserons le même programme *MPI* sur chacune des grappes. L'objectif de ce programme est d'exprimer la variation d'un indice de performance : le temps de chaque communication en concurrence. Ce programme consiste en un envoi simple de données utilisant les méthodes bloquantes *MPI_Send* et *MPI_Recv*. Il permet l'exécution répétée de plusieurs schémas de communications concurrentes. Il est possible de paramétrer, le nombre de communications, la taille des données de chaque communication, leurs dates de départ, ainsi que le nombre de tests à répéter. Les résultats obtenus sont des indices statistiques, comme par exemple la moyenne ou la médiane, sur un échantillon de 1000 temps de communication. Afin d'obtenir des mesures précises, les différents délais ou mesures de temps sont réalisés en

accédant aux compteurs matériels des processeurs. La description du code de ce programme test est détaillée dans la figure 4.3. Notons que, suivant l'implantation *MPI*, lors de message de petite taille, la mesure proposée n'est pas la latence réseau. En effet, pour des messages de petites tailles, ce temps correspond à la copie du message dans un tampon mémoire défini par l'implantation *MPI*.

Placement des tâches *MPI*

Chaque expérience comporte plusieurs tâches *MPI*, ce nombre est égal au double du nombre de communications de l'expérience. En effet, nous établirons un état unique pour chaque tâche, c'est-à-dire qu'une tâche peut soit recevoir soit émettre des données sans avoir la possibilité de changer d'état au cours du temps. De plus, chaque processeur exécute une seule tâche *MPI* qui sera placée sur ce processeur en fixant l'affinité de l'ordonnanceur système.

Le placement des tâches sur les nœuds est un paramètre variable, ce placement sera évalué au niveau de l'implantation *MPI* (à l'exception de l'implantation *MPI Bull* qui ne permet pas un placement fin, dans ce cas, le placement sera un paramètre supplémentaire du programme test). Le numéro du processeur sur laquelle une tâche sera figée est lui un paramètre du programme de tests. En outre, le nœud à partir duquel est lancé l'expérimentation ne fait pas partie des nœuds impliqués dans les communications.

Résumé des paramètres expérimentaux variables

Pour résumer, lors des expériences, les paramètres variables sont :
- L'implantation *MPI* et la grappe de calcul ;
- Le placement des tâches sur les nœuds de la grappe ;
- Les paramètres des communications concurrentes : nombres, taille, date de départ.

Les résultats obtenus expriment le temps additionnel causé par la concurrence des communications.

4.3.2 Expérimentations introductives

Nous choisissons de suivre une approche incrémentale dans la suite des expériences. De ce fait, les premières expériences sont réalisées avec simplement deux communications en concurrence. Chacune de ces communications a la même quantité de donnée à transférer et débute au même instant. Nous appellerons conflit un ensemble de communications en concurrence. La complexité d'un conflit dépend du nombre de nœuds et du nombre de communications en concurrence. Les premières expériences réalisées comportent un nombre limité de nœuds et de communication mises en concurrence. De telles expériences correspondent à l'étude de conflits simples, ou conflits élémentaires.

4.3.2.1 Description des conflits simples

Nous présentons quatre conflits simples, faisant intervenir deux communications. Dans la figure 4.4, les quatre conflits se décrivent comme suit :

Début du code

```
; Parameters in:
;        nb_test : integer        // nombre de tests
;        nb_pro  : integer        // nombre de tâches
;        size : array of nb_pro*integer        // taille des communications
;        delay : array of nb_pro*integer        // date de départ des communications
;
; Parameters out:
;        // résultats statistiques
;        median, average : double
;        standard deviation, absolute deviation : double
;        minimum, maximum : double
;        quartile : array of 4*double

if (process_rank % 2 ==  0) {        // tâche  paire correspondante à une émission
        repeat 5 times {        // éviter l'influence du cache
                MPI_Send (process_rank+1, size[process_rank])
        }
        MPI_Barrier
        repeat nb_test times {        // Mesures
                wait delay[process_rank]
                start timer
                MPI_Send (process_rank+1,size[process_rank])
                stop timer
                MPI_Barrier
        }
} else {         // tâche  impair correspondante à une réception
        repeat 5 times {        // éviter l'influence du cache
                MPI_Recv (process_rank-1, size[process_rank])
        }
        MPI_Barrier
        repeat nb_test times {        // Mesures
                MPI_Recv (process_rank-1,size[process_rank])
                MPI_Barrier
        }
}
```

Fin du code

FIG. 4.3 – Pseudo-code du programme de test

- Conflit simple entrant (Conflit E/E) : ce conflit fait intervenir deux communications entrantes simultanément sur un nœud. Dans ce conflit, chaque communication ne subit aucun retard provenant des nœuds émetteurs. Cependant, en arrivant simultanément au nœud récepteur, il se produit une répartition des accès aux ressources du nœud ou du commutateur.
- Conflit simple sortant (Conflit S/S) : ce conflit est opposé au conflit précédent. En effet, deux communications sortent simultanément d'un même nœud. Ainsi, dès le début des communications, elles sont en concurrence sur les ressources du nœud. Cependant, une fois arrivé au commutateur chaque message prend une route différente sans induire de nouveaux effets concurrents.
- Conflit entrant/sortant (Conflit E/S) : les deux communications de ce conflit se rencontrent au niveau du commutateur et d'un nœud. Cependant, la première communication entre dans le nœud par un brin du lien full-duplex, donc sans concurrence avec la communication sortante sur l'autre brin du lien. De plus, la carte réseau dispose d'un tampon mémoire en réception et en émission. De ces premières constatations, nous pouvons remarquer qu'un

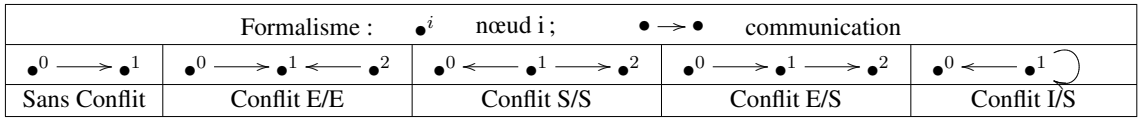


FIG. 4.4 – Conflits simples.

conflit du type E/S est probablement moins pénalisant que les autres conflits élémentaires.

– Conflit intra-nœud/extra-nœud (Conflit I/E ou Conflit I/S) : une des deux communications engendrée par ce conflit ne transfère pas de données sur le réseau. Il s'agit donc d'un conflit entre une communication intra-nœud et une communication extra-nœud. La communication extra-nœud peut soit être une communication entrante soit une communication sortante. Nous ne présentons pas dans cette section les conflits entre deux communications intra-nœud, car elles ne font pas intervenir la ressource réseau. Ce type de conflit, conflit I/I, sera présenté dans le chapitre 6.

4.3.2.2 Résultats expérimentaux

En se basant sur les quatre conflits précédents, nous présentons les résultats obtenus suivant le protocole expérimental. Les valeurs des temps présentées dans les courbes sont les médianes des temps de chaque communication pour un nombre de test égal à 1000. Nous faisons varier la taille des communications en concurrence de manière identique. Nous comparons, ainsi, les temps obtenus par rapport aux valeurs des temps d'une communication soumise à aucun conflit.

Ces résultats sont décrits dans les paragraphes suivants pour chaque réseau et leur implantation *MPI* associée.

Quadrics

Le premier réseau que nous proposons d'étudier est le réseau *Quadrics*. En effet, ce type de réseau permet d'avoir des résultats stables, ce qui en facilite l'interprétation.

Conflit E/E (Quadrics)

Dans le cas du conflit E/E, figure 4.5, nous remarquons un effet de séparation, à partir d'une taille de message seuil, entre les valeurs des temps de communication concurrente et non concurrente. L'écart entre les communications concurrentes et la communication seule est égal au temps de communication seul. En d'autres termes, le temps des communications concurrentes doublent. Nous remarquons que ce dédoublement des temps des communications se fait progressivement sans aucun effet de saut et à partir d'une petite taille seuile de message. Cette valeur du seuil est d'environ 4Ko. Si nous comparons de façon plus précise les indices statistiques[1] obtenus sur cette expérience, nous en déduisons que la variation des temps des deux communications en concurrence est très faible. Ces indices sont présentés dans le tableau 4.1.

[1]Moy : moyenne, Med : médiane, ET : écart type, Min : minimum, Max : maximum.

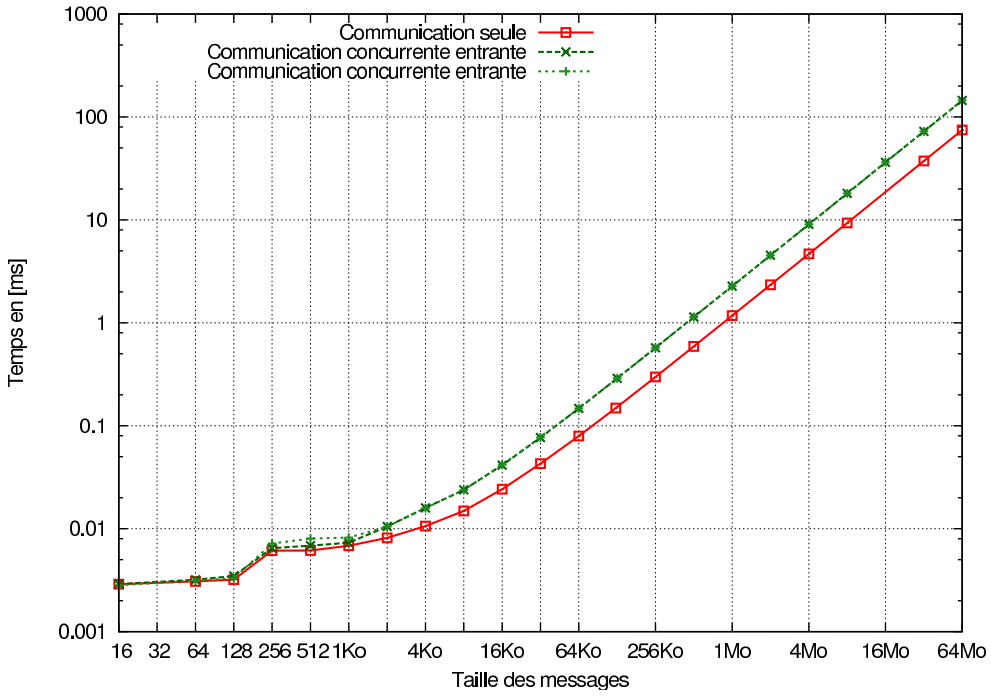


FIG. 4.5 – *Quadrics*, *MPI Bull*, Conflit E/E

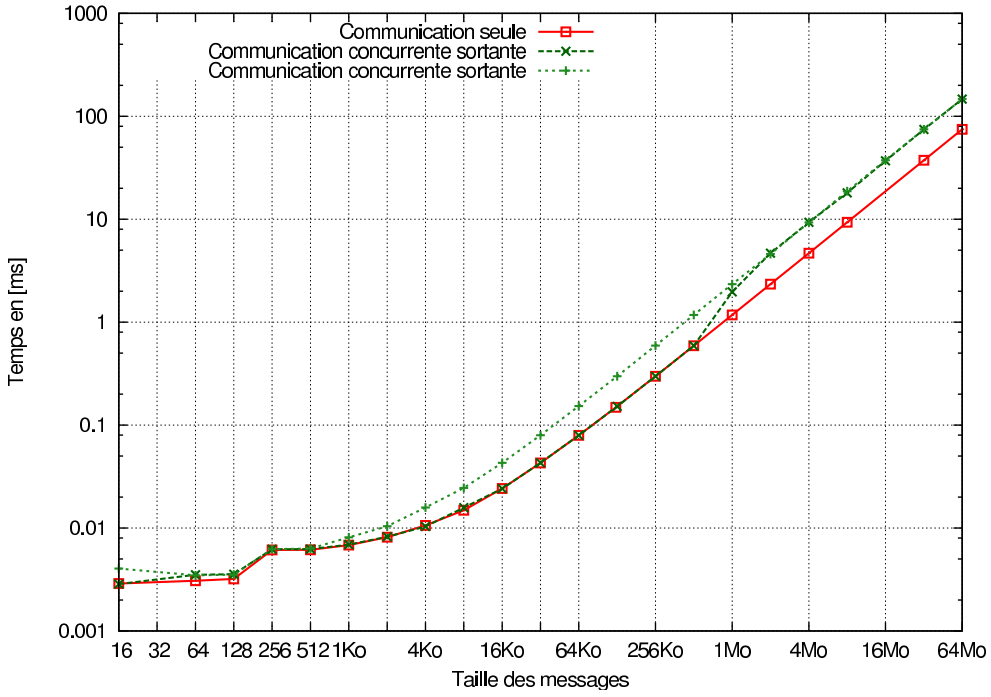


FIG. 4.6 – *Quadrics*, *MPI Bull*, Conflit S/S

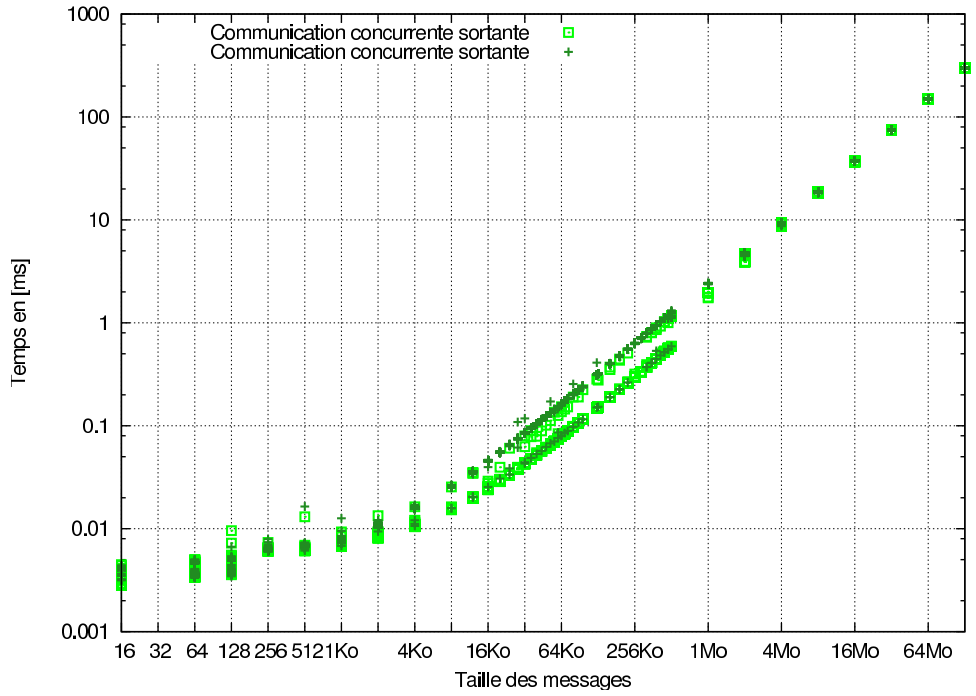


FIG. 4.7 – *Quadrics*, *MPI Bull*, détail des temps du conflit S/S

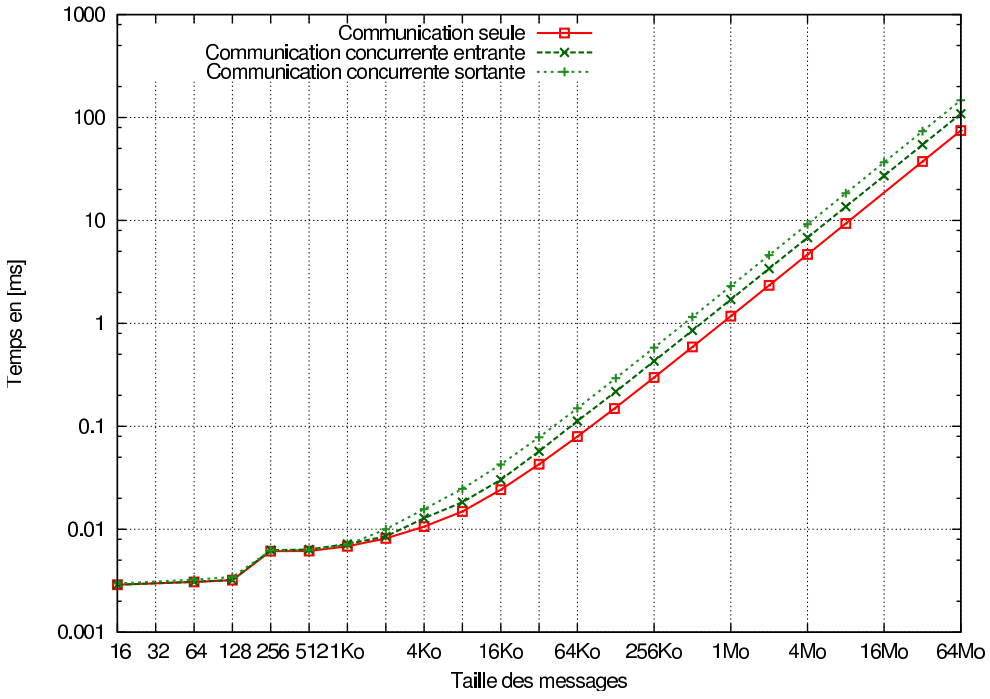


FIG. 4.8 – *Quadrics*, *MPI Bull*, Conflit E/S

Conflit S/S (Quadrics)

Pour le conflit S/S, figure 4.6, les communications concurrentes ont un comportement divergeant. En effet, à partir d'un seuil d'environ 4Ko, les temps des deux communications concurrentes diffèrent. Le temps d'une des communications ne subit aucun surcoût induit par la concurrence alors que le temps de la deuxième communication concurrente subit une pénalité similaire aux communications du conflit E/E. Pourtant, d'un point de vue expérimental, les communications sont semblables, l'expérience ne favorisant aucune des deux communications. Il en résulte, en analysant finement les indices statistiques du tableau 4.1, que chacune des communications peut jouer un rôle identique. De façon plus précise, si nous comparons les minima et maxima des deux communications, nous trouvons des valeurs identiques. De plus, en comparant la distribution des valeurs des temps de communication, nous remarquons que les valeurs observées varient uniquement autour de ces extremums (écart type très faible). La figure 4.7 décrit les temps bruts des communications observées. Cette courbe montre bien que pour des messages entre 4 Ko et 512 Ko chaque communication propose des temps variant uniquement autour de deux valeurs. Cela confirme deux points : les communications jouent un rôle identique et les temps observés correspondent à des temps sans concurrence ou au double de ce temps (comme pour des communications supérieures à 512Ko). Il se trouve qu'en observant les temps moyens de cette expérimentation, une des deux communications est plus souvent ralentie par le partage de ressource. Finalement, cette constatation peut-être assimilée à un effet "premier arrivé, premier servi" (*First Come, First Serve*) ce qui conduit la seconde communication à attendre la fin de la première. A partir, de 512 Ko, cet effet disparaît et chaque communication a une pénalité équivalente au double du temps de la communication seule. On retrouve un comportement similaire au conflit E/E.

Conflit E/S (Quadrics)

Le conflit E/S montre une plus grande hétérogénéité au niveau des surcoûts induits sur les communications concurrentes, figure 4.8. En effet, chaque communication a une pénalité différente à partir d'un seuil de 4Ko. Cependant, les valeurs des pénalités deviennent constantes pour des communications supérieures à 32Ko. Les deux communications en concurrence, dans ce conflit, jouent un rôle différent. En effet, une communication arrive pendant qu'une autre part. La pénalité est plus importante pour la communication sortante du nœud. Pour cette communication, cette pénalité équivaut à 2 fois le temps de communication seule, alors que pour la communication entrante elle vaut 1,5 fois le temps de communication seule. Néanmoins, il faut rester prudent quant aux valeurs des pénalités présentées. En effet, si une des deux communications en concurrence se termine avant l'autre, la communication restante ne sera plus en concurrence durant une certaine période de temps. Ce phénomène accélère la communication durant cette période par rapport à un conflit affectant toute la durée de la communication. Ainsi, dans ce cas précis, la pénalité mesurée comme 2 fois la valeur du temps de communication seul correspond en réalité à une pénalité impliquant 3 fois ce temps. Ces nouvelles pénalités sont obtenues en résolvant un simple système linéaire. Nous reviendrons sur cet aspect lors de la complexification des expériences, sous-section 4.3.4, en proposant une méthode pour obtenir les pénalités lors de communications ne se terminant pas au même moment.

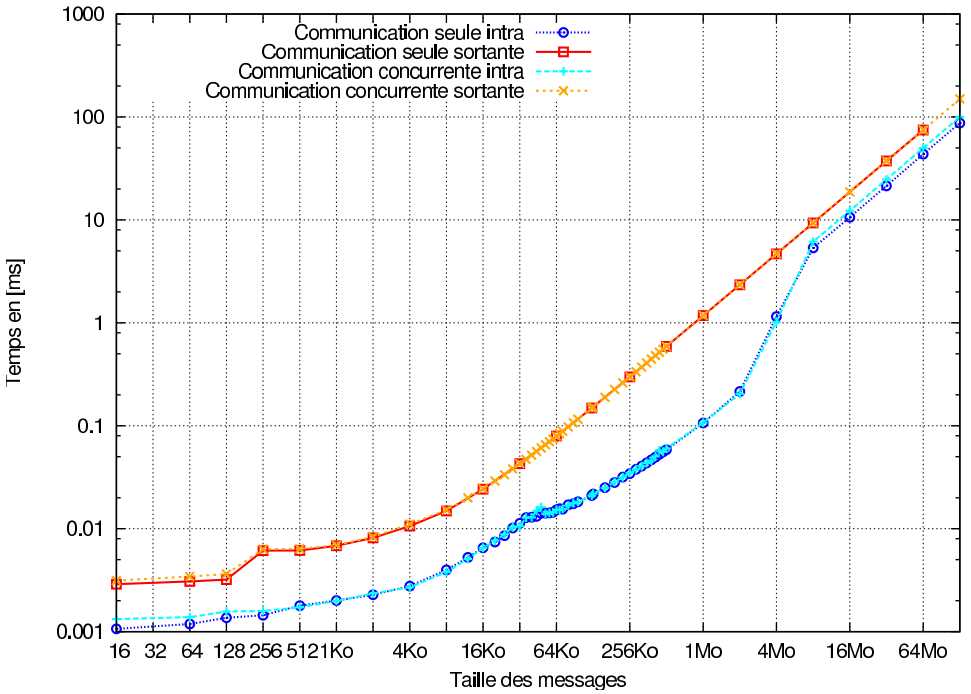


FIG. 4.9 – *Quadrics*, *MPI Bull*, Conflit I/S

Conflit I/S (Quadrics)

La figure 4.9 montre clairement qu'il n'y a qu'un très faible impact entre une communication intra-nœud et une communication inter-nœud. Nous notons uniquement un très léger ralentissement pour la communication intra-nœud et pour des messages de grandes tailles. Cet effet s'explique par le fait que les deux communications sont placées sur un même intra-nœud NUMA et partage donc une partie du bus mémoire. Les résultats sont similaires pour un conflit I/E.

En résumé, *Quadrics* montre un écart équivalent au double du temps sans communication. Le conflit E/S montre un comportement particulier en introduisant pour chaque communication en concurrence une pénalité différente. Les communications intra-nœud et inter-nœud n'ont qu'une très faible influence entre elles.

		16o [μs]					16Ko [ms]					16Mo[s]				
		Moy	Med	ET	Min	Max	Moy	Med	ET	Min	Max	Moy	Med	ET	Min	Max
Sans conflit		2.9	2.8	0.3	2.8	8.0	0.024	0.024	0	0.023	0.033	0.018	0.018	0	0.018	0.018
Conflit E/E	Com1	3.1	2.9	0.4	2.8	6.8	0.041	0.041	0.001	0.033	0.045	0.036	0.036	0	0.036	0.036
	Com2	2.9	2.8	0.4	2.7	5.0	0.041	0.042	0.002	0.033	0.046	0.036	0.036	0	0.036	0.036
Conflit S/S	Com1	3.0	2.8	0.4	2.8	4.9	0.024	0.024	0	0.024	0.043	0.037	0.037	0	0.036	0.037
	Com2	3.8	4.0	0.4	2.8	7.0	0.042	0.043	0	0.024	0.044	0.037	0.037	0	0.037	0.040
Conflit E/S	Com1	2.9	2.8	0.2	2.8	4.4	0.030	0.030	0.001	0.028	0.037	0.027	0.027	0	0.027	0.027
	Com2	3.4	2.9	0.6	2.7	7.8	0.042	0.042	0	0.041	0.046	0.036	0.036	0	0.036	0.037

TAB. 4.1 – Indices statistiques des temps de communication des conflits E/E, S/S, E/S pour le réseau *Quadrics* et *MPI Bull*.

Myrinet

Les conflits élémentaires sur le réseau *Myrinet* proposent des comportements communs entre eux.

Conflit E/E (Myrinet)

Myrinet propose des temps de communications très stables. Le conflit E/E montre deux phases distinctes, figure 4.10. Ces deux phases sont séparées par un saut très important des valeurs mesurées. Ce saut se situe vers 32Ko. La première phase, avant le saut, ne décrit aucune différence entre les communications concurrentes et non concurrentes. Durant cette phase, la concurrence n'induit aucun effet. Dans la deuxième phase, après le saut, les temps des communications concurrentes sont multipliés par 1,9. De manière approximative, nous pouvons dire que les temps doublent. Cette propriété est identique à celle rencontrée dans le réseau *Quadrics*.

Conflit S/S (Myrinet)

Le conflit S/S décrit exactement le même comportement que le conflit E/E, figure 4.11. *Myrinet* se comporte de manière similaire sur ces deux conflits.

Conflit E/S (Myrinet)

Le conflit E/S confirme, dans la figure 4.12, qu'il s'agit d'un conflit peu coûteux pour des communications concurrentes. Plus précisément, dans le cas de *Myrinet*, ce conflit ne génère aucune pénalité quel que soit la taille des communications concurrentes.

Conflit I/S (Myrinet)

En utilisant le protocole expérimental impliquant une tâche *MPI* par unité de calcul, il ne nous est pas possible de réaliser un tel conflit sur des machines ne possédant que deux unités de calcul. En effet, pour ce conflit, le nombre d'unités de calcul nécessaire est de 3 : 2 unités pour la communication intra-nœud et 1 unité pour la communication sortante.

En résumé, *Myrinet* a un comportement assez stable, proposant un saut important pour chaque conflit. Ce saut provoque un doublement des temps de communication dans le cas des conflits E/E et S/S mais n'ajoute aucune pénalité dans le cas du conflit E/S. Le tableau 4.2 résume les valeurs des courbes.

		16o [μs]					16Mo[s]				
		Moy	Med	ET	Min	Max	Moy	Med	ET	Min	Max
Sans conflit		0.5	0.49	0.1	0.49	0.97	0.071	0.071	0	0.071	0.073
Conflit E/E	Com1	0.59	0.59	0.1	0.58	1.13	0.134	0.133	0.001	0.130	0.138
	Com2	0.4	0.4	0.1	0.4	0.47	0.139	0.138	0	0.138	0.141
Conflit S/S	Com1	0.51	0.51	0	0.5	1.0	0.138	0.137	0	0.136	0.140
	Com2	0.59	0.59	0	0.5	1.0	0.138	0.138	0	0.137	0.140
Conflit E/S	Com1	0.56	0.56	0	0.56	0.82	0.071	0.071	0	0.071	0.072
	Com2	0.56	0.56	0	0.56	0.8	0.071	0.071	0	0.071	0.074

TAB. 4.2 – Indices statistiques des temps de communication des conflits E/E, S/S, E/S pour le réseau *Myrinet* et *Mpich* MX.

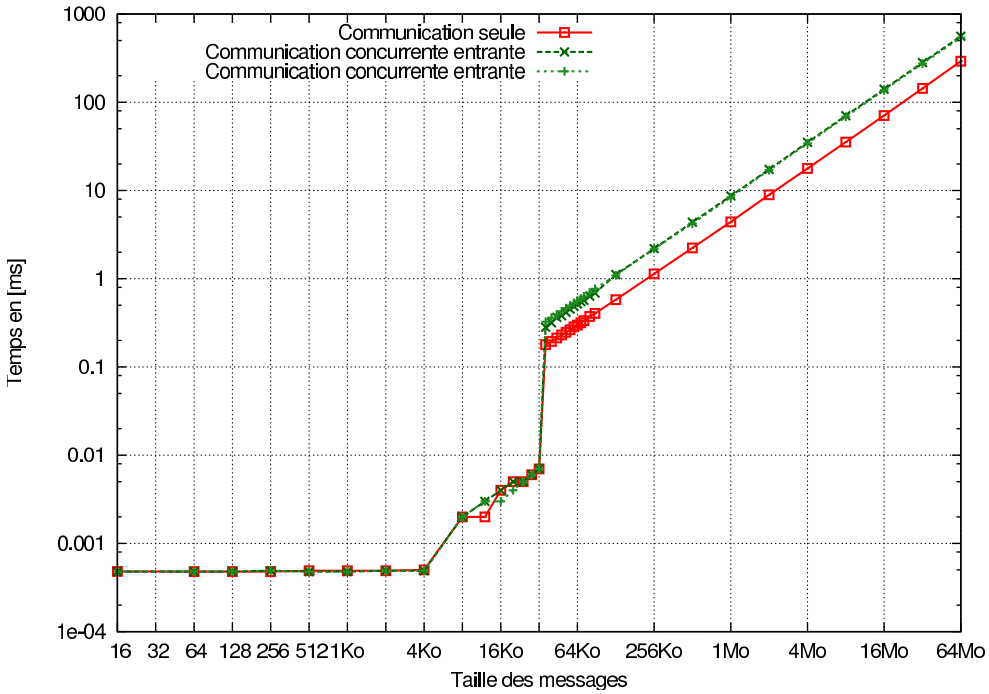


FIG. 4.10 – *Myrinet*, *Mpich* MX, Conflit E/E

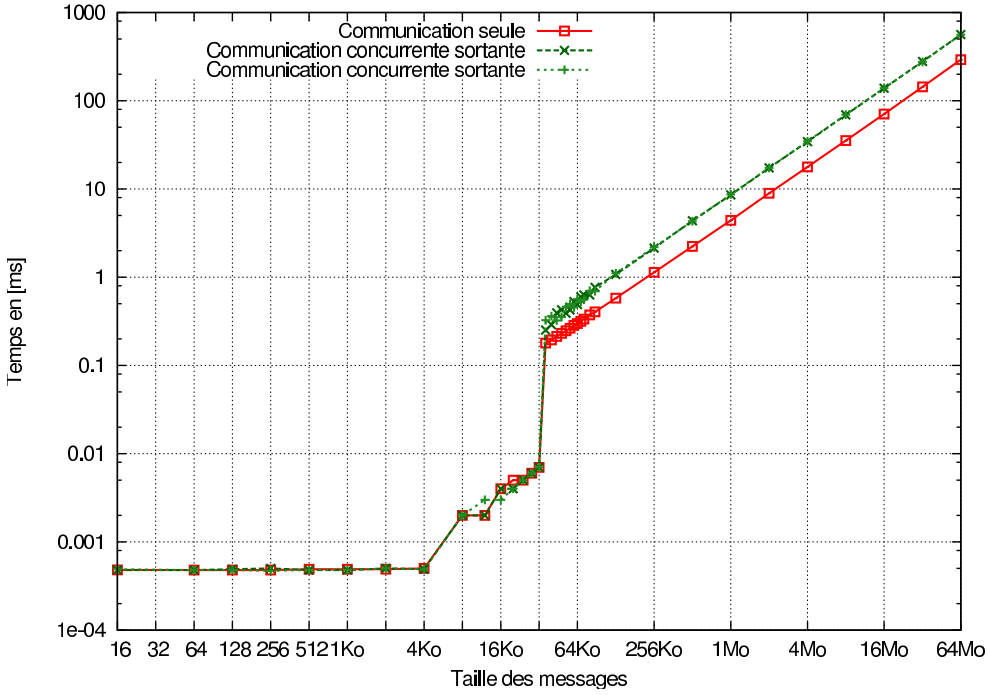


FIG. 4.11 – *Myrinet*, *Mpich* MX, Conflit S/S

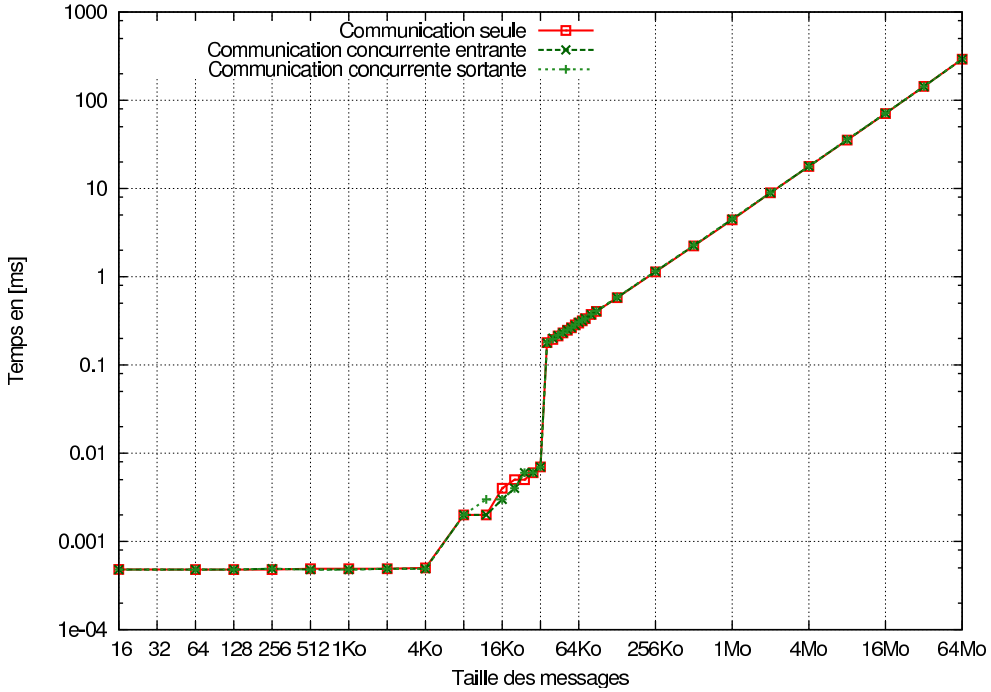


FIG. 4.12 – *Myrinet*, *Mpich* MX, Conflit E/S

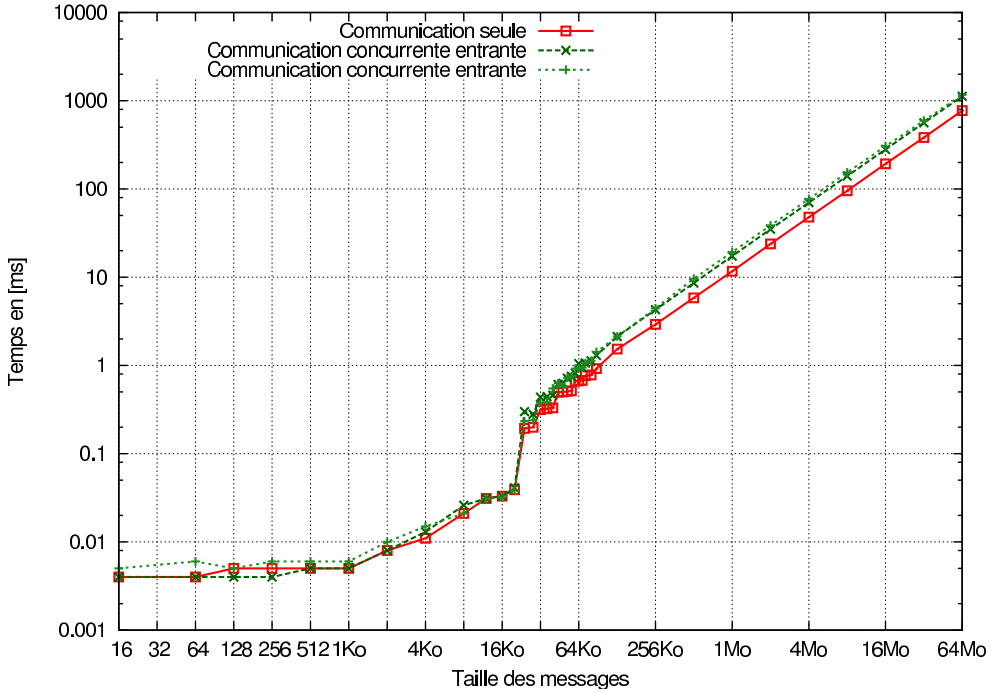


FIG. 4.13 – *Gigabit Ethernet*, *Mpich*, Conflit E/E

Gigabit Ethernet : *Mpich*

Les conflits élémentaires sur le réseau *Gigabit Ethernet* ont un comportement similaire aux conflits du réseau *Myrinet*.

Conflit E/E (Gigabit Ethernet, Mpich)

Le conflit E/E montre deux phases distinctement séparées par un saut au niveau des temps, figure 4.13. Ce saut intervient pour des messages de taille supérieure ou égale à 23Ko. Cette taille de message, comme dans le cas de *Myrinet*, correspond à la valeur seuil de changement de protocole *MPI*, pour le protocole de rendez-vous (dans l'implantation *Mpich*). Nous remarquons que ce saut est moins prononcé que dans le cas de *Myrinet*, probablement par le fait que TCP est moins sensible à l'attente d'un message supplémentaire. En amont de ce saut, nous observons une faible variabilité des valeurs observées. Ces valeurs restent proches des valeurs sans concurrence, comme le confirment les valeurs des indices statistiques[2] dans le tableau 4.3. Pour des communications supérieures à 23Ko, le temps de chaque communication concurrente devient identique valant 1,5 fois le temps sans communication (l'échelle logarithmique diminue cet effet sur la courbe).

Conflit S/S (Gigabit Ethernet, Mpich)

Le conflit S/S est très similaire au conflit E/E, figure 4.14. Nous retrouvons l'effet du saut pour une valeur identique de 23Ko. De plus, une fois les communications concurrentes supérieures à cette valeur, et comme précédemment, les temps obtenus sont multipliés par 1,5.

Conflit E/S (Gigabit Ethernet, Mpich)

Le conflit E/S décrit, à travers la figure 4.15, un comportement plus particulier que les deux conflits précédant. Même si l'effet de saut existe aussi à 23Ko, il n'implique pas un doublement du temps de communication. En effet, sur le tableau 4.3, nous observons que les temps de communication avec concurrence sont très légèrement supérieurs aux valeurs sans concurrence, pour des tailles de communication supérieure à la valeur seuil du saut. Pour des communications inférieures à cette valeur seuil, nous observons une variabilité similaire au conflit E/E. Cependant, nous remarquons que la communication sortante est très légèrement favorisée par rapport à la communication entrante.

Pour résumer, chaque conflit montre un saut à 23Ko, qui implique dans le cas des conflits E/E et S/S une multiplication par 1,5 des temps de communications concurrentes. Cette augmentation des temps représente nettement la pénalité induite par la concurrence. Dans le cas du conflit E/S, l'augmentation ne se produit pas. En outre, les temps des communications avant ce saut, montrent une variabilité autour du temps sans concurrence, ou, dans le cas du conflit S/S, du double de cette valeur.

[2]Moy : moyenne, Med : médiane, ET : écart type, Min : minimum, Max : maximum.

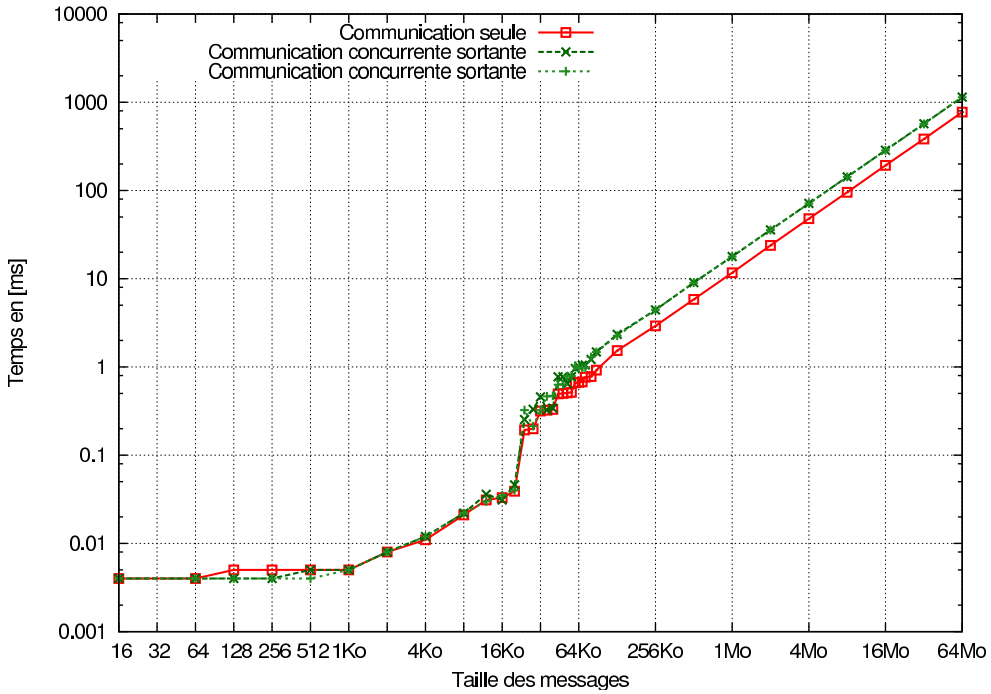


FIG. 4.14 – *Gigabit Ethernet*, *Mpich*, Conflit S/S

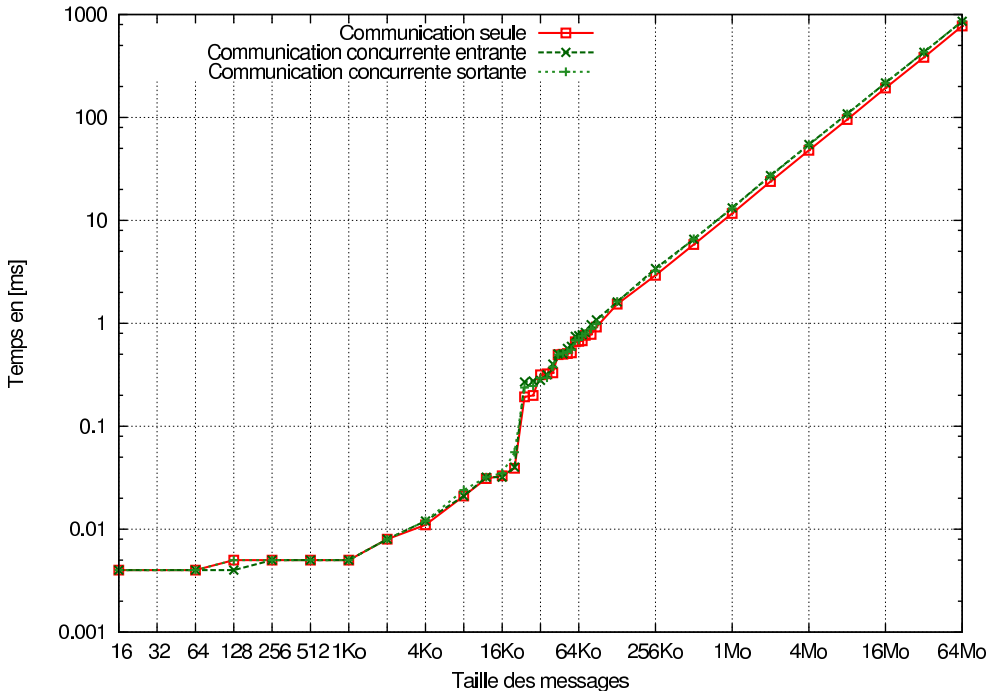


FIG. 4.15 – *Gigabit Ethernet*, *Mpich*, Conflit E/S

Conflit I/S (Gigabit Ethernet)

De la même manière que pour le réseau *Myrinet*, il ne nous est pas possible de réaliser un tel conflit sur des machines possédant uniquement deux unités de calculs.

		16o [μs]					16Ko [ms]					16Mo[s]				
		Moy	Med	ET	Min	Max	Moy	Med	ET	Min	Max	Moy	Med	ET	Min	Max
Sans conflit		4.2	4.3	0.1	3.6	4.7	0.033	0.032	0.004	0.030	0.071	0.194	0.194	0	0.192	0.197
Conflit E/E	Com1	4.4	4.4	0.2	3.8	5.1	0.035	0.035	0	0.033	0.038	0.285	0.285	0	0.284	0.286
	Com2	4.4	4.4	0.6	3.5	7.5	0.034	0.034	0.001	0.032	0.043	0.284	0.285	0	0.279	0.286
Conflit S/S	Com1	4.7	4.6	0.2	4.3	5.0	0.033	0.033	0.001	0.031	0.042	0.290	0.284	0.009	0.279	0.307
	Com2	4.2	4.2	0.2	3.7	6.2	0.033	0.033	0.001	0.032	0.041	0.291	0.285	0.009	0.280	0.307
Conflit E/S	Com1	4.3	4.3	0.2	3.6	4.8	0.033	0.033	0.001	0.032	0.043	0.218	0.202	0.005	0.201	0.229
	Com2	5.0	4.8	0.8	3.9	10.5	0.037	0.034	0.007	0.033	0.073	0.207	0.196	0.003	0.201	0.231

TAB. 4.3 – Indices statistiques des temps de communication des conflits E/E, S/S, E/S pour le réseau *Gigabit Ethernet* et *Mpich*.

4.3.2.3 Bilan des expériences introductives

Du point de vue des réseaux, les expériences confirment que les réseaux *Quadrics* et *Myrinet* ont des temps de communications plus stables que *Gigabit Ethernet*. Toutefois, *Myrinet* et *Gigabit Ethernet* ont un comportement très similaire pour les trois conflits. En effet, une augmentation des temps des communications concurrentes se produit à partir d'une valeur seuil correspondant au changement du protocole *MPI* pour un protocole avec rendez-vous. Dans le cas de *Myrinet*, ce seuil vaut 32Ko et la pénalité correspond à une multiplication par 2, alors que dans le cas de *Gigabit Ethernet*, le seuil vaut 23Ko et la pénalité est égale à 1,5. *Quadrics* a un comportement plus spécifique, sans effet de saut. Pour les conflits E/E et S/S, *Quadrics* se comporte de manière similaire à *Myrinet*, en doublant les temps de communications concurrentes, mais sans effet de saut significatif. Dans le cas du conflit E/S, une pénalité différente est ajoutée à chaque communication concurrente, alors que pour les autres réseaux la pénalité est quasiment nulle.

4.3.3 Simplification des résultats

Les résultats présentés dans la section précédente démontrent certains effets communs pour chaque conflit et type de réseau. En particulier, à partir d'un seuil sur la taille des messages, une pénalité induite par la concurrence se manifeste. Cette pénalité peut-être évalué par un nombre réel multiple du temps de communication sans concurrence. Même si la valeur du seuil est différente pour chaque réseau, nous choisissons de présenter de façon similaire les résultats d'expériences plus complexes. Nous associons à chaque communication une valeur réelle représentant la pénalité due à la concurrence. De plus, cette représentation sera simplifiée en présentant les schémas de communication de façon graphique. Par exemple pour le cas de *Quadrics*, si nous reprenons les conflits élémentaires, nous obtenons les graphiques du tableau 4.16. Sur ces graphiques, les valeurs entre paranthèses représentent les valeurs des pénalités. Pour des communications de taille inférieure à la valeur seuil, nous ne présenterons pas les valeurs mesurées. En effet, ces valeurs ont une faible variation quel que soit le nombre de communications ou conflits mis en jeu. Il en est de même pour les valeurs seuils.

Formalisme : $\bullet^i$ nœud i ; $\bullet \xrightarrow{(p)/n\rightarrow} \bullet$ n communications, p pénalité/communication			
$\bullet^0 \xrightarrow{(1)} \bullet^1$	$\bullet^0 \xrightarrow{(2)} \bullet^1 \xleftarrow[(2)]{} \bullet^2$	$\bullet^0 \xleftarrow[(2)]{} \bullet^1 \xrightarrow{(2)} \bullet^2$	$\bullet^0 \xrightarrow{(1.5)} \bullet^1 \xrightarrow{(3)} \bullet^2$
Sans Conflit	Conflit E/E	Conflit S/S	Conflit E/S

FIG. 4.16 – Conflits simples et leur pénalité associée pour le réseau *Quadrics*.

Dans la section suivante, nous décrivons des expériences plus élaborées pour explorer le comportement induit par la concurrence sur les performances des communications.

4.3.4 Schémas élaborés de communication

Objectif :

L'exploration du comportement concurrent, à travers des schémas complexes de communications, s'effectue en augmentant le nombre de conflits. L'augmentation du nombre de conflits est obtenu soit en incluant de nouveaux nœuds soit en ajoutant de nouvelles communications. Cependant, le nombre de communications par nœud est limité par le nombre d'unité de calcul des nœuds. Ainsi, dans le cas des réseaux *Gigabit Ethernet* et *Myrinet*, chaque nœud ne peut pas avoir plus de deux communications. De ce fait dans cette partie, nous présentons pour *Gigabit Ethernet* et *Myrinet*, des expériences complexes entre conflits sous forme de chaînes de conflits. L'objectif de l'étude d'expériences élaborées est de déterminer la dépendance entre les pénalités des conflits suivant différents schémas de communications.

Méthode expérimentale :

La complexification des conflits entraîne une plus grande variabilité de la durée des communications concurrentes. En effet, dans les expériences précédentes, notre objectif était juste d'obtenir une vision du comportement de conflits simples pour toutes tailles de messages. Ainsi, une approche comparant les temps des communications était suffisante. Une comparaison de ces temps avec des temps de communications non soumis à la concurrence déterminait la pénalité de la concurrence pour chaque communication. Néanmoins, comme vu précédemment dans le cas de *Quadrics*, un conflit peut proposer des temps différents par communication. Il en résulte que des communications peuvent se terminer avant d'autres et ainsi modifier le conflit initial. Il en est de même pour la pénalité observée. Suivant ce contexte, la pénalité ne correspond pas exclusivement aux pénalités d'un seul conflit mais de plusieurs conflits se succédant jusqu'à la terminaison de chaque communication. Ainsi, les expériences étudiant uniquement les temps de communications, proposent des pénalités ne représentant pas le partage de la bande passante pendant la durée du seul conflit étudié. De ce fait, nous modifions les expériences de manière à obtenir la répartition de la bande passante du réseau en fonction des communications concurrentes, pour un seul conflit.

Ces nouvelles expériences ne proposent plus des temps de communications différents pour des tailles fixes de messages, comme lors des expériences introductives. Au contraire, elles proposent de déterminer les tailles de messages permettant d'obtenir des temps égaux pour chaque communication. Les pénalités sont calculées en comparant les tailles de messages envoyés durant

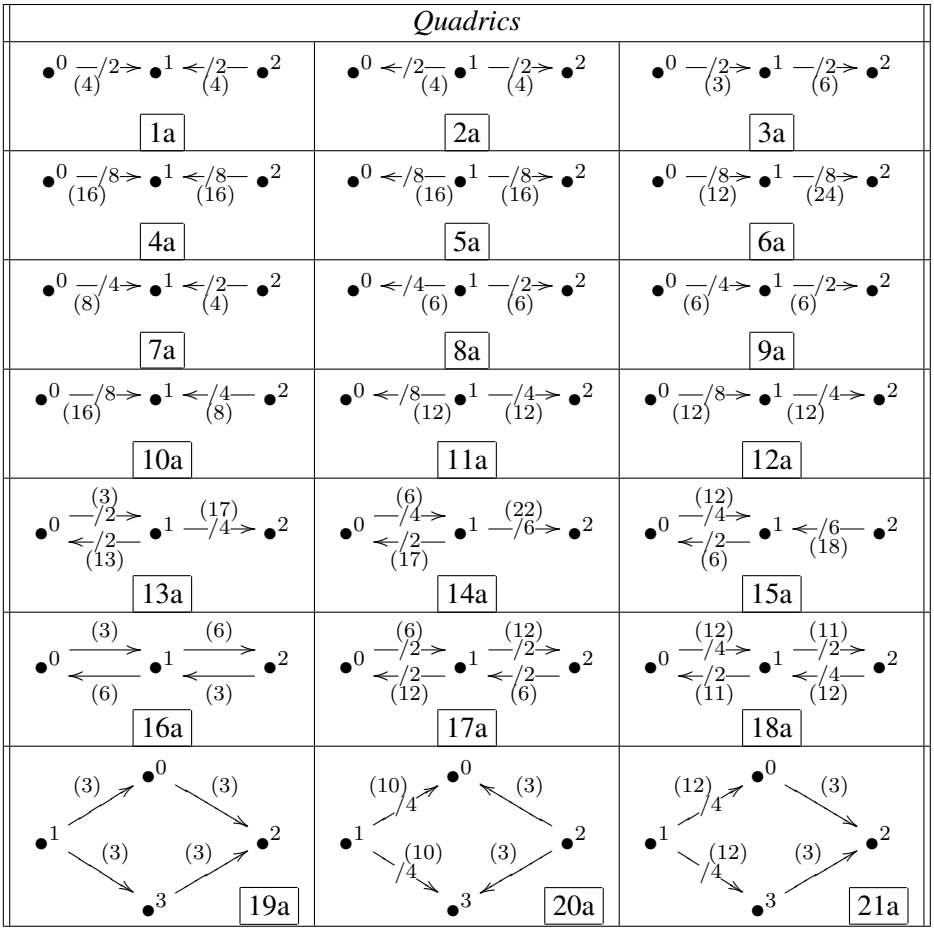


FIG. 4.17 – Conflits complexes *Quadrics*.

le conflit, et celles envoyées sans concurrence qui correspondent aux temps égaux observés. Ces temps sont de l'ordre de la seconde, ce qui représente des messages de l'ordre de la centaine de méga-octets. Néanmoins, la recherche de la taille des messages pour chaque communication nécessaire à l'obtention de temps égaux est réalisée manuellement. En effet, l'automatisation de ce procédé implique une recherche d'une égalité entre différents temps qui peut ne pas être trouvée automatiquement dans un délai raisonnable. Ainsi la précision des valeurs des pénalités diminue.

Les expériences suivantes proposent l'analyse des dépendances entre conflits simples et leurs influences sur la répartition de la bande passante entre les communications.

***Quadrics* :**

Dans le tableau 4.17, les résultats sur le réseau *Quadrics* montrent, dans un premier temps une forte similitude entre le nombre de communications en conflits et les valeurs des pénalités. En effet, par exemple, les conflits 1a, 2a, 4a et 5a ont leurs pénalités égales au nombre de communications en conflit. Plus précisément, dans le conflit 4a, chaque communication est ralentie d'un facteur 16

ou acquiert 1/16 de la bande passante durant la totalité du conflit. Ce phénomène intervient lors de conflits E/E ou S/S équilibrés en nombre de communications. Cependant, dans le cas de conflits non équilibrés en termes de communications entrantes et sortantes sur les nœuds, par exemple les conflits 7a, 8a, 10a et 11a les pénalités induites proposent des valeurs différentes. Ces valeurs bien que corrélées au nombre de communications ne suivent pas une formule directe en fonction du nombre de communications. De plus, les deux conflits E/E et S/S qui semblaient avoir le même comportement au niveau des pénalités lors de conflits équilibrés montrent une divergence. Cette divergence est nettement visible dans les conflits 10a et 11a. Le conflit E/S montre un comportement particulier. En effet, les communications entrantes sont faiblement ralenties en comparaison de celles sortantes. Ce phénomène se produit que les conflits soient équilibrés (conflits 3a et 6a) ou déséquilibrés (conflits 9a et 12a). Lorsque l'on mélange différents conflits sur plusieurs nœuds, nous obtenons des pénalités parfois très importantes. C'est par exemple le cas du conflit 14a, où le groupe de 6 communications sortant du nœud 1 prend une pénalité de 22 par communication. Il en résulte que la concurrence pose pour *Quadrics* de très forts problèmes de performance. Cette tendance se confirme pour les conflits 13a, 15a, 16a, 17a et 18a, où, dans certain cas, les pénalités sont proches de trois fois le nombre de communications sortantes. Nous remarquons sur ce point que les valeurs des pénalités sont souvent proches d'un multiple (entier ou rationnel) du nombre de communications sortantes ou entrantes. Finalement, lorsque le nombre de conflits est moins regroupé sur un ensemble de nœud restreint, les pénalités tendent à diminuer. Cette constatation que l'on peut qualifier de prévisible se produit dans les conflits 19a, 20a et 21a. En particulier, ce phénomène est visible entre les conflits 16a et 19a.

***Myrinet* :**

Dans les expériences présentées dans la figure 4.18, nous remarquons une similitude des pénalités entre les schémas 1b, 4b, 5b, 6b et 10b. En effet, ces schémas présentent tous une pénalité par communication de 1,9. Nous en déduisons, comme dans les expériences introductives, que des conflits E/E et S/S provoquent une même pénalité. Cependant, lorsque l'on construit une chaîne de conflits E/E et S/S, ces pénalités se modifient. Par exemple, dans les schémas 7b et 8b, nous notons une répartition non homogène des pénalités, qui se caractérise par une augmentation significative indiquant un plus grand ralentissement. De manière plus précise, l'ajout d'une communication au schéma 7b pour obtenir le schéma 8b (communication entre les nœuds 4 et 2) renforce d'une unité le ralentissement des communications sortantes du nœud 1. Parallèlement, en confondant les nœuds 3 et 4 du schéma 8b, donnant le schéma 10b, les pénalités se rééquilibrent. Si nous considérons les conflits E/S, conflits 2b et 6b, nous remarquons qu'aucune pénalité n'est ajoutée aux temps des communications.

Finalement, nous mélangeons, dans le schéma 11b, plusieurs conflits entre eux. Ainsi, ce dernier schéma peut-être vu comme la superposition des schémas 8b (nœuds 1, 0, 4, 5, 6) et 9b (nœuds 1, 2, 3, 6, 5). Le sous-schéma correspondant au schéma 8b prend des valeurs de pénalités proches du schéma 8b avec des ralentissements moins forts pour les communications sortantes du nœud 3. Le sous-schéma restant sans prendre en considération la communication entre les nœuds 5 et 6 (intervenant dans le sous schéma précédent), donne des valeurs identiques au schéma 9b.

En conclusion, nous constatons l'apparition d'une influence entre les pénalités des conflits E/E et S/S. Cette influence tend à augmenter les pénalités. Néanmoins, si les conflits sont équilibrés entre eux, c'est-à-dire même nombre de conflits E/E et S/S sur un même nombre de communi-

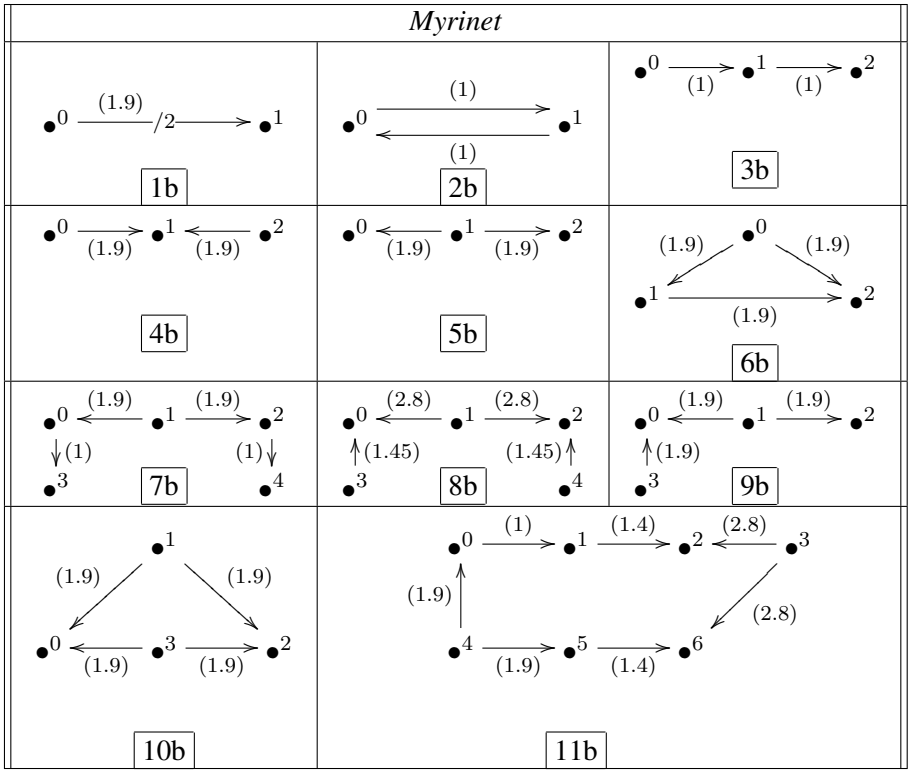


FIG. 4.18 – Conflits complexes *Myrinet*.

cations communes entre elles, les communications ne subissent que des pénalités propres à des conflits E/E ou S/S, sans une influence additionnelle. En opposition, les communications incluses uniquement dans des conflits E/S ne subissent aucune pénalité additionnelle.

***Gigabit Ethernet* :**

Les expériences complexes réalisées sur le réseau *Gigabit Ethernet* sont présentées dans la figure 4.19. Une première constatation est la faible différence entre les pénalités des schémas de manière générale. Plus particulièrement, chaque communication d'un schéma faisant intervenir plusieurs conflits E/E ou S/S prend une pénalité proche de la pénalité d'un schéma E/E ou S/S seul. Ce phénomène est visible dans le schéma 11c. Néanmoins, un léger déséquilibre intervient dans les schémas 3c et 7c qui proposent des pénalités plus élevées aux communications plus fortement soumises aux conflits E/E et S/S. En particulier, la communication entre les nœuds 1 et 0 du schéma 7c montre une pénalité augmentée de 0,2 par rapport au conflit S/S simple. Nous remarquons, que dans ce cas, cette augmentation se traduit aussi par l'accélération des deux autres communications. Cette accélération équivaut à une pénalité inférieure de 0,2 par rapport aux conflits E/E et S/S simples. Comme dans le cas de *Myrinet*, les conflits E/S sont moins soumis à la concurrence. Cependant, *Gigabit Ethernet* ajoute un faible surcoût, que l'on retrouve par exemple dans le schéma 2c.

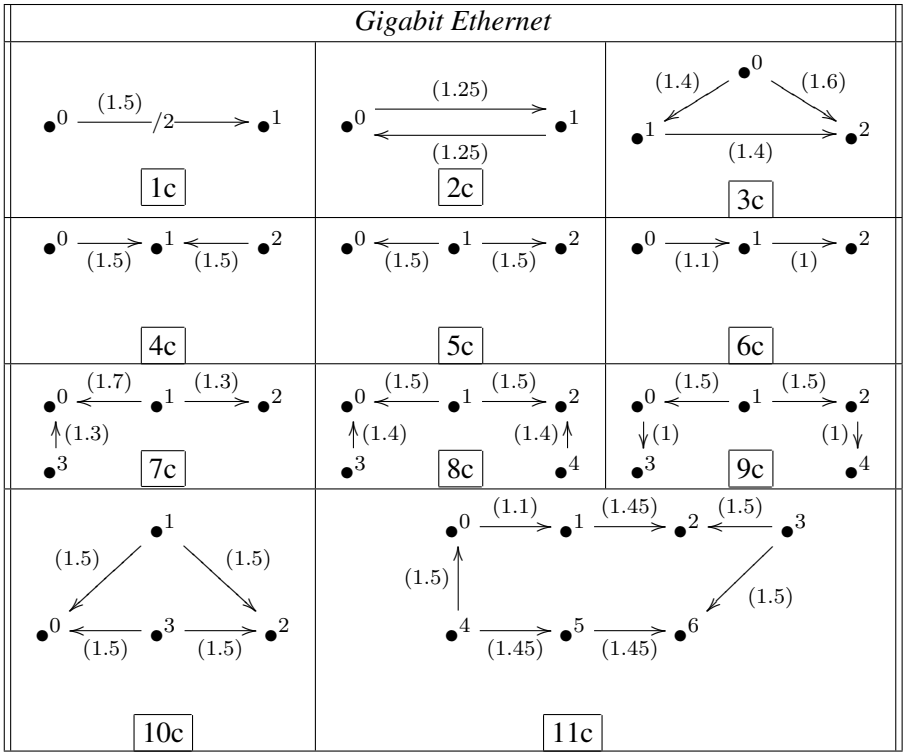


FIG. 4.19 – Conflits complexes, *Gigabit Ethernet* et *Mpich*.

Pour conclure, nous dirons que de façon générale les expériences *Gigabit Ethernet* donnent des résultats en termes de pénalités proches des pénalités des conflits élémentaires. Cela se vérifie quel que soit la complexité du schéma. En outre, dans le cas d'un déséquilibre du schéma en termes de conflits, aucune pénalité additionnelle apparaît mais au contraire une compensation se crée. Plus précisément, une communication fortement ralentie a tendance à accélérer ses communications voisines (schéma 7c ou 3c). Les conflits E/S, comme pour *Myrinet*, proposent une indépendance relative aux autres conflits en subissant une légère pénalité.

4.3.5 Modification des paramètres des communications

Nous avons vu précédemment que des expériences de concurrence entre communications de même taille et partant simultanément provoquent des pénalités différentes. En outre, en modifiant les expériences se basant uniquement sur les temps de communications, nous avons cherché à montrer l'influence des conflits sur la répartition de la bande passante. Cependant, en se basant sur la notion de communications différentes en conflit, certaines communications se terminent avant d'autres, modifiant ainsi les conflits et les pénalités associées. En effet, pour un schéma donné, une communication soumise à un fort ralentissement ou une communication de grande taille se terminera probablement après d'autres communications. Il convient donc d'étudier des cas de communications hétérogènes en taille et dates de départ. Cette étude a pour but de déterminer l'évolution

	Sans conflit	Conflit S/S	Conflit S/S + date de départ
Temps (4Mo) [ms]	47.9	com1 : 70.3 ; com2 : 70.3	com1 : 60.6 ; com retardée : 59.6

TAB. 4.4 – Observation des dates de départ différentes sur la concurrence, *Gigabit Ethernet Mpich*.

des pénalités de la concurrence lors de la terminaison d'une communication d'un conflit. Dans les paragraphes suivants nous étudions deux principaux cas. Le premier cas modifie la date de départ de deux communications de même taille. Le deuxième cas propose une expérience totalement hétérogène à la fois en taille et en date de départ. Pour une plus facile compréhension des résultats expérimentaux, chacun des deux cas présentés sera appliqué au conflit S/S. Ce conflit a l'avantage de proposer des pénalités de concurrence similaires aux deux communications le constituant et ce pour chaque réseau haute performance.

4.3.5.1 Temps de départ différents

L'obtention de dates de départ différentes est réalisé, dans le programme test, grâce à un délai avant l'émission des données. Cette méthode donne des prédictions de l'ordre de la microseconde. Elle effectue un nombre d'instructions nulles (*Nop instruction*) correspondant au temps d'attente désiré. Dans les expériences suivantes, nous utiliserons le conflit S/S avec deux communications de taille égale à 4Mo.

Sur un réseau *Gigabit Ethernet* et une implantation *Mpich*, envoyer 4Mo de données prend un temps de 47,9 ms. Ainsi, deux communications en concurrence suivant le conflit S/S prendrons chacune un temps égal à 70,3 ms. De cette constatation, nous proposons de retarder une des deux communications de la moitié du temps sans concurrence, soit 24 ms. De cette manière, le conflit apparaîtra à la moitié du temps de diffusion de la communication non retardée et se poursuivra le temps qu'une des deux communications se termine.

La réalisation de cette expérience montre dans le tableau 4.4 un comportement pour lequel chacune des communications est retardée d'un temps nécessaire à l'envoi de 1Mo. La première communication envoie ses données avec une pénalité de 1 (sans ralentissement) jusqu'au moment du départ de la seconde communication. A cet instant, la première communication aura déjà envoyée la moitié de ses données. Une fois que le conflit apparaît, chacune des deux communications transfère ses données avec une pénalité de 1,5 (cette valeur est spécifique au conflit S/S). Seulement, la communication retardée a deux fois moins de données à envoyer, elle se termine donc avant. Au moment de cette terminaison, le conflit disparaît et la pénalité de la communication restante vaut 1. Finalement, les deux communications ne sont en conflit que pendant l'envoi de 2Mo, expliquant la valeur ajoutée aux temps. Nous pouvons calculer directement le temps des communications : $t_{4Mo} = t_{2Mo} + 1.5 * t_{2Mo} = 60ms$. Nous en concluons, et grâce à d'autres expérimentations non présentées dans ce document, que les pénalités de la concurrence se modifient au moment de la création mais aussi de la terminaison des communications. Le même phénomène se produit pour le réseau *Myrinet* et le réseau *Quadrics*.

4.3.5.2 Temps de départ différents et tailles différentes

Le changement de taille n'apporte pas de nouveaux comportements. En effet, modifier la taille d'une communication modifie intrinsèquement son temps de diffusion. Par conséquent, ce cas de figure correspond à la situation précédemment mentionnée.

4.3.6 Bilan des expériences

Ces expériences démontrent une certaine cohérence entre le nombre de communications en concurrence et les valeurs des pénalités créées par les phénomènes de concurrence. Cependant, une dépendance entre les conflits intervient. Chaque réseau gère de manière différente la répartition de la bande passante entre les communications concurrentes. Il est intéressant de noter que les réseaux les plus optimisés en bande passante et en terme de latence sont les moins performants dans la gestion de la concurrence. Ainsi *Quadrics* et *Myrinet* présentent des pénalités parfois supérieures au nombre de communications signifiant qu'un temps additionnel est perdu dans la gestion de la concurrence. A l'inverse, *Gigabit Ethernet* limite les effets de la concurrence en proposant une compensation entre communications fortement soumises aux conflits et celles moins influencées. Cet effet provient de la manière dont le contrôle de flux est réalisé. Dans le cas de *Gigabit Ethernet*, nous remarquons que TCP est nettement plus performant que les contrôles de flux bas niveau de *Quadrics* ou *Myrinet*.

Un point intéressant, soulevé par ces expériences, est le caractère dynamique des schémas. En effet, suivant le schéma et les conflits engendrés, les pénalités induites par la concurrence sont elles aussi variables. Cet effet provoque une hétérogénéité des terminaisons des communications et donc une modification du schéma initialement utilisé. L'analyse de communications hétérogènes en taille de message et en date de départ démontre que les pénalités appliquées suivent la dynamique du schéma de communication. Ainsi, si une nouvelle communication modifie un schéma à un instant donné, les pénalités se modifient en fonction du nouveau schéma créé, et sont appliquées jusqu'à une nouvelle modification du schéma.

4.4 Discussion

L'objectif des expérimentations réalisées est d'analyser le comportement de communications en concurrence. Il a été montré que certains comportements peuvent être observés sur plusieurs réseaux. Chaque réseau a aussi ses propres spécificités qui font apparaître des comportements particuliers. Ces différents comportements ont été détaillés dans les paragraphes synthétisant les résultats obtenus. Dans cette section, notre objectif est de discuter l'utilité et l'applicabilité de ces résultats pour établir un modèle prédictif.

De ce fait, nous proposons une classification des propriétés observées en deux catégories : la première regroupe les comportements simples et la seconde les comportements élaborés. Cette classification nous permettra dans le chapitre suivant d'aborder l'origine des modèles proposés.

Comportements simples

Une propriété simple qui peut être facilement modélisée est la correspondance entre les pénalités et les temps de communications. En effet, une pénalité correspond à une valeur multipliant le

temps de communication sans concurrence. Ainsi, connaissant la pénalité nous pouvons prédire la durée d'une communication. L'obtention des pénalités est elle plus compliquée à réaliser, nous la classerons dans la catégorie des comportements complexes. L'évolution des pénalités des communications suit l'évolution du schéma de communications. Ainsi, connaissant une méthode pour obtenir les pénalités, il devient facile de prédire les temps de communications hétérogènes en date de départ et en taille.

Comportements élaborés

Il peut s'avérer difficile d'obtenir des pénalités pour un schéma donné. Cependant, nous avons remarqué, dans les expériences précédentes, que les pénalités peuvent être associées au comportement du contrôle de flux du réseau. Ainsi, une première approche pour l'obtention de ces valeurs serait une modélisation du comportement du contrôle de flux. Néanmoins, la complexité d'une telle modélisation reste délicate.

En conclusion, un modèle de prédiction devra donc permettre de déterminer une méthode de calcul des pénalités pour un schéma donné. Cette méthode devra être basée, a priori, sur le comportement du contrôle de flux associé au réseau. Une fois cette méthode connue, le modèle prédictif se basera sur deux propriétés simples : une méthode calculant le temps des communications et une associant les pénalités à l'évolution dynamique du schéma.

Définitions de modèles de communications concurrentes 5

5.1 Introduction

Les expériences du chapitre précédent ont montré que la concurrence entre communications se manifeste à travers un ensemble de pénalités. Ces pénalités représentent les surcoûts associés aux temps de communication. En outre, ces expériences ont révélé une hiérarchie entre pénalités et schémas de communications. Il devient alors possible de proposer des modèles prédictifs fondés sur la hiérarchie observée entre pénalités.

Dans le développement de ces modèles prédictifs, nous choisirons deux principaux axes :

– une approche descriptive qui décrit les mécanismes de contrôle de la concurrence établis par les technologies matérielles et logicielles du réseau considéré ;
– une approche quantitative, qui sans décrire ce mécanisme de contrôle, cherche à retrouver la hiérarchie entre pénalités et schémas de communications.

Avant de décrire les modèles réseaux, il est nécessaire de détailler, dans la section 5.2, l'orgine des modèles et de présenter, dans la section 5.3, une formalisation du contexte.

Ce chapitre présente différents modèles prédictifs pour deux des réseaux étudiés précédemment. Les modèles sont basés sur une partie commune et une partie spécifique à chaque réseau.

L'évolution dynamique d'une application, en terme d'accès à la ressource réseau, est commune à chaque réseau (c'est-à-dire qu'elle ne dépend pas du réseau). Pour prédire les temps de communications de chaque processus de l'application, nous utiliserons une simulation à événements discrets [53]. Une description de son principe de fonctionnement est proposée dans la section 5.4. Cette simulation utilisera entre chaque phase dynamique le modèle de répartition de la bande passante du réseau modélisé.

La répartition de la bande passante, c'est-à-dire le calcul des pénalités, est spécifique aux caractéristiques de chaque réseau. La section 5.5 présente ces modèles. Ainsi, pour chaque modèle, nous proposons, de manière descriptive ou quantitative, des techniques pour obtenir ces pénalités.

5.2 Origine, objectif et conception des modèles

Le chapitre 4 a défini un protocole expérimental qui a conduit à la réalisation de plusieurs expériences. Des études préliminaire [57] et les résultats de ces expériences ont permis de comprendre

et appréhender le phénomène de concurrence réseau.

En résumant ces résultats, les temps des communications concurrentes sont fonction du graphe de communication qui conduit à une répartition de la bande passante. Une fois cette répartition connue pour un graphe donné, il devient possible par une simulation à événements discrets de déterminer le temps de chaque communication soumise à une évolution du graphe. La conception du modèle de prédiction se base sur ces remarques en se scindant en deux parties indépendantes :

- une première partie consiste à appliquer ces pénalités suivant la dynamique du graphe des communications ;
- une seconde partie a pour objectif de déterminer les pénalités associées à un graphe.

Nous avons montré à la fois l'importance du contrôle de flux dans la gestion de la concurrence et la corrélation entre les valeurs des pénalités et les graphes de communications. De ce fait, l'origine des modèles de répartition de la bande passante provient aussi bien de l'étude du comportement du contrôle de flux que de l'étude des corrélations ou hiérarchies entre pénalités. Nous appellerons, modèle descriptif un modèle basé sur le comportement du contrôle de flux et modèle quantitatif un modèle basé sur l'observation des hiérarchies entre pénalités. Les réseaux *Myrinet* ou *Quadrics* ont un contrôle de flux moins complexe qui peut permettre une modélisation descriptive. A l'inverse, le réseau *Gigabit Ethernet* propose un contrôle de flux basé sur TCP qui a un comportement difficile à appréhender comme l'ont montré les auteurs de [77]. Ainsi, pour ce réseau nous proposons uniquement un modèle quantitatif.

Avant de présenter les différents modèles, nous formalisons, dans la section suivante, les notions de communications concurrentes et de graphes de communications.

5.3 Formalisation du problème

Dans cette section, nous proposons un formalisme permettant de représenter les communications concurrentes générées par une application. Ce formalisme, à base de graphe, englobe les différents cas de concurrence entre les communications. Nous rappelons que pour une application, la concurrence entre les communications n'est pas constante au cours du temps. Nous pouvons facilement imaginer durant l'exécution d'une communication, l'apparition et la terminaison de communications entrant en concurrence avec celle-ci. De cette manière, la concurrence entre les communications est un phénomène évolutif dans le temps.

5.3.1 Formalisme de communications

5.3.1.1 Communication simple

Une communication réseau point à point consiste à envoyer ou à recevoir des données entre deux tâches communicantes. Une communication est généralement définie par trois attributs : le nœud source émetteur des données, le nœud destinataire des données, et le volume des données transférées. Pour l'analyse des performances, nous pouvons rajouter deux attributs supplémentaires : la date de départ et la durée de la communication. En numérotant les nœuds source et destination par des nombres entiers, nous pouvons proposer la définition suivante pour une communication :

Définition 1 *Une communication correspond à un quintuplé de valeurs :* $e = (s_p, d_p, l, t_d, t_l)$ *avec :*

- s_p *le numéro du nœud émetteur ;*
- d_p *le numéro du nœud récepteur ;*
- l *la taille du message à transférer ;*
- t_d *la date de début de la communication ;*
- t_l *la durée de la communication.*

□

Une communication possédant un nœud émetteur et un nœud destinataire identique correspond à une communication intra-nœud. Par analogie, une communication ayant un nœud source et un nœud destination différent est appelée une communication inter-nœud.

5.3.1.2 Communication concurrente

Un nœud multiprocesseur peut-être la source et/ou la destination de plusieurs communications. Dans une telle situation, des communications ayant une source ou une destination commune deviennent des communications concurrentes. En d'autres termes, une communication est dite en concurrence si durant le transfert des données, elle partage une partie du chemin réseau qu'elle emprunte avec d'autres communications. Par exemple, deux communications ayant une même destination sont en concurrence si durant un certain laps de temps elles partagent une partie du chemin réseau tel que le lien réseau, la carte réseau ou le bus *PCI* du nœud destinataire. Cependant, nous avons vu dans les expériences du chapitre 4 que les communications sont en concurrence uniquement à partir d'une certaine taille de données à envoyer. De ce fait, nous introduisons une taille minimale des données pour que deux communications soient en concurrence. Ceci nous conduit à la définition des communications concurrentes :

Définition 2 *Une communication* $e = (s_p, d_p, l, t_d, t_l)$ *est dite concurrente, s'il existe au moins une communication* $e' = (s'_p, d'_p, l', t'_d, t'_l)$ *telle que :*

- $s_p = s'_p$ *ou* $d_p = d'_p$ *ou* $s_p = d'_p$ *ou* $d_p = s'_p$ *;*
- *et* $[t_d; t_d + t_l] \bigcap [t'_d; t'_d + t'_l] \neq \emptyset$ *;*
- *et* $l > L$ *et* $l' > L$*, avec* L *taille minimale des données à envoyer pour être en concurrence.*

On dit alors que e *et* e' *sont en concurrence.*

Pour représenter, les différentes possibilités de concurrence réseau générées par une application, nous définissons un formalisme des schémas de communications.

5.3.2 Schéma de communications

Le schéma de communications ou graphe de communications tient son origine aussi bien des communications créées par les tâches que du placement de ces tâches sur les nœuds de la grappe.

5.3.2.1 Formalisme des applications

Une application est formalisée par un ensemble de tâches. Une tâche correspond à une suite d'événements. Un événement peut être de deux types :

- un événement de calcul correspond à une phase de calcul de la tâche. Il est caractérisé par une durée du calcul.
- un événement de communication correspond soit à l'émission soit à la réception de données. Un événement de communication en émission contient comme information le numéro de la tâche destinataire ainsi que le volume de données à transférer. Un événement de communication en réception contient uniquement le numéro de la tâche émettrice.

Suivant le placement de ces tâches sur les nœuds, il est possible d'obtenir différents graphes de communications.

5.3.2.2 Graphe de communications

Une application parallèle génère lors de son exécution plusieurs phases de calcul et de communication. En outre, l'apparition ou la terminaison d'une communication au sein de l'application évolue au cours du temps impliquant une modification du graphe de communications.

Pour représenter ce phénomène, nous considérons un graphe de communications comme étant une suite de graphes $G_L(t)$ où t représente l'instant considéré. A ce stade, il s'agit d'un graphe logique entre les tâches de l'application.

Définition 3 *Un ensemble de tâches d'une application, et de communications entre ces tâches est représenté, à l'instant t, par un graphe orienté $G_L(V_l, E_l, t)$:*

- *V_l est l'ensemble des tâches de l'application ;*
- *E_l est l'ensemble des communications entre les tâches ;*
- *t est l'instant considéré.*

□

Le choix d'un placement des tâches sur les nœuds d'une grappe transforme le graphe logique (et par conséquent la suite de graphes logiques) en un graphe physique.

Définition 4 *Le placement des tâches sur les nœuds d'une grappe est représenté par une relation P transformant une suite de graphes logiques $G_L(V_l, E_l)$ en une suite de graphes physiques $G_P(V_p, E_p) = P(G_L)$. Cette relation fusionne les éléments de l'ensemble V_l pour former l'ensemble V_p en préservant le nombre d'arcs.* □

Une suite de graphes physiques est donc définie comme suit :

Définition 5 *Un ensemble de nœuds d'une grappe de calcul, et de communications entre ces nœuds est représenté, à l'instant t, par un graphe orienté $G_P(V_p, E_p, t)$:*

- *V_p est l'ensemble des nœuds du graphe correspondant aux nœuds de la grappe de calcul ;*
- *E_p est l'ensemble des communications entre les nœuds ;*
- *t est l'instant considéré.*

□

Nous noterons une suite de graphe $G_P(V_p, E_p, t)$ par G_t. Une suite de graphes G_t passe de l'instant t à $t+1$ soit par l'initiation soit par la terminaison d'une communication. Une communication $e \in E_p$, avec $e = (v_s, v_d) \in V_p$, est représentée par un arc orienté, où le nœud v_s correspond au nœud émetteur de la grappe de calcul et où le nœud v_d correspond au nœud récepteur. Une communication intra-nœud correspond à une boucle, c'est-à-dire $v_s = v_d$.

De plus, pour chaque nœud, nous définirons par Δ_e le degré entrant d'un nœud, c'est-à-dire le nombre de communications ayant comme destination ce nœud. Par analogie, Δ_s est le degré sortant d'un nœud, c'est-à-dire le nombre de communications émises par un nœud.

Nous illustrons, dans la sous-section suivante, ce formalisme basé sur ces définitions.

5.3.3 Exemples

Afin d'illustrer la création d'une suite de graphe G_t, nous décrivons une application. Par simplicité, chaque tâche de l'application est restreinte à être uniquement une tâche en réception ou une tâche en émission. Les tâches sont identifiées par un entier n. Nous noterons par R, une tâche réceptrice et par E une tâche émettrice. Nous notons également, pour une tâche en réception, s_l le numéro logique de la tâche source de la communication, et pour une tâche émettrice, nous notons par d_l le numéro logique de la tâche destinataire. Dans le cas d'une tâche émettrice, nous ajoutons T_d la date de départ d'émission des données et T_f le temps final d'émission. Ainsi, par exemple, une tâche émettrice identifiée par le numéro 1 est dénoté par $E_1(d = 2, T_d = 0, T_f = 0.3)$, cette tâche envoie ses données à la tâche réceptrice numérotée 2 dénoté $R_2(s_l = 1)$. Dans cet exemple, la taille des données n'est pas considérée car le temps de communication est suffisant.

En utilisant ce formalisme, la figure 5.1 montre une illustration d'un ensemble de 8 tâches. Cet ensemble de 8 tâches, qui génère 4 communications, sera utilisé comme base d'exemple des tâches communicantes afin de générer la suite de graphes G_t.

$E_1(d_l = 2, T_d = 0.0, T_f = 0.3)$	$R_2(s_l = 1)$
$E_3(d_l = 4, T_d = 0.1, T_f = 0.5)$	$R_4(s_l = 3)$
$E_5(d_l = 6, T_d = 0.2, T_f = 0.5)$	$R_6(s_l = 5)$
$E_7(d_l = 8, T_d = 0.3, T_f = 0.6)$	$R_8(s_l = 7)$

FIG. 5.1 – Exemple d'un ensemble de 8 tâches communicantes.

Pour obtenir G_t, il est nécessaire de choisir un placement des tâches sur les nœuds d'une grappe de calcul. Nous proposons deux exemples de grappes. La grappe $A(4 \times 2)$ est constituée de quatre nœuds biprocesseurs, et la grappe $A(4 \times 4)$ est constituée de quatre nœuds quadriprocesseur. Chaque nœud est identifié par un entier. A chaque grappe est associé un placement des tâches. Le tableau 5.1 exprime un placement des 8 tâches sur les nœuds des deux grappes proposées. Ainsi, par exemple pour la grappe $A(4 \times 2)$, les nœuds 2, 5 et 8 du graphe logique sont fusionnés vers le nœud 2 du graphe physique.

		Grappes	
		$A(4 \times 2)$	$A(4 \times 4)$
Nœuds physiques	0	E_1	E_1,R_4,E_7
	1	R_4,R_6	E_5,R_6,R_8
	2	R_2,E_5,R_8	R_2
	3	E_3,E_7	E_3

TAB. 5.1 – Exemple de placement des tâches sur les grappes.

Exemples de graphes :

A partir d'une grappe de calcul, d'un ensemble de tâches communicantes, et d'un placement de ces tâches sur les nœuds de la grappe, une suite de graphes G_t est générée. La figure 5.2 représente cette suite G_t de l'ensemble des tâches sur la grappe $A(4 \times 2)$. Il est intéressant de noter que par exemple au temps t_1 les communications e_1 et e_2 ne sont pas en concurrence car elle n'ont pas une source ou une destination commune. Cependant, au temps t_3, l'ajout de la communication e_3 provoque une concurrence entre e_1 et e_3 ainsi qu'entre e_2 et e_3. Notons que cette concurrence ne se produit que si la taille des données envoyées par e_3 est suffisante. A ce point, les degrés du nœud 1 sont $(\Delta_e = 2, \Delta_s = 0)$ et les degrés du nœud 2 sont $(\Delta_e = 1, \Delta_s = 1)$. Lorsque e_1 se termine e_4 commence. A l'instant t_3, les trois communications e_2, e_3 et e_4 sont en concurrence. Puis finalement, au temps t_4, e_2 et e_3 sont terminées et e_4 reste seule sans concurrence.

En utilisant le même ensemble de tâches, mais en changeant la grappe de calcul et le placement, nous obtenons une autre suite de graphes G_t. La figure 5.3 propose la suite G_t pour la grappe $A(4 \times 4)$. Un point intéressant se situe au temps t_3. En effet, comme la grappe $A(4 \times 4)$ comporte des nœuds à quatre processeurs, il est possible d'obtenir une communication interne en concurrence avec une communication externe. C'est le cas entre les communications e_3 et e_4. Remarquons qu'en se limitant à la contrainte d'une tâche par processeur, ce cas ne pouvait pas se produire dans la grappe $A(4 \times 2)$.

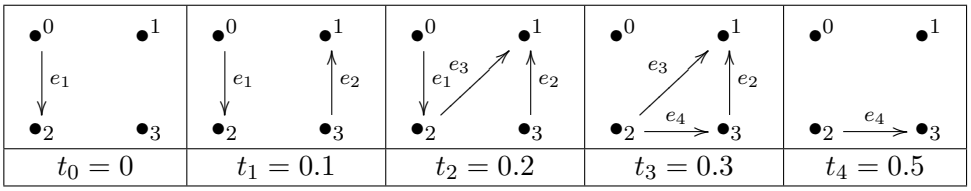


FIG. 5.2 – G_t généré sur la grappe $A(4 \times 2)$

5.4 Modèle prédictif des temps de communications

Avant de déterminer les modèles de répartition de la bande passante ou en d'autres termes les valeurs des pénalités, il convient de déterminer comment ces pénalités s'appliquent pour déterminer le temps des communications en concurrence d'une application.

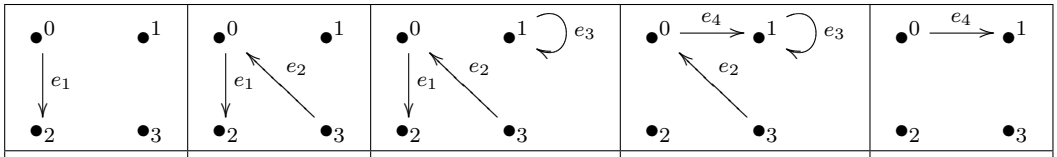


FIG. 5.3 – G_t généré sur la grappe $A(4 \times 4)$

5.4.1 Objectif

Le formalisme décrit au début de ce chapitre montre l'aspect dynamique des graphes de communications d'une application. Cet aspect dynamique est basé sur l'apparition ou la terminaison des communications. L'objectif de ce modèle est de prédire le temps de communication d'une suite de graphes G_t.

5.4.2 Approche algorithmique

Pour déterminer le temps des communications d'une application, nous utiliserons une simulation à événements discrets [53]. Les événements considérés sont l'initiation et la terminaison des communications d'un graphe dynamique G_t. De ce fait, nous définissons deux types d'événements : événement début de communication, et un événement fin de communication. En d'autres termes, deux cas de transition sont possibles : un graphe G_t passe de l'instant t à $t + 1$ soit par l'initiation soit par la terminaison d'une communication.

L'application fournit une suite déterministe d'événements initiateur des communications. Les événements de terminaison se produisent lorsqu'une communication qui émet n'a plus de données à envoyer. L'algorithme de la simulation consiste à calculer la quantité de données envoyées pour chaque communication entre deux événements, jusqu'à la terminaison de toutes les communications.

Pour calculer la quantité de données envoyées, nous utilisons une simple équation affine, comme proposé dans le modèle de Hockney [78] :

$$t_c = \frac{p * m}{b} + l$$

Cette équation calcule le temps d'une communication c en fonction de la bande passante du réseau b, de la latence l, de la taille de message m et de la pénalité p engendrée par le graphe G_t à l'instant t.

Suivant cette équation, entre deux événements et pour une pénalité donnée, il devient facile de calculer la quantité de données envoyées. L'algorithme est succinctement présenté dans la figure 5.4.

5.4.3 Exemple

La figure 5.5 est un exemple d'exécution de l'algorithme présenté. Durant la première étape entre les temps A et B, la communication (1) envoie ses données sans conflit. Quand la communication (2) commence à émettre au temps B, les deux communications sont en conflit E/E. Donc

Début de l'algorithme

```
; Parameters in:
;        Cluster : Formalism   // Formalisme de la grappe
;        Application  : Formalism   // Formalisme de l'application
;        Mapping : Formalism   // Formalisme du placement des tâches  sur les nœuds
;        Model : Parameter   // Paramètres du modèle (bande passante,etc)
;
; Parameters out:
;        communications : array of Communication // Temps des communications


G : Graph // graphe de communication
c : Communication
p : array of double // tableau des pénalités, p[c] est la pénalité de la communicaction c
time_interval_end : time
time_interval_start : time
time_interval : time
time_step : time
bw : double
l : double


bw = GetModelBandwidth(Model)
l = GetModelLatency(Model)
time_step = 0
// construction du graphe au temps 0
G := buildGraph(Cluster,Application,Mapping,time_step)

while( G has communication) {
        // déterminer les pénalités à partir du modèle
        p := model_of_bandwidth_sharing(G,Model)

        // déterminer le premier événement  à se produire: début ou fin d'une communication

        // intervalle de temps avant le début d'une nouvelle communication
        time_interval_start = TimeOfNextCommunicationToStart(G,time_step) - time_step

        // intervalle de temps jusqu'à la terminaison d'une communication
        c = FindFirstCommunicationToEnd(G,p,time_step)
        time_interval_end = GetCommunicationSize(c)*p[c]/bw + l

        // intervalle de temps avant le premier événement
        time_interval = min(time_interval_start,time_interval_end)


        // déterminer les quantités de données envoyées pour chaque communication
        foreach c in G {
                SetCommunicationSize(GetCommunicationSize(c) - (bw*(time_interval - l))/p[c])
                SetCommunicationTime(GetCommunicationTime(c) + time_interval)
        }

        // construction du graphe au temps time_step + time_interval
        time_step = time_step + time_interval
        G := buildGraph(Cluster,Application,Mapping,time_step)

}
```

Fin de l'algorithme

FIG. 5.4 – Simulation à événements discrets

en appliquant le modèle de répartition de la bande passante, nous déterminons les pénalités et par conséquent la quantité de données envoyées par chaque communication. Lorsqu'il n'y a plus de nouvelle communication, ou quand une communication se termine avant le départ d'une nouvelle communication, chaque communication a une quantité de données restantes à envoyer avec un nombre de conflits décroissant. C'est le cas à l'étape D, quand la communication (4) commence à émettre. A ce point, la première communication qui doit se terminer peut facilement être calculée, ici la communication (3), et ainsi obtenir le point E. Nous calculons pour les communications (1), (2) et (4) la quantité de données envoyées durant l'intervalle D-E.

D'un point de vue plus général, cet algorithme ne dépend pas du modèle de répartition de la bande passante. Ce qui permet de faire varier le modèle suivant le réseau choisi.

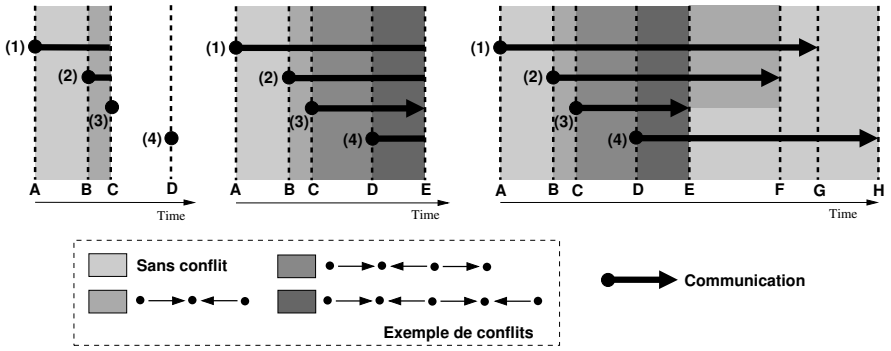


FIG. 5.5 – Exemple d'application du modèle prédictif des temps de communication.

5.5 Modèle de répartition de la bande passante

Après avoir formalisé le problème et montré comment il est possible de calculer les temps des communications d'une application, nous définissons les modèles de répartition de la bande passante. Ces modèles sont spécifiques à chaque réseau.

5.5.1 Objectif

L'objectif du modèle est de prédire les pénalités associées à chaque communication pour un graphe donné. Ces pénalités décrivent la répartition de la bande passante entre communications. Pour ces modèles, les graphes considérés sont statiques et correspondent aux schémas de communication entre deux événements, c'est-à-dire l'apparition ou la terminaison d'une communication. La prédiction est réalisée, soit par modèle quantitatif, soit par un modèle descriptif.

5.5.2 Approche quantitative

Un modèle quantitatif détermine les pénalités associées aux communications complexes en fonction de mesures réalisées sur des graphes simples [60]. L'objectif étant de trouver des règles qui permettent de décomposer un graphe complexe en graphe simple et de pouvoir y associer

des pénalités [61][58]. Dans ce cas, le mécanisme du partage de bande passante n'est pas décrit, seulement ses effets en sont modélisés [59].

Dans la sous-section suivante, nous proposons deux modèles quantitatifs pour les réseaux *Myrinet* et *Gigabit Ethernet*. Ces modèles sont dans un premier temps limité à des graphes comportant un degré des nœuds inférieur ou égale à 2, c'est-à-dire graphe en forme de chaîne ou anneau. Dans une seconde étape, nous modifions le protocole expérimental du chapitre 4 pour franchir cette limitation sur le degré des nœuds. Après avoir succinctement présenté quelques nouvelles expériences suivant ce nouveau protocole, nous établirons un modèle quantitatif général pour *Gigabit Ethernet*. Pour *Myrinet*, nous proposons ensuite un modèle descriptif généralisé.

5.5.2.1 Modèle quantitatif pour 2 processeurs

Les modèles quantitatifs des réseaux *Gigabit Ethernet* et *Myrinet* se basent sur un ensemble de règles. Ces règles permettent de décomposer le graphe en sous graphes élémentaires pour lesquels les pénalités sont connues (elles ont été évaluées par expérimentations).

Réseau *Gigabit Ethernet*

Graphes élémentaires : Les graphes élémentaires sont : les communications en conflit E/S (par exemple les schémas 2c et 6c de la figure 4.19 du chapitre précédent) et les suites de communications en conflit E/E et S/S, ou en d'autres termes, les suites de conflit ne comportant pas un conflit E/S (par exemple les schémas 10c et 7c).

Règles :

Equilibre Si pour une suite de communications en conflit E/E et S/S, le nombre de communications est pair (équilibre) alors chaque communication prend une pénalité équivalente. Si ce nombre est impair (déséquilibre), les pénalités appliquées oscillent entre deux valeurs de façon périodique.

Dominance Si une communication appartient à la fois à un conflit E/E ou S/S et à un conflit E/S, alors la valeur de la pénalité appliquée est celle du conflit E/E ou S/S. Les conflits E/E et S/S dominent les conflits E/S.

Pénalités des graphes élémentaires : pour la grappe de Rennes, les communications faisant partie d'un conflit ou d'une suite de conflit E/S prennent comme pénalité une valeur de 1,1. Lors d'une suite de conflit S/S ou E/E et un nombre pair de communication les pénalités valent 1,5. Si le nombre de communication est impair alors les pénalités oscillent entre 1,3 et 1,7.

La règle d'équilibre peut être expliquée par le protocole TCP qui tend à homogénéiser les pénalités, ce qui se traduit par une certaine forme de régularité. La règle de dominance s'explique par l'utilisation de liens full-duplex. En effet, dans un conflit E/S, une communication sortante et une communication entrante peuvent acquérir le lien simultanément.

Réseau *Myrinet*

Graphes élémentaires : Les graphes élémentaires sont les conflits élémentaires : conflit E/S, conflit S/S et conflit E/E. De plus, nous ajoutons le graphe élémentaire constitué de trois communications en concurrence comprises dans un conflit S/S et un conflit E/E (schéma 9b, figure 4.18, chapitre 4). Nous noterons ce graphe par conflit E/S/S (mais la notation S/S/E serait équivalente).

Règles :

Dominance Les conflits S/S sont les conflits les plus dominants. Un graphe en forme de chaîne ou anneau se décompose en premier en conflit S/S, puis le cas échéant en conflit E/S/S si un conflit E/E précède ou suit un conflit S/S. Par la suite, le graphe se découpe ensuite en conflit E/E puis finalement en conflit E/S. Si nous représentons l'ordre de dominance par le signe > nous obtenons : S/S (ou E/S/S le cas échéant) > E/E > E/S.

Pénalités des graphes élémentaires : Les communications d'un graphe élémentaire du type E/S ont pour pénalités 1,0 ; les pénalités d'un conflit E/E ou S/S sont de 1,9 par communication. Finalement, le graphe élémentaire conflit E/S/S prend pour pénalités : 3,0 pour les deux communications du conflit S/S et 1,3 pour la communication restante au conflit E/E.

L'ordre de dominance entre les conflits se justifie par l'ordre d'apparition dans le chemin réseau des conflits. En effet, un conflit S/S se produit avant un conflit E/E. Le conflit E/S/S représente l'influence entre les conflits E/E et les conflits S/S. Finalement, comme pour *Gigabit Ethernet*, l'utilisation de liens full-duplex, implique une pénalité minimale et une dominance minimale des conflits E/S.

Exemples : La figure 5.6 propose des exemples de pénalités calculées suivant les deux modèles présentés précédemment.

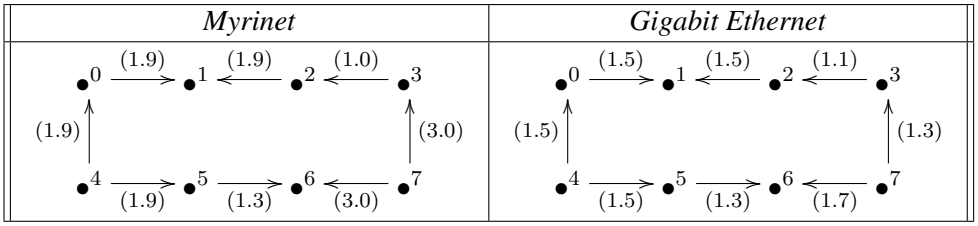


FIG. 5.6 – Exemples de pénalités prédites.

5.5.2.2 Généralisation à N processeurs

La généralisation des modèles pour des graphes comportant des nœuds de degré N, implique une modification du protocole expérimental du chapitre précédent. Cette modification consiste à dépasser la limite d'une tâche par processeur. De ce fait, chaque processeur exécute plusieurs

tâches de communication, permettant l'obtention de graphes avec des nœuds de degrés supérieurs à 2. Néanmoins, cette modification ajoute l'influence de l'ordonnanceur système dans la prise de performance. En étudiant la consommation de temps CPU des tâches communicantes, nous observons que cette consommation est inférieure à 10% pour *Gigabit Ethernet* et inférieure a 5% pour *Myrinet*. Dans le cas de *Myrinet*, pour obtenir une faible consommation de ressource CPU, il convient de choisir une option de remonter des événements réseaux par interruption (*blocking*) et non par scrutation (*polling*). Du fait de la faible consommation CPU, nous considérons comme négligeable l'influence de l'ordonnanceur système. En utilisant ce nouveau protocole expérimental, nous proposons dans le paragraphe suivant quelques expérimentations significatives.

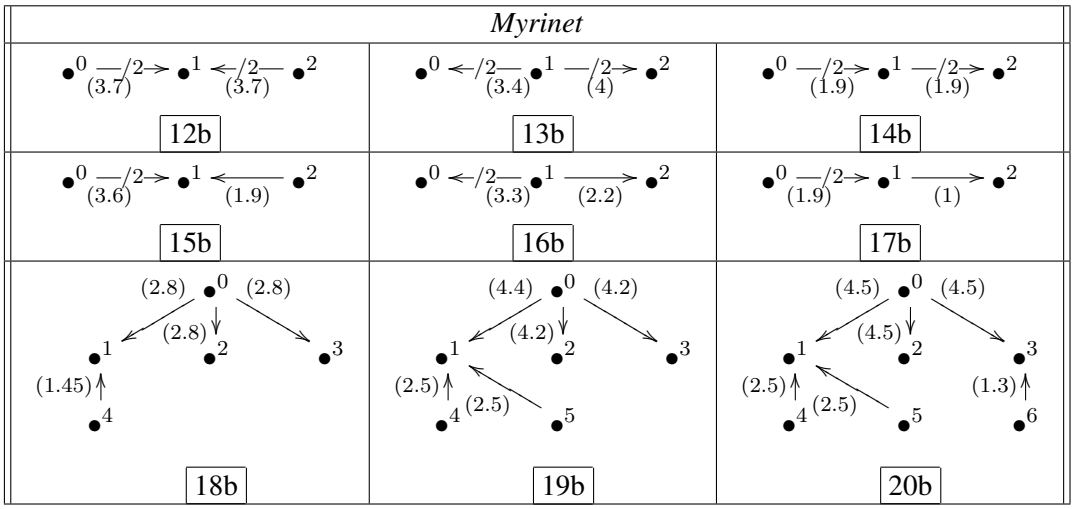


FIG. 5.7 – Conflits à N tâches pour *Myrinet*.

Expérimentations à N tâches *Myrinet*

Les expériences 12b, 14b, 15b et 17b de la figure 5.7 montrent une certaine régularité dans leurs pénalités par rapport aux expériences du chapitre précédent. En effet, par exemple l'indépendance entre conflits S/S et conflit E/S est toujours significative, schéma 14b et 17b. De même pour les schémas 12b et 15b, où les pénalités doublent en doublant le nombre de communications en conflit. Cependant, une irrégularité apparaît au niveau des schémas 13b et 16b. Les pénalités observées pour ces deux schémas ne correspondent pas aux pénalités précédemment rencontrées ni à un de leur multiple. Cette observation se confirme pour les schémas 18b, 19b et 20b. Ces schémas complexes font varier les pénalités en augmentant considérablement les pénalités du conflit S/S du nœud 0. Ainsi, en ajoutant des conflits E/E aux nœuds 1 et 3, il se produit une forte irrégularité. Toutefois, nous notons que pour le schéma 20b, quelque soit le type de conflit E/E aux nœuds 1 et 3, les trois communications du nœud 0 ont la même pénalité. Il semblerait que la pénalité la plus élevée s'applique indistinctement à chaque communication. D'autres schémas, non présentés dans ce document, nous ont conduit à conclure qu'il est difficile d'obtenir un modèle quantitatif pour *Myrinet*. Pour cette raison, nous avons opté dans l'étude approfondie du comportement du contrôle de flux de *Myrinet* afin de proposer un modèle descriptif dans la sous-section 5.5.3.

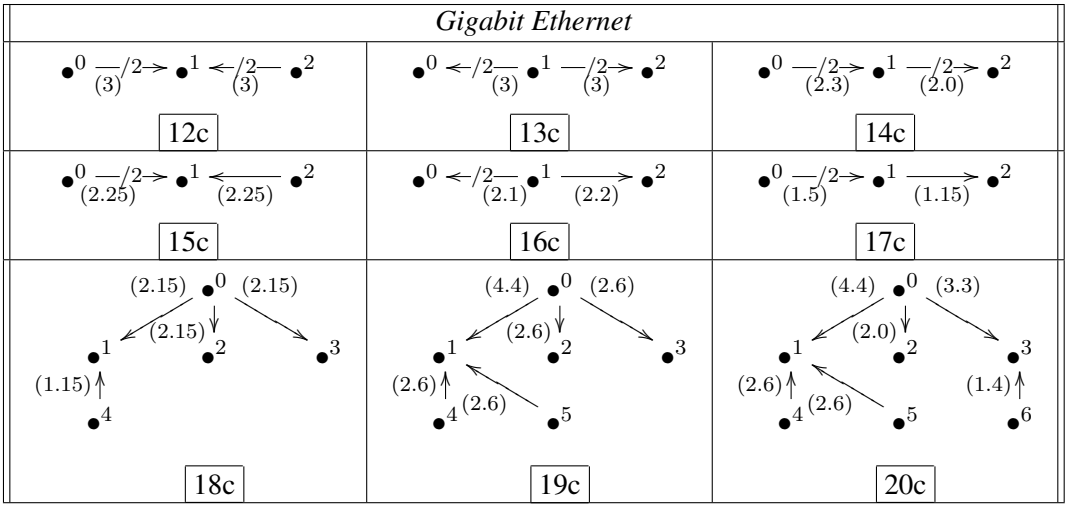


FIG. 5.8 – Conflits à N tâches pour *Gigabit Ethernet*.

Expérimentations à N tâches *Gigabit Ethernet*

A l'inverse de *Myrinet*, *Gigabit Ethernet* montre une régularité pour chaque schéma présenté dans la figure 5.8. En particulier, les pénalités sont fonction du nombre de communications, schéma 12c, 13c, 15c et 16c. Nous pouvons remarquer que pour ces expériences chaque communication ajoutée augmente de 0,75 la pénalité de l'ensemble des communications. Les pénalités du schéma 14c doublent par rapport au schéma 6c (figure 4.19). Le schéma 17c démontre de nouveau une indépendance entre conflit S/S et conflit E/S. Notons que lorsque le nombre de communications entrantes et le nombre de communications sortantes sont identiques dans un conflit E/S (schéma 14c), les pénalités appliquées sont les pénalités doublées d'un conflit E/S, et non les pénalités doublées d'un conflit S/S. Nous retrouverons ce point lors de l'évaluation des modèles dans le chapitre 7, sous-section 7.4.2. Finalement, si nous considérons des schémas complexes : 18c, 19c et 20c nous observons un déséquilibre entre les pénalités. En particulier pour le schéma 20c, chaque communication émettant du nœud 0 a une pénalité différente. Cette pénalité semble être proportionnelle au conflit E/E qu'elle subit en réception. De cette constatation et grâce à d'autres expérimentations, nous choisissons de garder une approche quantitative dans la modélisation du réseau *Gigabit Ethernet*.

5.5.2.3 Modèle quantitatif *Gigabit Ethernet* pour N processeurs

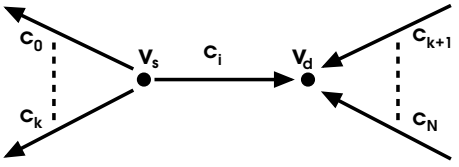


FIG. 5.9 – Schéma détaillé d'une communication.

Avant d'exposer le modèle quantitatif proposé pour *Gigabit Ethernet*, nous précisons quelques éléments nécessaires à la modélisation.

Considérons le schéma de la figure 5.9 représentant une situation quelconque dans un graphe. Ce schéma met en avant une communication c_i émettant du nœud v_s vers le nœud v_d. Cette communication est en conflit en émission avec les communications c_0 à c_k sur le nœud v_s. De plus cette communication est en conflit en réception sur le nœud v_d avec les communications c_{k+1} à c_n. Comme nous avons vu au début de ce chapitre, nous notons par $\Delta_s(v_s)$ le degré sortant du nœud v_s et par $\Delta_e(v_d)$ le degré entrant du nœud v_d.

Dans les expériences précédentes, nous avons constaté que le contrôle de flux provoque une répartition entre les valeurs des pénalités. Les conflits mettant en jeu un nombre important de communications provoquent des pénalités importantes. Ainsi, pour un conflit, nous définissons deux ensembles de communications : les communications fortement pénalisées et celles qui le sont beaucoup moins :

Définition 6 *Soit une communication c_i correspondant à un arc (v_s, v_d) d'un graphe G, ayant comme degrés $\Delta_s(i) = \Delta_s(v_s)$ et $\Delta_e(i) = \Delta_e(v_d)$. Soit C_s l'ensemble des communications ayant la même source que c_i dans le graphe G. Soit C_e l'ensemble des communications ayant la même destination que c_i dans ce même graphe G.*

Alors si $\Delta_e(i) = max\{\Delta_e(j), \forall j \mid c_j \in C_s\}$ nous dirons que c_i appartient à l'ensemble C_s^m des communications sortantes fortement ralenties du nœud s. Réciproquement, nous dirons que c_i appartient à l'ensemble des communications entrantes fortement ralenties C_e^m si $\Delta_s(i) = max\{\Delta_s(j), \forall j \mid c_j \in C_e\}$. □

Nous noterons par $card(C_e^m)$, réciproquement $card(C_s^m)$, le nombre de communications appartenant à l'ensemble C_e^m, réciproquement C_s^m.

Modèle quantitatif : pour déterminer la pénalité d'une communication c_i, nous calculons deux valeurs. Une première valeur représente la pénalité provoquée par le conflit en émission, notée p_s, alors que la seconde valeur correspond à la pénalité due au conflit en réception, notée p_e.

Nous calculons ces deux valeurs comme suit :

$$p_s = \begin{cases} 1 \text{ si } \Delta_s(i) = 1 \\ \begin{cases} \text{si } c_i \in C_s^m : \ \Delta_s(i) \times \beta \times (1 + \gamma_s(\Delta_s(i) - card(C_s^m))) \\ \text{sinon : } \ \Delta_s(i) \times \beta \times (1 - \gamma_s / card(C_s^m)) \end{cases} \end{cases}$$

$$p_e = \begin{cases} 1 \text{ si } \Delta_e(i) = 1 \\ \begin{cases} \text{si } c_i \in C_e^m : \ \Delta_e(i) \times \beta \times (1 + \gamma_e(\Delta_e(i) - card(C_e^m))) \\ \text{sinon : } \ \Delta_e(i) \times \beta \times (1 - \gamma_e / card(C_e^m)) \end{cases} \end{cases}$$

Finalement la pénalité associée à la communication c_i est :

$$p = max(p_s, p_e)$$

Le terme principal, impliquant le paramètre β, décrit la pénalité causée par le partage de ressource. Le second terme décrit la compensation entre pénalités des communications fortement

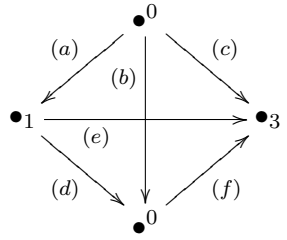
en conflit et celle moins ralenties. En effet, si une communication appartient à un ensemble C_e^m ou C_s^m, sa pénalité va augmenter et affaiblira les pénalités des communications de même source ou destination n'appartenant à un des deux ensembles. Les paramètres γ_s et γ_e correspondent au pourcentage de répartition en émission et réception. De la même manière, l'utilisation des paramètres $card(C_e^m)$ et $card(C_s^m)$ atténue cette répartition dans le cas de plusieurs communications fortement en conflit. Notons que $1 \geq \gamma_s \geq 0$ et $1 \geq \gamma_e \geq 0$, en revanche $\beta > 0$ peut être supérieur à 1, mais cela indiquerait de très mauvaises performances des composants réseaux.

Calcul des paramètres : Pour calculer le paramètre β, nous utilisons le conflit simple S/S. Nous évaluons les pénalités de ce conflit en augmentant le nombre de communications sortantes. Puis nous divisons les valeurs obtenues par le nombre de communications pour obtenir le paramètre β. A ce stade, une explication est nécessaire sur l'utilisation d'un paramètre unique β pour les calculs de p_s et p_e. En effet, supposons deux paramètres β_e et β_s correspondant à la spécification de β dans le cas entrant et sortant. L'évaluation précise de β_e en fonction du conflit E/E est difficile. Chaque communication du conflit E/E propose des valeurs proches mais légèrement différentes. Ce phénomène est dû à la variabilité intrinsèque de l'expérience : variabilité des dates de départ sur chaque nœud et variabilité au niveau du commutateur. En opposition, le conflit S/S propose des temps égaux, car la variabilité expérimentale est négligeable. De ce constat, nous supposons que β_s est le plus apte à correspondre au phénomène de partage des composants réseaux. Nous définissons un partage équivalent $\beta_s = \beta$ et $\beta_e = \beta_s$. A titre d'exemple, pour la grappe de Rennes β est égal à 0,71.

Pour évaluer les valeurs de γ_e et γ_s, nous utilisons le conflit de la figure 5.2 où chaque communication envoie la même quantité de données. Ce conflit a la particularité d'être symétrique suivant le nombre de communications entrantes et sortantes de ces nœuds. En effet, le nœud 0 comporte 3 communications sortantes, le nœud 3 a 3 communications entrantes, et les nœuds 1 et 2 ont respectivement 2 communications sortantes et 1 communication entrante, et respectivement 1 communication sortante et 2 communications entrantes. Suivant ce schéma, le paramètre γ_s influence fortement la communication (a) du nœud 0 et le paramètre γ_e la communication (f) du nœuds 3. Pour mesurer les valeurs de ces paramètres, nous évaluons le temps de chacune de ces deux communications suivant ce conflit. En se basant sur les temps des communications (a) et (f) nous en déduisons les valeurs de γ_s et γ_e. Nous obtenons $\gamma_s = 1 - t_a/(3 \times \beta \times t_{ref})$ et $\gamma_e = 1 - t_f/(3 \times \beta \times t_{ref})$ avec t_{ref} le temps nécessaire pour envoyer la même quantité de données que (a) et (f) mais sans concurrence. Il est intéressant de comparer les temps prédits et les temps mesurés pour toutes les communications du conflit, afin de vérifier les paramètres. La figure 5.2 montre cette vérification sur la grappe de Rennes en obtenant $\gamma_s = 0,115$ et $\gamma_e = 0,036$.

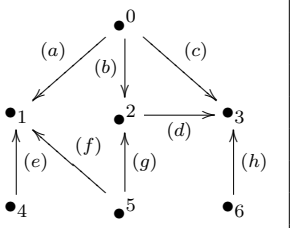
Exemples : A titre d'exemple numérique, nous considérons le graphe du tableau 5.3. Nous déterminons pour la grappe de Rennes les paramètres $\beta = 0,71$, $\gamma_s = 0,115$ et $\gamma_e = 0,036$. Le tableau 5.3 décrit le calcul détaillé de p_s, p_e et p.

Dans la sous-section suivante, nous proposons la généralisation du modèle *Myrinet*. Ce modèle utilise une approche descriptive qui se base sur le contrôle de flux de *Myrinet*. A la différence du modèle du réseau *Gigabit Ethernet*, ce modèle ne nécessite pas l'utilisation de paramètres qui doivent être au préalablement mesurés sur la grappe, à l'exception de la latence et la bande



Coms	T mesuré [s]	T prédit [s]
a	0.095	0.095
b	0.099	0.095
c	0.118	0.113
d	0.068	0.069
e	0.099	0.103
f	0.103	0.103

TAB. 5.2 – Vérification des paramètres, taille des communications de 4Mo, grappe de Rennes.



Coms	Δ_s	Δ_e	$card(C_s^m)$	$card(C_e^m)$	p_s	p_e	p
a	3	3	2 {a,c}	1 {a}	2.375	2.2866	2.375
b	3	2	2 {a,c}	1 {b}	2.13	1.4722	2.13
c	3	3	2 {a,c}	1 {c}	2.375	2.2866	2.375
d	1	3	-	1 {c}	1	2.0517	2.0517
e	1	3	-	1 {a}	1	2.0517	2.0517
f	2	3	1 {f}	1 {a}	1.5833	2.0517	2.0517
g	2	2	1 {f}	1 {b}	1.2567	1.3678	1.3678
h	1	3	-	1 {c}	1	2.0517	2.0517

TAB. 5.3 – Exemple d'application du modèle quantitatif du modèle *Gigabit Ethernet* généralisé.

passante.

5.5.3 Approche descriptive

Les contrôles de flux des modèles *Myrinet* et *Quadrics* ont la particularité de mettre en évidence des mécanismes d'une complexité réduite. Cette réduction provient à la fois de la fiabilité des composants réseaux et de la recherche de communication à faible latence. Par conséquent, une modélisation directe est possible pour le comportement de ces contrôles de flux. Dans cette section, nous présentons un modèle descriptif pour le réseau *Myrinet* et ouvrons sur la possibilité de définir un modèle pour *Quadrics*. Pour chaque réseau, nous présentons plus en détail le mécanisme de contrôle de flux avant de décrire le modèle lui-même.

5.5.3.1 *Myrinet*

Détail du contrôle de flux du réseau *Myrinet* : *Myrinet* propose un contrôle de flux basé sur le protocole *Stop&Go*. Ce protocole consiste en l'envoi d'un symbole STOP à une source si celle-ci provoque de la contention. Une source recevant un symbole STOP arrête l'émission des données jusqu'à l'obtention d'un symbole GO qui provoque le redémarrage de l'émission. Il s'agit donc du nœud en réception qui bloque les communications des nœuds en émission. Cependant, lorsque deux communications tentent d'acquérir simultanément la carte réseau, un ordre *First In First Out* (*FIFO*) se créé. Lorsqu'une communication qui a acquis la carte réseau (première arrivée) reçoit un symbole STOP, elle arrête son émission et ne permet pas à une autre communication, dans

la file d'attente *FIFO*, d'utiliser la carte pour transférer des données. En d'autres termes, il n'y a pas d'ordonnancement des communications, et une communication bloquée ralentit toutes les communications suivantes dans la file *FIFO*. Cet effet, s'explique par la simplicité du protocole *Stop&Go* qui ne décrit pas quelle partie du chemin réseau est congestionnée. Le cas le plus défavorable est considéré, c'est-à-dire le lien sortant de la carte réseau est congestionné permettant à aucune communication d'émettre. En se basant sur ces considérations, nous proposons dans le paragraphe suivant un modèle descriptif de la gestion de la concurrence par le réseau *Myrinet*.

Modèle descriptif *Myrinet* : le modèle descriptif que nous présentons dans ce paragraphe se base sur le comportement du contrôle de flux de données expliqué précédemment. Pour calculer les pénalités nous utiliserons une approche algorithmique basée sur les différentes possibilités des états des communications. En effet, nous définissons deux états possibles pour une communication. L'état "envoi" correspond à une communication qui envoie des données sur le réseau et l'état "attend" correspond à une communication qui est en attente d'envoi. En utilisant ces deux états, nous cherchons à déterminer toutes les combinaisons possibles des états des communications d'un graphe. Pour ce faire, nous utilisons une seule règle :

– Lorsqu'une communication est dans l'état "envoi", chaque communication ayant le même nœud source ou le même nœud destinataire prend l'état "attend".

Cette règle s'explique par les propriétés du contrôle de flux décrit précédemment. En effet, si une communication acquiert la carte pour envoyer des données, toutes les communications du même nœud voulant émettre doivent attendre d'acquérir la carte réseau. De même, si un nœud en réception est congestionné, le contrôle de flux choisi une communication qui continue à émettre et arrête toutes les autres communications entrantes.

Une fois les ensembles d'états déterminés, nous calculons le coefficient d'émission défini comme le nombre des états "envoi" pour une communication. Ensuite, nous analysons l'ensemble des coefficients d'émission des communications sortantes d'un nœud, et nous associons à chacune de ces communications le coefficient d'émission minimal. En effet, nous considérons le pire cas pour lequel chaque communication, sortante d'un même nœud, est autant ralentie que la communication la plus lente. Cela se justifie en considérant que chaque communication acquiert de façon équitable la carte réseau.

Finalement, la pénalité d'une communication est calculée en divisant le nombre des ensembles d'états par le coefficient d'émission. En d'autres termes, une pénalité correspond à l'inverse de la moyenne du nombre d'états d'émissions de la communication sortante la plus ralentie du même nœud.

Pour expliquer plus en détail le calcul des pénalités, nous présentons dans le paragraphe suivant un exemple basé sur un graphe élaboré.

Exemples : l'exemple que nous avons choisi d'étudier est présenté dans la figure 5.10 schéma 21b. Il fait intervenir 5 nœuds et 6 communications. En appliquant la règle précédente, nous déterminons l'ensemble des états possibles des communications. Ces états sont représentés graphiquement dans la figure 5.10 du schéma 22b au schéma 26b. Sur ces schémas, une communication qui est dans l'état envoi correspond à une flèche pleine et une communication dans un état attente est dessinée par une flèche en pointillé. De la même manière, le tableau 5.4 décrit ces ensembles

d'états en associant la valeur 1 à un état en envoi et la valeur 0 pour un état en attente. Par exemple, dans le schéma 22b, si la communication (a) émet, les communications (b) et (c) sont en attente pour acquérir la carte réseau. Lorsque cette communication arrive à son nœud destinataire (nœud 1) le contrôle de flux bloque les communications (d) et (e). Finalement, la communication (f) ne subit aucune influence et peut librement émettre ses données.

Il existe pour ce schéma 5 combinaisons possibles entre les états des communications. Les communications (a), (b) et (c) prennent un coefficient d'émission valant 1, causé par la communication (a). De la même façon, les communications (f) et (e) ont un coefficient d'émission valant 2, ainsi que la communication (d). En utilisant ces valeurs, nous calculons les pénalités dans le tableau 5.4.

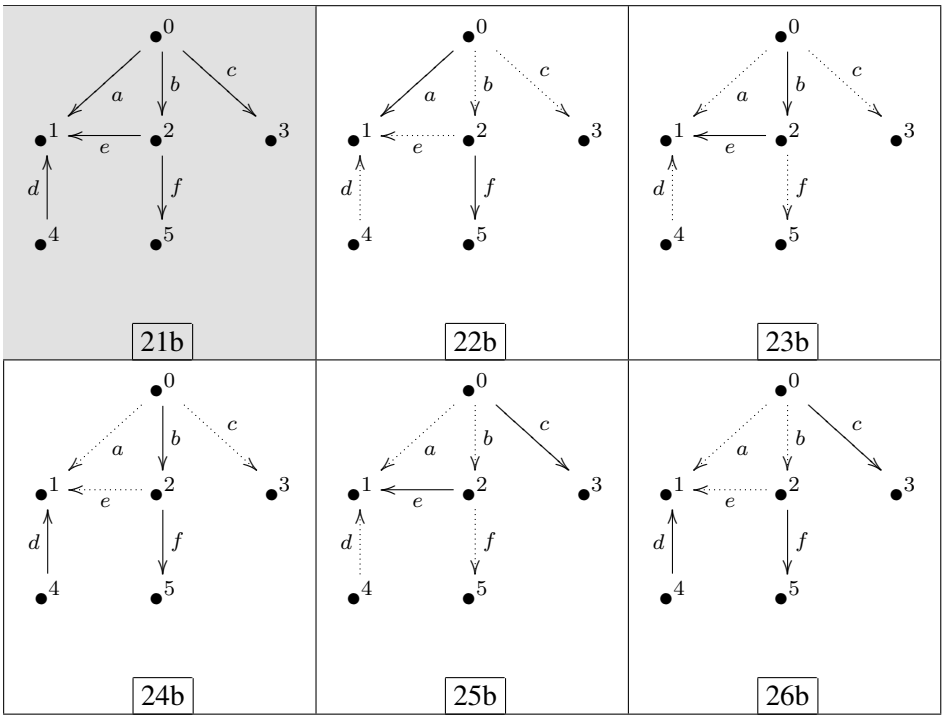


FIG. 5.10 – Exemple des différents ensembles d'états des communications.

5.5.3.2 *Quadrics*

L'analyse sur le réseau *Quadrics* s'articule autour de deux axes : l'étude du mécanisme de contrôle de flux et l'évolution des pénalités lors de l'augmentation de conflits simples.

Détail du contrôle de flux du réseau *Quadrics* : le contrôle de flux de *Quadrics* est plus complexe que celui de *Myrinet*. *Quadrics* utilise un mécanisme basé sur deux aspects :

– la fragmentation des messages en pipeline de petites cellules (*flits* d'une taille de 16o) ou *wormhole routing* [29] ;

	Communications					
	a	b	c	d	e	f
Etats du schéma 22b	1	0	0	0	0	1
Etats du schéma 23b	0	1	0	0	1	0
Etats du schéma 24b	0	1	0	1	0	1
Etats du schéma 25b	0	0	1	0	1	0
Etats du schéma 26b	0	0	1	1	0	1
Somme	1	2	2	2	2	3
Minimum	1	1	1	2	2	2
Pénalités	5	5	5	2.5	2.5	2.5

TAB. 5.4 – Exemples du calcul des pénalités.

– l'utilisation de canaux virtuels (*virtual channel*) [99] ;

Le routage de type *wormhole* décompose un message en un pipeline de cellules. Lorsque deux pipelines partagent une même partie du chemin réseau, un des deux pipelines est stoppé. Le pipeline stoppé est réparti entre les différents tampons du chemin réseau qu'il a emprunté de sa source jusqu'au composant où il a été stoppé. Par conséquent, il occupe toute une partie du chemin réseau et empêche un autre pipeline d'en utiliser les composants. Cette situation peut être génératrice de blocage du réseau (*deadlock*) entre plusieurs pipelines.

Pour éviter les situations de blocage, les concepteurs de *Quadrics* font appel à la technologie des canaux virtuels [40][73]. Les canaux virtuels permettent de diviser les différents tampons physiques des composants réseaux entre plusieurs pipelines. Ce découpage évite l'apparition de blocage du réseau, car elle autorise plusieurs pipelines à emprunter simultanément une portion identique d'un chemin réseau. Néanmoins, un tel découpage entraîne une baisse des performances lorsque le nombre de canaux virtuels est grand d'autant plus que les liens physiques ne sont pas divisibles entre les pipelines et qu'un ordre de passage est obligatoire. *Quadrics* utilise deux canaux virtuels par lien half-duplex (2 canaux pour la réception et 2 canaux pour l'émission).

Certaines caractéristiques de ces mécanismes peuvent influer de manière significatif sur la définition d'un modèle pour *Quadrics*. Le routage *wormhole* est responsable du blocage des chemins réseaux. Même si les canaux virtuels évitent les blocages réseaux, un tel mécanisme peut avoir comme effet des temps d'attente importants lors de plusieurs communications concurrentes. En découpant les tampons mémoires des composants réseaux, les canaux virtuels réduisent la bande passante disponible pour chaque communication.

Evolution des pénalités : dans ce paragraphe, nous donnons quelques éléments susceptibles de conduire à une modélisation du réseau *Quadrics*. Pour ce faire, nous modélisons le comportement de *Quadrics* sur des conflits simples mais comportant plusieurs communications. Comme il nous est possible d'accéder à des machines ayant 16 processeurs, nous pouvons créer des conflits avec plusieurs communications ayant la même source et la même destination. Nous appelons par groupe de communications un tel ensemble de communications. Avec des machines à 16 processeurs, la taille maximale d'un groupe de communications est de 8 communications. Les conflits simples entre groupe de communications offrent un premier aperçu de la façon dont *Quadrics* gère

la concurrence. Nous reprenons les conflits S/S, E/E et E/S en ajoutant deux groupes de communications, mais toujours avec seulement 3 nœuds. Un groupe de communications est noté Gr avec une cardinalité notée $card(Gr)$ et une communication d'un groupe est notée c_i. Nous obtenons les conflits suivants :

- Conflit S/S(Gr_1,Gr_2) : un nœud source émet un groupe de $card(Gr_1)$ communications vers un nœud et un autre groupe de $card(Gr_2)$ communications vers un autre nœud ;
- Conflit E/E(Gr_1,Gr_2) : un nœud destination reçoit $card(Gr_1)$ communications provenant d'un même nœud source et $card(Gr_2)$ communications provenant d'une autre nœud source ;
- Conflit E/S(Gr_1,Gr_2) : un nœud reçoit $card(Gr_1)$ communications ayant la même source et émet $card(Gr_2)$ communications vers une même destination.

Toutes les communications démarrent en même temps. Quelques-uns des résultats de ces expériences sont rappelés dans la figure 5.11. Pour plus de précisions, le lecteur peut se référer aux schémas 1a à 12a de la figure 4.17 du chapitre 4, sous-section 4.3.4.

Quadrics		
$\bullet^0 \underset{(4)}{\overset{}{\text{—}/2\text{→}}} \bullet^1 \underset{(4)}{\overset{}{\text{←}/2\text{—}}} \bullet^2$ 22a	$\bullet^0 \underset{(4)}{\overset{}{\text{←}/2\text{—}}} \bullet^1 \underset{(4)}{\overset{}{\text{—}/2\text{→}}} \bullet^2$ 23a	$\bullet^0 \underset{(3)}{\overset{}{\text{—}/2\text{→}}} \bullet^1 \underset{(6)}{\overset{}{\text{—}/2\text{→}}} \bullet^2$ 24a
$\bullet^0 \underset{(8)}{\overset{}{\text{—}/4\text{→}}} \bullet^1 \underset{(4)}{\overset{}{\text{←}/2\text{—}}} \bullet^2$ 25a	$\bullet^0 \underset{(6)}{\overset{}{\text{←}/4\text{—}}} \bullet^1 \underset{(6)}{\overset{}{\text{—}/2\text{→}}} \bullet^2$ 26a	$\bullet^0 \underset{(6)}{\overset{}{\text{—}/4\text{→}}} \bullet^1 \underset{(6)}{\overset{}{\text{—}/2\text{→}}} \bullet^2$ 27a
$\bullet^0 \underset{(4)}{\overset{}{\text{—}/2\text{→}}} \bullet^1 \underset{(12)}{\overset{}{\text{←}/6\text{—}}} \bullet^2$ 28a	$\bullet^0 \underset{(8)}{\overset{}{\text{←}/2\text{—}}} \bullet^1 \underset{(8)}{\overset{}{\text{—}/6\text{→}}} \bullet^2$ 29a	$\bullet^0 \underset{(3)}{\overset{}{\text{—}/2\text{→}}} \bullet^1 \underset{(18)}{\overset{}{\text{—}/6\text{→}}} \bullet^2$ 30a

FIG. 5.11 – Conflits simples augmentés pour *Quadrics*.

Pour chaque conflit, nous proposons un modèle simple qui caractérise la répartition de la bande passante. Ces modèles découlent directement de ces valeurs observées suivant différentes tailles de groupes. Ces modèles sont donc des modèles quantitatifs, mais nous tentons de mettre en évidence leurs corrélations avec le comportement du contrôle de flux décrit préalablement.

Conflit S/S(Gr_1,Gr_2) :

Dans le cas de ce conflit, les pénalités observées suivent l'équation suivante :

$$\forall c_i \ \in Gr_1, \ \text{ ou } \ c_i \ \in Gr_2, \ \ p_{c_i} = card(Gr_1) + card(Gr_2)$$

Pour ce modèle, il semble que le nœud émetteur ne fait pas de distinction entre les groupes et alloue les deux canaux virtuels pour chaque communication. Ainsi, chaque communication acquiert la ressource réseau de façon équitable. Comme cette acquisition est exclusive (pas d'ordonnancement entre communication), chaque communication est retardée d'un temps fonction du nombre de communications et donc de la somme des cardinalités des groupes.

Conflit E/E(Gr_1,Gr_2) :
L'équation suivante représente les pénalités pour ce conflit :

$$\begin{cases} \forall c_i \in Gr_1, & p_{c_i} = 2 \times card(Gr_1) \\ \forall c_i \in Gr_2, & p_{c_i} = 2 \times card(Gr_2) \end{cases}$$

A la différence du conflit précédent, nous observons deux phénomènes : la pénalité d'une communication d'un groupe est uniquement fonction de la cardinalité de son groupe et les pénalités sont multipliées par deux. Il est difficile de mettre en place des expériences particulières pour relier ces deux phénomènes aux mécanismes de contrôle de flux. D'autant plus que les cartes *Quadrics* ne permettent pas de remonter certaines informations bas-niveau qui pourraient être observées par les valeurs de certains compteurs matériels des cartes. Toutefois, nous pouvons émettre quelques suppositions raisonnables. Au niveau du nœud source, nous supposons comme précédemment qu'une communication sortante acquiert les 2 canaux virtuels. Si nous supposons que le nœud récepteur alloue aussi les deux canaux virtuels par communication, nous devrions obtenir la même équation que pour le conflit précédent, or ce n'est pas le cas. Cependant, si le nœud récepteur alloue qu'un seul canal par groupe, il concentre les deux canaux d'une communication provenant d'un groupe vers un seul canal virtuel. D'une part, un effet direct sur les pénalités serait une multiplication par deux (car la bande passante serait réduite de moitié). D'autre part, chaque pénalité serait fonction uniquement du nombre de communications du groupe auquel elle appartient, car l'utilisation des deux canaux séparent les influences entre les deux groupes. Au regard des résultats obtenus sur les expériences de ce conflit, l'influence du contrôle de flux sur les valeurs des pénalités semble probable.

Conflit E/S(Gr_1,Gr_2) :
Les valeurs observées sur le dernier conflit se décrivent par :

$$\begin{cases} \forall c_i \in Gr_1, & p_{c_i} = \frac{3}{2} \times card(Gr_1) \\ \forall c_i \in Gr_2, & p_{c_i} = 3 \times card(Gr_2) \end{cases}$$

Il résulte de ce conflit des valeurs bien particulières. En se basant sur le contrôle de flux, il est difficile d'émettre des suppositions expliquant l'apparition des facteurs multiplicateurs $\frac{3}{2}$ et 3. En effet, pour ce conflit, un réseau avec des liens full-duplex ne devrait pas ajouter de facteurs multiplicateurs aux pénalités. Chaque communication devrait avoir une pénalité égale à la cardinalité de son groupe. C'est le cas pour les réseaux *Myrinet* et *Gigabit Ethernet* utilisant également des liens full-duplex. Il est possible de considérer que ces facteurs multiplicateurs sont causés au niveau logiciel et non par le contrôle de flux bas-niveau. Pour mettre en évidence l'influence logiciel, il faudrait mener de plus amples expérimentations. Toutefois, la recherche des causes de ces phénomènes est limitée par plusieurs raisons. Premièrement, il conviendrait d'accéder à une autre grappe utilisant le réseau *Quadrics* pour tester la reproductibilité de ces phénomènes. Cependant, il ne nous a pas été possible d'accéder à une grappe contenant plus de deux nœuds homogènes reliés par un réseau *Quadrics*. Deuxièmement, pour mettre en évidence l'impact logiciel, il faudrait pouvoir faire varier l'implantation *MPI* utilisée. Il ne nous a pas été possible de réaliser ce type d'expérience.

Ce manque d'expérience a limité à la fois la définition d'un modèle descriptif que la définition d'un modèle quantitatif. Nous rappelons qu'à titre d'exemple s'agissant de pénalités observées sur des schémas plus complexes, le lecteur peut se référer aux expériences du chapitre précédent : figure 4.17, par exemple, schémas 13a, 14a ou 20a.

5.6 Exemple numérique

Avant de présenter un exemple complet du calcul des effets de la concurrence, nous résumons les points importants qui sont à l'origine des conflits.

- Le squelette logique des communications de l'application, c'est-à-dire les origines et les destinations des communications correspondent aux rangs des tâches *MPI*, et leur taille de message ;
- La structure de la grappe de calcul : nombre de nœuds et type de réseaux ;
- Une fois connues la structure logique et les caractéristiques de la grappe, le placement des tâches sur les nœuds permet de déterminer le squelette physique des communications, c'est-à-dire, que dans ce cas les origines et destinations des communications correspondent aux nœuds des grappes.
- Les caractéristiques des communications (date de départ, taille, origine et destination) et le type de réseau (*Myrinet* ou *Gigabit Ethernet*) nous permettent d'appliquer les modèles précédents afin de déterminer à la fois l'évolution des conflits (suite de graphes G_t) et les temps des communications.

Dans cette section, nous présentons un exemple complet en reprenant les formalismes étudiés dans la sous-section 5.3.3.

5.6.1 Grappe, application et placement

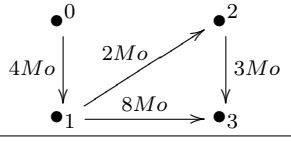
Pour plus de simplicité, nous utiliserons de nouveau l'application décrite par la figure 5.1 avec la grappe $A(4 \times 2)$. Une telle grappe est constituée de quatre nœuds biprocesseurs. Nous ajoutons à cette grappe un réseau *Myrinet* qui propose une bande passante effective de 223 Mo/s et une latence de 4 μs. Ces valeurs sont obtenues par des mesures réalisées sur une grappe réelle en utilisant *Mpich*. Nous modifions l'application de la figure 5.1 en remplaçant le temps de terminaison des communications par les tailles de messages. Cette nouvelle application est présentée dans la figure 5.12.

$E_1(d=2, T_d=0.0, m=4Mo)$	$R_2(s=1)$
$E_3(d=4, T_d=0.01, m=8Mo)$	$R_4(s=3)$
$E_5(d=6, T_d=0.02, m=2Mo)$	$R_6(s=5)$
$E_7(d=8, T_d=0.03, m=3Mo)$	$R_8(s=7)$

FIG. 5.12 – Exemple d'un ensemble de 8 tâches communicantes.

Il convient maintenant de définir un placement des tâches sur les nœuds de la grappe. Nous choisissons un placement provoquant des conflits complexes. Ce placement est résumé dans le

tableau 5.5. En outre, ce tableau contient le graphe de placement, c'est-à-dire si toutes les communications avaient la même date de départ.

		Placement	Graphe de placement
Nœuds	0	E_1	
	1	R_2,E_3,E_5	
	2	R_6,E_7	
	3	R_4,R_8	

TAB. 5.5 – Exemple de placement et de son graphe associé.

5.6.2 Calcul des temps de communications

En partant du placement physique obtenu précédemment, nous allons dans cette section détailler le calcul des temps de communications en fonction des conflits et des pénalités. Nous utilisons le modèle de répartition de la bande passante proposé pour *Myrinet*. Le tableau 5.6 récapitule les différentes étapes dans l'évolution des conflits.

Durant la première étape, seule la communication (a) émet des données pendant une période de 0.01 seconde. Après ce laps de temps, la communication (b) commence à émettre. Cependant, le conflit généré par ces deux communications ne provoque aucun surcoût. L'événement suivant est la terminaison de la communication (a). En effet, la communication (c) ne débute qu'à partir de la date globale égale à 0.02, or la communication (a) se termine au temps 0.01776. Notons que la communication (a) n'a pas été retardée durant son émission, elle a, soit émis seule, soit subi une pénalité de 1. L'étape 3 correspond au moment après la fin de la communication (a) et avant le début de la communication (c). Dans cette étape, seule (b) émet durant une période de 0.002224 et envoi 524938 octets. A l'étape 4, la communication (b) a une durée de 0.01 mais a débutée à la date 0.01, la date globale actuelle est de 0.02. Cette date correspond au départ de (c). Les communications (b) et (c) émettent en concurrence suivant une pénalité de 2 pour une durée de 0.01. Notons qu'avec une pénalité de 2, ces deux communications émettent deux fois moins de données durant cette période. Au temps global 0.03, la communication (d) commence à émettre. A ce point les trois communications ont une pénalité de 2. La communication (c) est la première à se terminer. Au final, cette communication a la même durée que la communication (a) mais a envoyé deux fois moins de données. Ceci est un exemple de l'influence des autres communications qui ont provoqué une pénalité de 2 durant toute l'émission de (c). Cet exemple se termine avec la fin de la communication (d) à l'étape 6 et la fin de la communication (b) à l'étape 7.

Nous conclurons cet exemple en montrant la variabilité des temps obtenus. Par exemple, la communication (b) transite entre plusieurs pénalités durant son émission. Ces différentes transitions déterminent la durée de cette communication qui n'est pas triviale à calculer.

5.7 Conclusion

Ce chapitre illustre la modélisation des comportements réseaux. Cette modélisation comporte deux étapes :

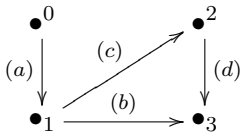
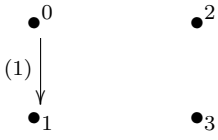
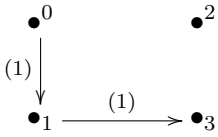
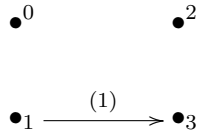
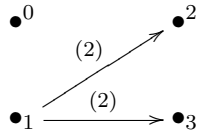
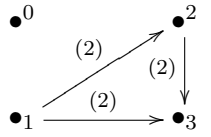
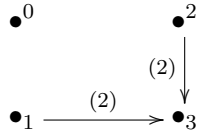
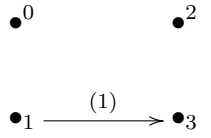
– une étape qui se consacre à l'application du modèle précédent suivant l'évolution des conflits dans le temps par une simulation à événements discrets ;
– une étape qui modélise la répartition de la bande passante entre les communications, c'est-à-dire déterminer les pénalités induites par les conflits.

Un point important est l'indépendance entre ces deux étapes. Il est possible de modifier un modèle de répartition de la bande passante ou d'en définir un nouveau sans changer le modèle de la première étape.

Les modèles de répartition ou du calcul des pénalités peuvent être basés sur deux principes :
– une description du comportement du réseau induit par le contrôle de flux sous-jacent ;
– une évaluation des pénalités par une équation dépendante de paramètres mesurés sur le réseau.

Par exemple, le modèle *Myrinet* basé sur une description du réseau ne nécessite aucun paramètre autre que la latence et la bande passante. En l'inverse, le modèle *Gigabit Ethernet* nécessite la mesure de trois paramètres supplémentaires.

Dans le chapitre 7, nous évaluerons l'efficacité de ces modèles face à des applications existantes. Avant leurs évaluations, nous allons déterminer les impacts des différents composants réseaux sur la concurrence, et par conséquent, sur les modèles.

		Coms	Temps de départ [s]	Taille [o]
		a	0	4194304
		b	0.01	8388608
		c	0.02	2097152
		d	0.03	3145728
	Etape 1 Ev. suivant : début de (b)	Coms	Taille envoyée	Durée Totale
		a	2360330	0.010
		b	-	-
		c	-	-
		d	-	-
	Etape 2 Ev. suivant : fin de (a)	Coms	Taille envoyée	Durée Totale
		a	1835390	0.017776
		b	1835390	0.007776
		c	-	-
		d	-	-
	Etape 3 Ev. suivant : début de (c)	Coms	Taille envoyée	Durée Totale
		a	-	0.017776
		b	524938	0.010
		c	-	-
		d	-	-
	Etape 4 Ev. suivant : début de (d)	Coms	Taille envoyée	Durée Totale
		a	-	0.017776
		b	1180160	0.020
		c	1180160	0.010
		d	-	-
	Etape 5 Ev. suivant : fin de (c)	Coms	Taille envoyée	Durée Totale
		a	-	0.017776
		b	917696	0.027776
		c	917696	0.017776
		d	917696	0.007776
	Etape 6 Ev. suivant : fin de (d)	Coms	Taille envoyée	Durée Totale
		a	-	0.017776
		b	2228740	0.046661
		c	-	0.017776
		d	2228740	0.026661
	Etape 7 Ev. suivant : fin de (b)	Coms	Taille envoyée	Durée Totale
		a	-	0.017776
		b	1703090	0.053876
		c	-	0.017776
		d	-	0.026661

TAB. 5.6 – Exemple numérique du calcul des temps de communications en conflits.

III

Evaluation des modèles de communications concurrentes

La définition des modèles réseaux donne lieu à une meilleure compréhension des phénomènes concurrents. Les modèles sont capables de prédire les temps de communications en concurrences quel que soit les conditions nécessaires à l'exécution de l'application. Pour caractériser cette capacité, il est important d'explorer deux points : déterminer les influences matérielles et logicielles d'une grappe sur les prédictions et d'évaluer les prédictions des modèles.

Cette dernière partie se focalise sur les deux points énoncés précédemment. Il a été montré dans la première partie de ce document, que de nombreux paramètres influent sur les performances des applications. Ainsi, dans le premier chapitre de cette partie, nous proposons des expériences montrant l'influence de certains paramètres sur les performances et le comportement concurrent des processus. Pour ce faire, soit nous isolerons par de nouvelles expériences certains composants réseaux, soit nous modifierons un ensemble de paramètres des expériences présentées dans le chapitre 4. Le dernier chapitre de ce document présente une évaluation des prédictions des modèles réseaux. Cette évaluation présente l'erreur des modèles provenant de la confrontation entre valeurs prédites et mesurées. Les contextes de validation, correspondant aux applications et grappes utilisées, sont multiples.

Paramètres techniques influençant les communications concurrentes 6

6.1 Introduction

Ce premier chapitre de cette partie est dédié à l'étude des paramètres susceptibles d'influencer le comportement concurrent. En effet, le choix de certaines caractéristiques matérielles ou logicielles du contexte expérimental peut avoir un impact sur les valeurs des pénalités obtenues. Plusieurs questions se posent :

- Il existe plusieurs fabricants de commutateurs *Gigabit Ethernet*. Quelle est l'influence de leurs choix de fabrication sur les performances du réseau *Gigabit Ethernet* ?
- Quelle proportion des pénalités est due au partage du bus *PCI* ?
- Comment se comportent des communications concurrentes n'utilisant pas une interface *MPI*, mais une librairie de plus bas-niveau ?
- L'ajout d'une carte réseau diminue-t-il par deux les pénalités ?

Pour répondre à ces différentes questions, nous présentons plusieurs expériences décrites suivant un tryptique : objectif, contexte expérimental et résultats obtenus. Les expériences de ce chapitre se découpent en deux parties. Une première partie 6.2 étudie les résultats obtenus en modifiant certains composants matériel réseau. Une seconde partie 6.3 se consacre à l'étude des influences logicielles. Ce chapitre se termine par une discussion sur l'importance de certains paramètres et leurs incidences sur les modèles du chapitre précédent.

Nous rappelons que la description matérielle des grappes est précisée dans l'annexe A.

6.2 Influence matérielle

Cette première section étudie l'influence des composants matériels sur les pénalités. Elle est constituée de cinq sous-parties dédiées respectivement au bus *PCI*, au commutateur, à l'utilisation de deux cartes réseaux, à l'interaction des communications intra-nœuds et inter-nœuds, et finalement à l'étude du comportement de communications intra-nœuds sur une machine *NUMA*.

6.2.1 Bus *PCI*

Le bus *PCI* est un des premiers composants utilisés par les messages sur le chemin réseau d'une communication. Ce composant est interne au nœud.

Objectif : L'objectif des expériences de cette sous-section est de déterminer le coût créé par le partage du bus *PCI* entre communications concurrentes. Ce coût sera évalué en terme de temps de communication pour trois catégories de taille de messages :

- messages de petite taille 1ko ;
- messages de taille moyenne 64Ko ;
- messages de grande taille 10Mo.

Contexte expérimental : Nous nous basons sur les expériences du chapitre 4. Pour réaliser cette étude, nous utilisons la grappe de Lille. La grappe de Lille a l'avantage de disposer de deux cartes *Gigabit Ethernet* par nœud. Une carte est utilisée pour le réseau de calcul et une autre carte pour le réseau d'administration. Le réseau de calcul et le réseau d'administration comportent chacun un commutateur identique (même fabricant et version). Le routage est disjoint entre les deux réseaux, c'est-à-dire que la carte du réseau de calcul n'émet pas de messages vers le commutateur du réseau d'administration et vice-versa. En configurant le système d'exploitation de chaque nœud afin de pouvoir utiliser simultanément les deux cartes réseaux, nous réalisons des expériences mettant en évidence l'influence du bus *PCI*. En effet, en reprenant les conflits élémentaires E/E, S/S et E/S et en affectant une communication par carte, celles-ci se retrouvent en conflit qu'au niveau du bus *PCI*. Comme exemple, la figure 6.1 schématise l'expérience obtenue pour un conflit E/E. Notons que le bus *PCI* étudié est un bus *PCI-X* 64 bits.

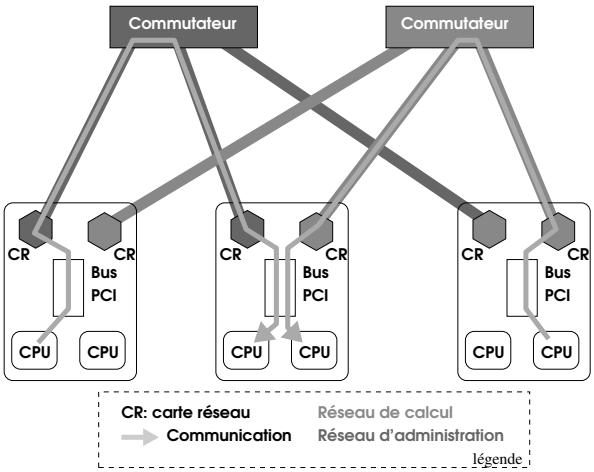


FIG. 6.1 – Conflit E/E utilisant deux réseaux disjoints.

Résultats : Les temps de communication obtenus pour chacune des expériences sont présentés dans le tableau 6.1. Ce tableau rappelle les valeurs observées en utilisant un seul réseau et propose

		1 seul réseau		2 réseaux	
		com 1	com 2	com 1	com 2
Sans conflit	1ko [μs]	5.2		-	
	64Ko [ms]	62		-	
	10Mo [s]	0.118		-	
Conflit E/E	1ko [μs]	4.9	5	5.2	4.9
	64Ko [ms]	95	96	74	74
	10Mo [s]	0.195	0.195	0.128	0.128
Conflit S/S	1ko [μs]	5	5	5.2	5.5
	64Ko [ms]	101	101	70	70
	10Mo [s]	0.195	0.195	0.122	0.122
Conflit E/S	1ko [μs]	5	5.6	5.3	5.8
	64Ko [ms]	75	74	64	71
	10Mo [s]	0.134	0.131	0.123	0.126

TAB. 6.1 – Influence du bus *PCI* sur les temps de communication en conflit.

les valeurs obtenues avec deux réseaux. Il rappelle également les valeurs mesurées sans conflits, c'est-à-dire avec une seule communication.

Concernant le bus *PCI*, nous remarquons que le partage n'a un effet significatif que pour des messages de taille moyenne ou de grande taille. Le coût ajouté aux conflits utilisant deux cartes par rapport au coût de la communication sans conflit varie entre 10% et 20%. Il existe une légère différence entre le conflit E/E et S/S. Le conflit S/S semble subir un ralentissement plus faible que le conflit E/E. Ce phénomène peut s'expliquer par le fait qu'un conflit S/S apporte une plus grande stabilité au système quant à la gestion des événements de communication. De ce fait, les communications des deux cartes sont traitées de façon plus homogène. Par conséquent, la gestion par le système des deux cartes est plus coûteux en réception qu'en émission. Toutefois, il est difficile de déterminer si les valeurs observées sont effectivement dues au partage du bus *PCI* ou à la gestion des deux cartes par le système d'exploitation. Néanmoins, nous concluons que le bus *PCI* a une influence limitée sur les pénalités associées aux communications concurrentes.

Il est intéressant d'observer que la grappe de Lille a des pénalités très proches à la grappe de Rennes pour les trois conflits. Pourtant, les commutateurs des deux grappes sont différents. Dans la sous-section suivante, nous approfondirons les modifications causées par le changement de commutateur.

6.2.2 Commutateur

Les commutateurs sont d'autres composants dont il convient d'étudier l'influence. Cette étude est intéressante dans le cas du réseau *Gigabit Ethernet* dont les commutateurs sont conçus par différents fabricants.

Objectif : Pour caractériser l'influence du commutateur sur les pénalités des communications concurrentes, nous menons les expériences complexes du chapitre 4 sur plusieurs grappes. Ces

grappes sont comparables en termes de nœud mais possèdent des commutateurs différents. L'objectif comme précédemment est de comparer les pénalités entre les expériences.

Contexte expérimental : Nous comparerons quatre grappes deux à deux. Les grappes de Lille et Sophia sont assez similaires, tant du point de vue processeur (même fréquence), que des composants entrée/sortie (bus *PCI*, carte réseau). Cependant, leurs commutateurs diffèrent. La grappe de Lille comporte un commutateur dédié au calcul haute performance alors que le commutateur de la grappe de Sophia est conçu sans optimisation spécifique. Les nœuds des grappes Nancy et Rennes sont identiques. De plus, chaque commutateur de ces deux grappes comporte des options spécifiques d'optimisation pour la gestion des communications. Dans les expériences suivantes, nous comparerons les pénalités entre les grappes Lille et Sophia et entre les grappes Nancy et Rennes.

Résultats : En comparant les grappes de Lille et Sophia, figure 6.2, nous remarquons une nette différence entre les pénalités. Le commutateur de la grappe de Sophia ralentit de manière significative les communications concurrentes par rapport au commutateur de Lille. En détaillant cette analyse, nous remarquons que la hiérarchie des pénalités est analogue pour les deux grappes. En effet, pour chaque expérience, les communications les plus retardées sont identiques dans les deux grappes. En outre, lorsque deux communications d'une expérience subissent une même pénalité dans une grappe cela ce vérifie également dans l'autre grappe.

Ces phénomènes se retrouvent aussi dans la comparaison entre les grappes de Rennes et Nancy, figure 6.3. Cependant, la différence entre les valeurs des pénalités entre ces deux grappes est plus faible. Le commutateur de la grappe de Nancy provoque un faible surcoût par rapport à celui de Rennes.

Finalement, en comparant toutes les grappes, les trois grappes avec un commutateur performant proposent des pénalités plus faibles. En revanche, la hiérarchie des pénalités entre elles est préservée quelque soit la grappe. Ainsi, le commutateur est un composant influençant les valeurs des pénalités mais non leur hiérarchie.

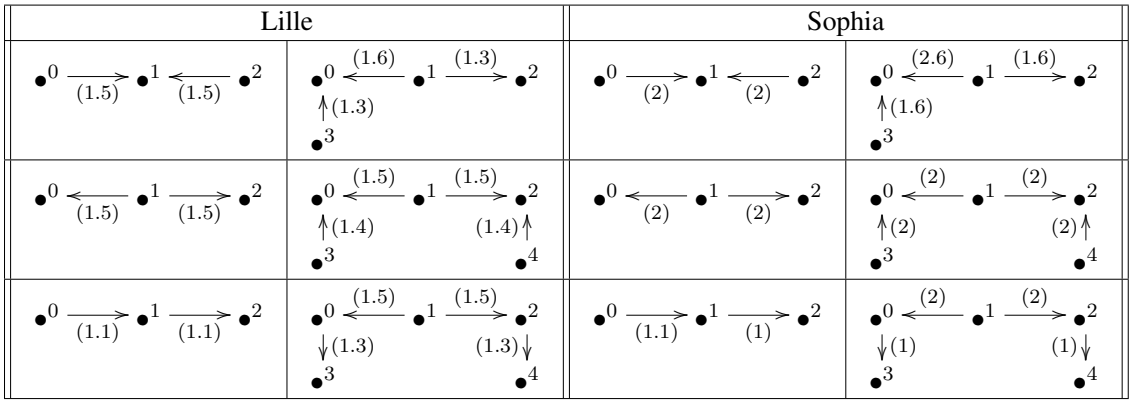


FIG. 6.2 – Comparaison de conflits complexes, grappes Lille et Sophia, *Gigabit Ethernet*.

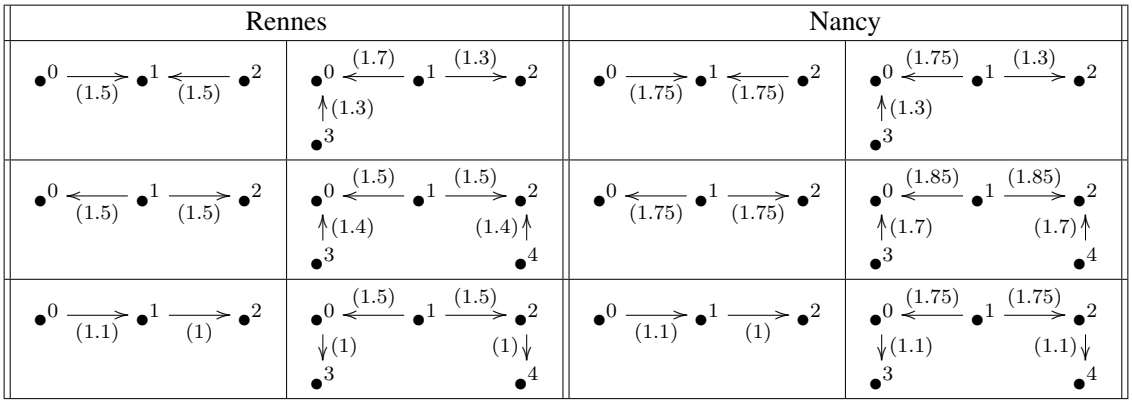


FIG. 6.3 – Comparaison de conflits complexes, grappes Rennes et Nancy, *Gigabit Ethernet*.

6.2.3 Utilisation de deux cartes réseaux

Nous avons vu précédemment lors des expériences traitant de l'impact du bus *PCI* que l'ajout d'une carte réseau génère un surcoût au niveau système et donc des performances. Cependant, dans les expériences que nous proposons dans cette sous-section, nous utiliserons le réseau *Quadrics* et non plus le réseau *Gigabit Ethernet*. En effet, la société *Quadrics* donne la possibilité d'ajouter des cartes réseaux (ou rail) par nœud. Elle développe donc des pilotes optimisés dans la gestion de plusieurs cartes. Néanmoins, l'utilisation de deux cartes réseaux peut-être réalisée de différentes manières suivant par exemple la taille des messages. Dans la nouvelle version de son pilote, *Quadrics* répartit de façon équitable un message de grande taille entre les cartes. De la même manière, une rafale de petits messages sera partagée équitablement sur les deux cartes.

Objectif : L'objectif des expériences est donc d'évaluer le gain obtenu sur les pénalités en ajoutant une seconde carte *Quadrics*. Dans une première série d'expériences, nous reprenons les conflits de base. Dans une seconde série, nous multiplierons par deux le nombre de communications de ces mêmes conflits.

Contexte expérimental : Il s'agit du même contexte expérimental que les expériences introductives sur *Quadrics*, mais en utilisant deux cartes. Remarquons, qu'il est important dans un conflit E/S ou E/E de déterminer si les deux cartes sont bien utilisées et donc s'il y a une coordination entre les nœuds. L'étude du conflit S/S semble évidente pour déterminer la répartition sur les pénalités.

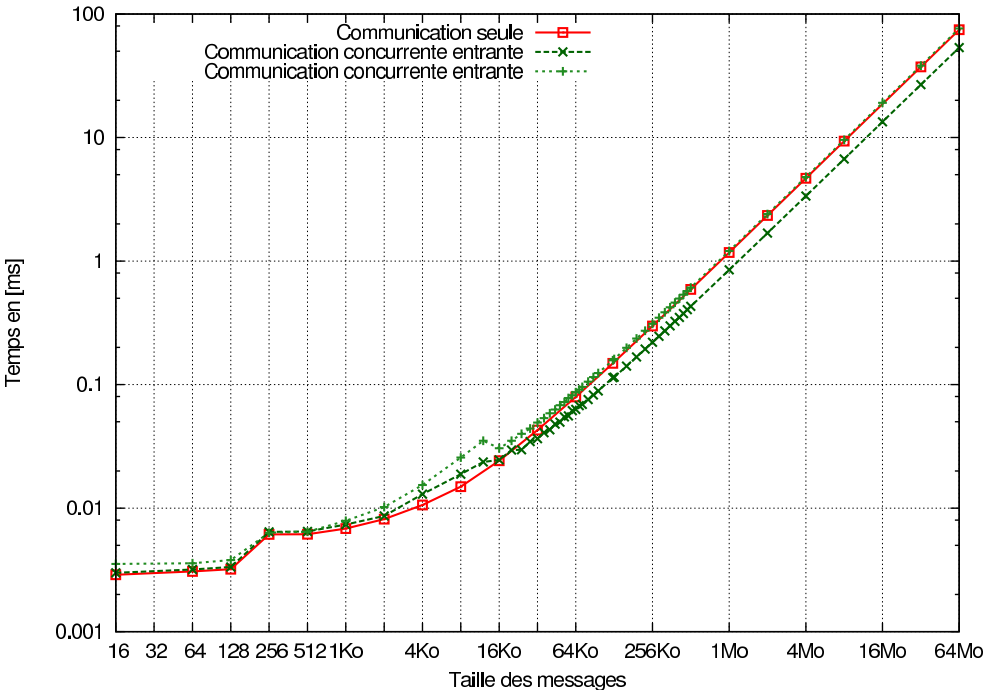


FIG. 6.4 – Conflit E/S sur 2 rails, *Quadrics*.

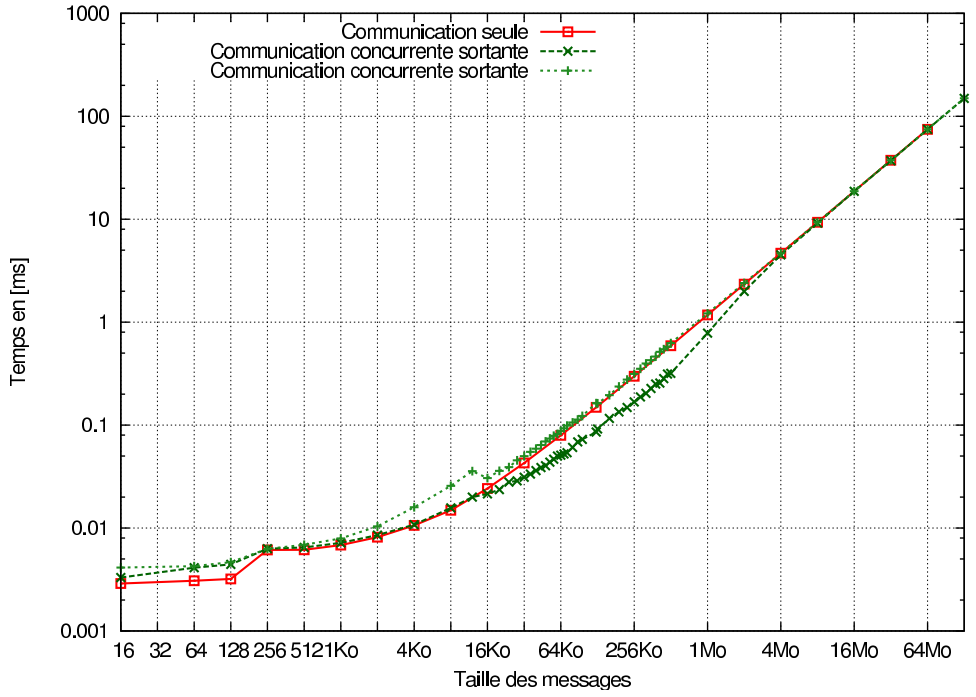


FIG. 6.5 – Conflit S/S sur 2 rails, *Quadrics*.

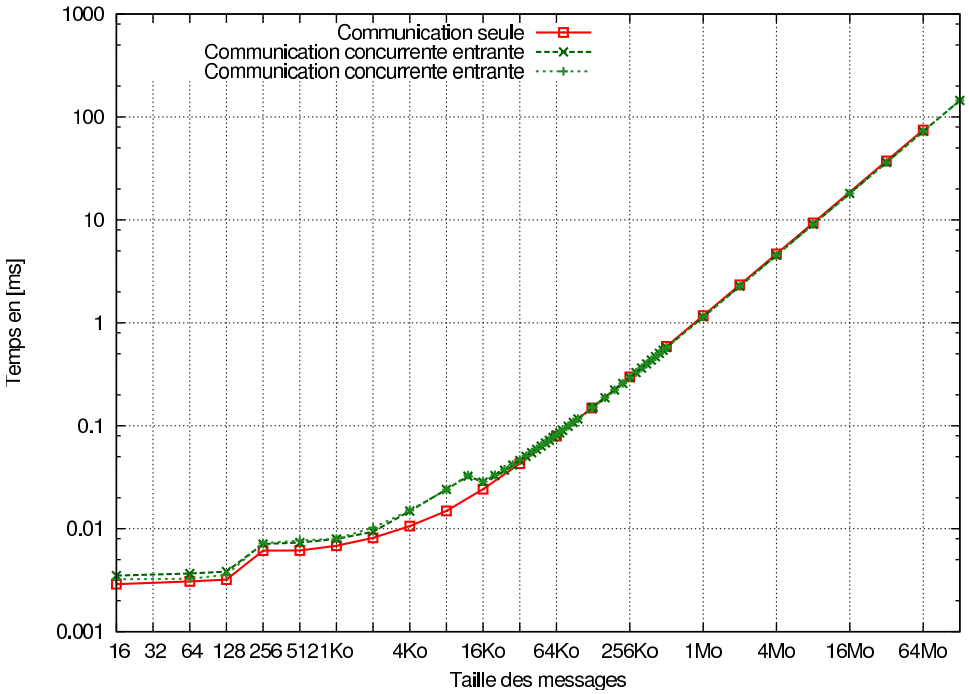


FIG. 6.6 – Conflit E/E sur 2 rails, *Quadrics*.

Résultats : Chacun des trois conflits montre que les temps des communications en concurrence sont proches, voire égaux à ceux d'une communication sans concurrence. En explorant de manière plus approfondie les résultats du conflit E/S, nous observons une amélioration des temps pour l'une des deux communications. Cela nous conduit à penser que le pilote de *Quadrics* gère parfaitement les deux rails. Néanmoins, pour chaque conflit, un coût est visible pour des messages entre 4 et 16 Ko. En effet, le découpage des messages entre les rails ne s'effectue qu'à partir de 16 Ko. La réduction de cette taille de 16 Ko à 4 Ko pourrait améliorer les performances obtenues.

Si nous augmentons le conflit S/S en faisant intervenir deux groupes de 4 communications à la place des deux communications, nous obtenons une division des pénalités par deux. Ainsi, ce conflit génère des pénalités valant 4 par communication.

Finalement, *Quadrics* propose une gestion efficace des deux rails qui divise, pour des conflits simples, par deux les pénalités.

6.2.4 Communication intra-nœud et communication inter-nœud

Une question qui peut-être posée dans des nœuds multiprocesseurs est l'influence entre communications intra-nœuds et communications inter-nœuds. Nous avons vu précédemment dans le cas de *Quadrics* qu'il n'y avait quasiment aucune influence entre ces communications. Toutefois, dans le cas de *Gigabit Ethernet* et *Myrinet*, cette propriété reste-t-elle vraie ?

Objectif : Nous allons donc présenter quelques expériences montrant l'existence ou non d'une influence entre communication intra-nœud et inter-nœud. Pour se faire, nous reprendrons les ex-

périences introductives traitant les conflits I/S et I/E.

Contexte expérimental : Le contexte expérimental reprend les mêmes caractéristiques que celui présenté dans le premier chapitre de la partie II. Cependant, pour organiser sur des machines biprocesseurs un conflit I/S ou I/E, et comme dans le chapitre précédent, nous affectons plusieurs tâches à un processeur.

Résultats : Dans le cas de *Myrinet*, figure 6.7, les temps de communication intra-nœud et inter-nœud semblent être indépendants. Toutefois, la concurrence entre ces deux communications produit un ralentissement pour des messages de petite taille. Nous ne présentons pas le conflit I/E car les courbes sont très similaires.

Le cas du réseau *Gigabit Ethernet* est très proche de *Myrinet* dans le comportement des coûts de communication, figure 6.8. Néanmoins, le ralentissement des communications intra-nœud par les communications inter-nœud pour des messages de petite taille ne se reproduit plus. Les latences des communications inter-nœud sont largement plus élevées que celles du réseau *Myrinet*, ce qui en gomme probablement l'effet.

En conclusion, les communications intra-nœuds et inter-nœuds n'ont que de très faibles interactions entre elles. De plus, si on augmente le nombre de communications des conflits I/E ou I/S, il est fort probable que les comportements spécifiques à ces conflits seront difficiles à observer. Effectivement, si nous ajoutons des communications inter-nœuds, des conflits plus pénalisant tels que des conflits E/E ou S/S empêcheront l'observation des comportements I/E ou I/S. Cependant, dans le cas où nous ajoutons des communications intra-nœuds, il reste à déterminer le comportement des conflits générés. L'étude de ces conflits est traitée dans la sous-section suivante.

6.2.5 Conflits intra-nœuds

La hiérarchie mémoire des nœuds *NUMA* se caractérise par plusieurs chemins physiques menant aux blocs mémoires. Cette propriété est d'autant plus importante lorsque nous traitons uniquement des communications intra-nœuds entre blocs. De ce fait, existe-t-il des effets similaires pour des conflits E/E, S/S ou E/S entre les différents blocs mémoire du nœud ?

Objectif : L'objectif de cette étude est d'identifier les pénalités de conflit uniquement à l'intérieur du nœud. Une machine *NUMA* est considérée comme un ensemble de blocs associant un nombre de processeur et une mémoire. De même, une communication n'est plus entrante ou sortante mais en lecture ou écriture sur la mémoire d'un bloc. Ainsi, nous considérons les mêmes expériences que précédemment mais en modifiant les correspondances entre termes (un nœud devient un bloc, une communication entrante est dite en écriture et une communication sortante en lecture).

Contexte expérimental : La plate-forme utilisée est une machine *Bull Novascale* (nœud de la grappe Teratec) comportant quatre blocs de quatre processeurs. Cela étant, un bloc permet d'exécuter quatre communications simultanément. L'interface *MPI* utilisée est *MPI Bull* qui, dans le cas

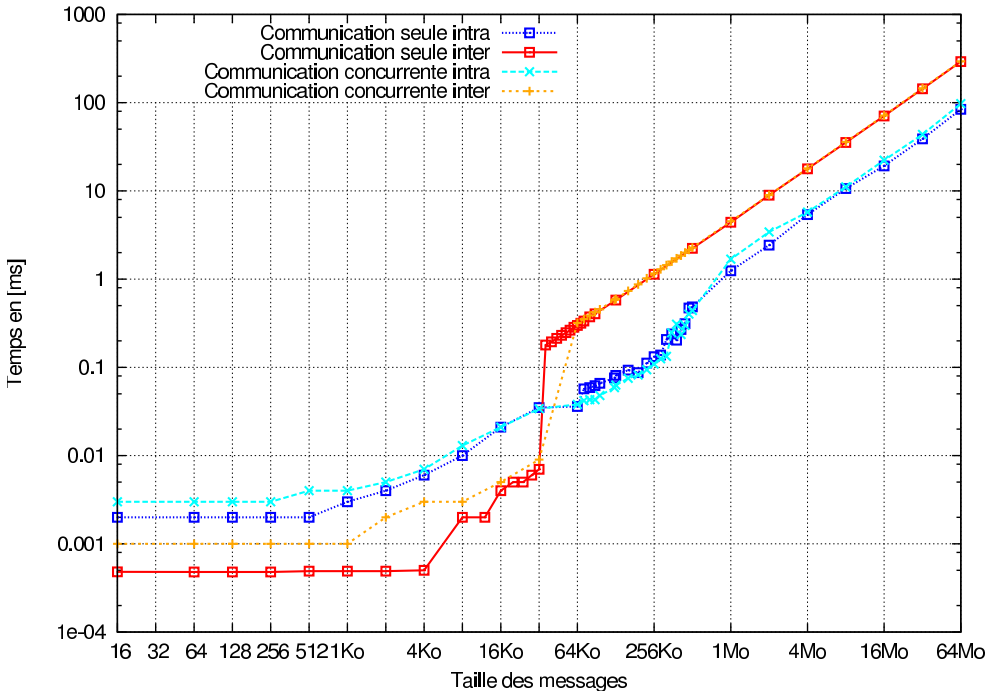


FIG. 6.7 – Conflit I/S, *Myrinet*.

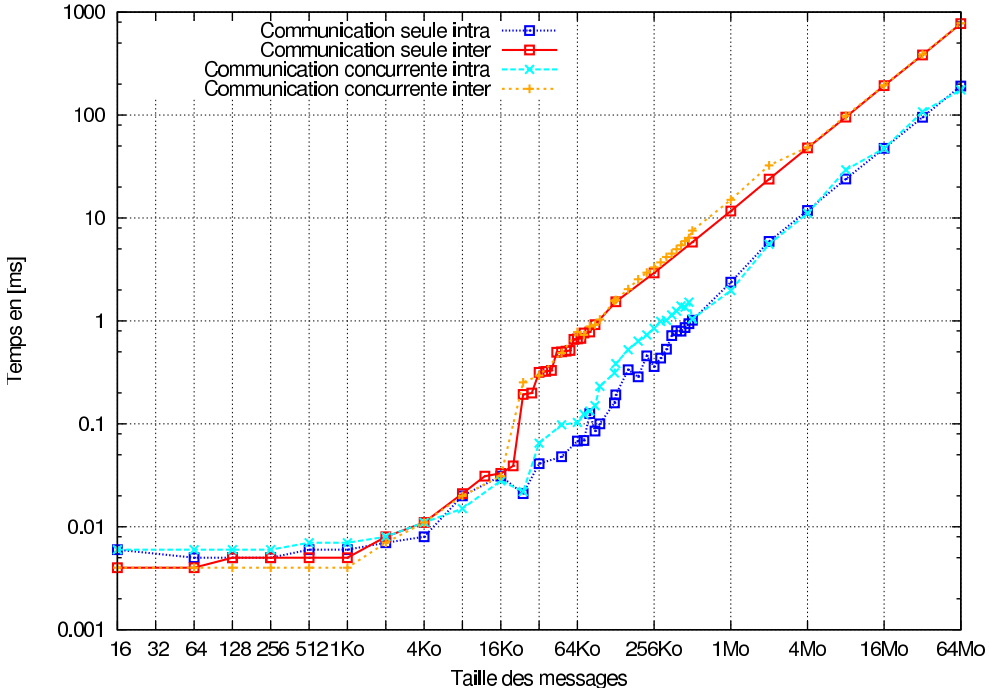


FIG. 6.8 – Conflit I/S, *Gigabit Ethernet*.

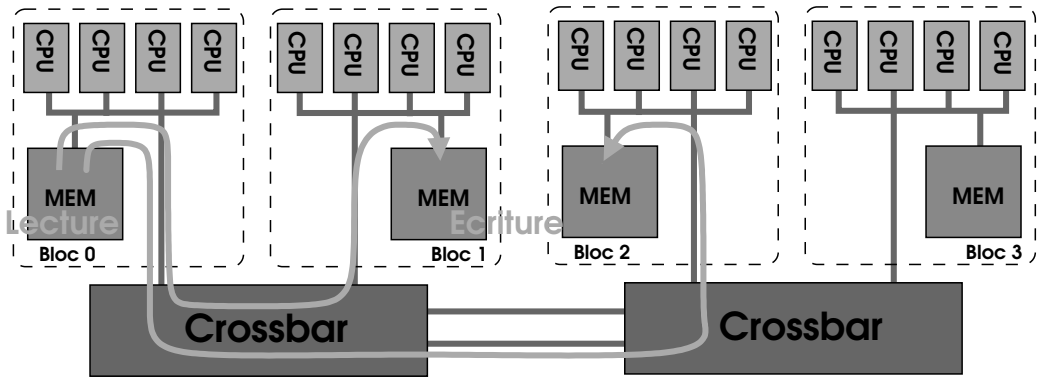


FIG. 6.9 – Exemple d'expérience entre deux communications intra-nœud.

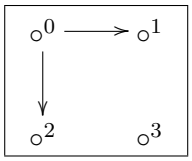


FIG. 6.10 – Exemple de schématisation des expériences intra-nœud.

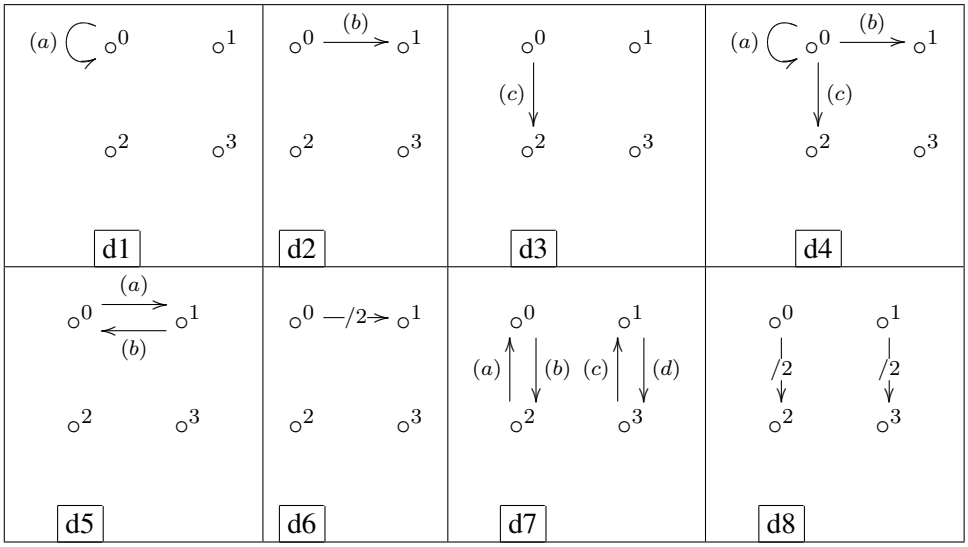


FIG. 6.11 – Schémas des expériences entre blocs.

de communication intra-nœud, utilise un chemin le plus direct pour copier la zone mémoire. La figure 6.9 explicite une expérience entre communication intra-nœud. Dans cette expérience deux communications lisent des données depuis la mémoire du bloc 0, une les écrit dans la mémoire du bloc 1 et l'autre dans celle du bloc 2. La figure 6.10 schématise cette expérience.

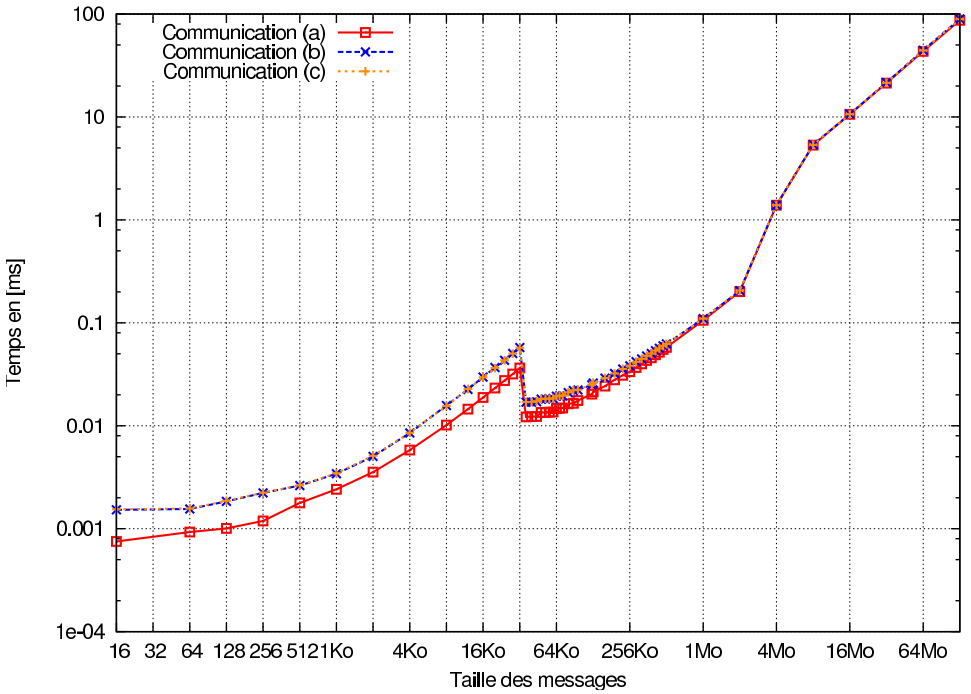


FIG. 6.12 – Communications intra-nœuds des schémas d1,d2 et d3.

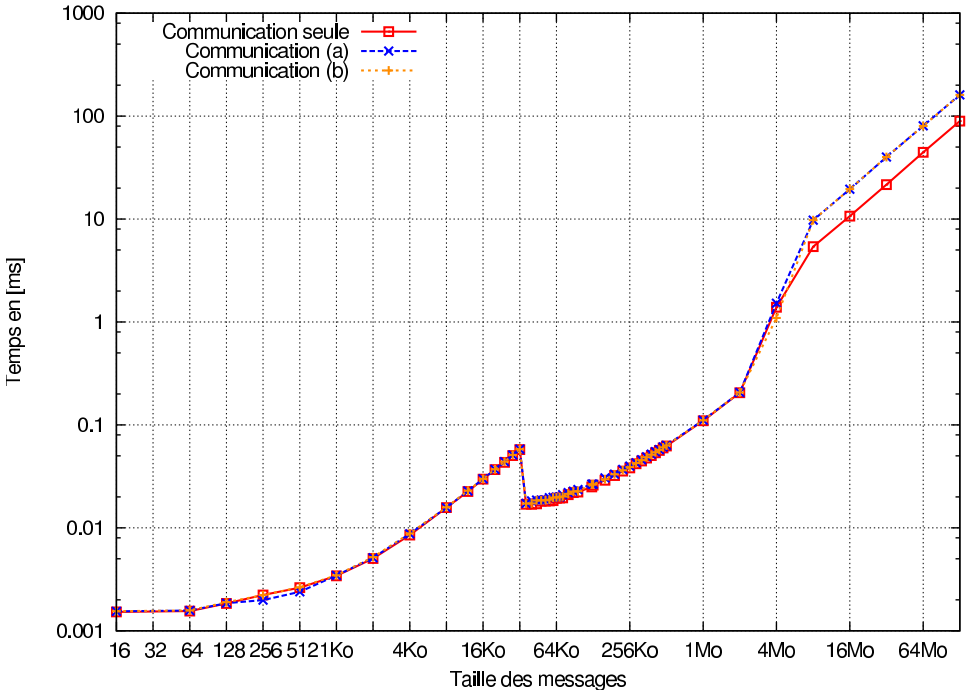


FIG. 6.13 – Communications intra-nœuds du schéma d6.

Résultats : Les premières expériences déterminent le comportement des communications seules transférant leurs données vers les autres blocs. La schématisation, figure 6.11, décrit ces différentes expériences par les schémas d1, d2 et d3. Les résultats obtenus, dans la figure 6.12, montrent pour des messages de petite taille (< 32Ko) une latence supplémentaire pour les communications (b) et (c). Cette latence correspond au déplacement des données à travers les commutateurs internes. Cependant, les valeurs pour les communications (b) et (c) sont identiques. Nous en déduisons que le coût de traversée d'un ou deux commutateurs est similaire. A 32Ko, les communications sont plus rapides au moment de l'utilisation du protocole par rendez-vous. Cela peut s'expliquer par le fait que ce protocole évite une copie entre un tampon mémoire temporaire et le tampon destinataire de la communication. Finalement, au-delà de cette taille, la bande passante a un impact supérieur à la latence et les trois communications se comportent de manière similaire.

Le schéma d4 regroupe les trois précédents schémas. Nous ne présentons pas les résultats obtenus pour ce schéma car ils sont similaires aux résultats de la figure 6.12. Partant de cette constatation, nous déduisons que les trois communications de ce schéma créent une perturbation très faible entre elles.

Il en est de même pour le schéma d5 qui ne montre aucune pénalité sur les communications (a) et (b) par rapport à une communication seule (schéma d2). Cependant, dans la figure 6.13, le schéma d6 montre un comportement que nous avions déjà rencontré pour des communications inter-nœud. En effet, les deux communications concurrentes ne subissent aucune pénalité jusqu'à un seuil (8 Mo) à partir duquel leurs temps doublent. En examinant ce phénomène avec un nombre de communications plus élevé, nous concluons que la pénalité associée aux communications est égale au nombre de communications. De plus, la comparaison de ce schéma, avec un schéma où deux communications sortantes d'un même nœud (schéma d4 sans la communication (a)) révèle une différence entre la lecture et l'écriture des données. En effet, un schéma impliquant uniquement des lectures depuis une même mémoire ne provoque pas de ralentissement. A l'inverse, l'écriture simultanée d'une grande quantité de données provoque un fort ralentissement.

De manière similaire aux schémas d4 et d5, le schéma d7 ne montre qu'un surcoût très faible pour chaque communication. Enfin, le schéma d8 est identique au schéma d6, c'est-à-dire que chaque communication de chaque groupe de deux communications prend une pénalité de deux.

Au vu de ces expériences nous remarquons une grande similitude avec les expériences inter-nœuds. Il ressort de ces résultats que peut être soulevée la question de savoir si la définition d'un modèle prédictif est concevable ? Nous pensons qu'effectivement un modèle est réalisable mais relatif à la structure interne de la machine. Pour explorer plus avant la faisabilité d'un tel modèle, il faudrait mener des expériences complexes sur des machines *FAME* composées de 8 blocs.

Finalement, les communications intra-nœuds ont une faible influence sur des communications inter-nœud, mais révèlent une hiérarchie lors de conflits internes à la machine.

6.3 Influence logicielle

La section précédente avait pour objectif de déterminer l'influence matérielle de certains composants du chemin réseau. Cette section, en revanche, s'attèle à déterminer des paramètres logiciels modifiant significativement les comportements concurrents. Nous traitons en particulier certains aspects logiciels relatifs aux choix des implantations ou du standard *MPI*. L'ensemble de ces ex-

périences est limité au réseau *Gigabit Ethernet*.

Dans la première série d'expériences, nous reviendrons sur l'impact que provoque un changement d'implantation *MPI*. Nous traiterons le cas du passage de *Mpich* vers *Lam/MPI*. Dans une deuxième partie, nous utiliserons la librairie *Socket* pour tester certains choix du standard *MPI*. En particulier, le protocole de rendez-vous est-il source de modification des pénalités ?

6.3.1 Implantations du standard *MPI*

Il existe plusieurs implantations *MPI* libres telles que *Mpich* et *Lam/MPI*. Chacune de ces implantations propose une conception propre avec un ensemble de choix de programmation qui peut avoir un impact sur les comportements concurrents. Il convient d'en déterminer leurs effets.

Objectif : Pour ce faire, nous commenterons les différences des pénalités pour les implantations *Mpich* et *Lam/MPI*.

Contexte expérimental : Les résultats commentés sont disponibles pour *Mpich* dans le chapitre 4, sous-section 4.3.2 et pour *Lam/MPI* dans l'annexe B. Chacune de ces expériences est identique en tout point au prorata du choix de l'implantation utilisée.

Résultats : En utilisant *Lam/MPI*, nous remarquons un comportement similaire pour chaque conflit. L'effet de saut, ainsi que ses conséquences, se répète pour chaque conflit. Cette valeur de saut correspond pour les deux implantations au changement de protocole par *MPI* pour un protocole avec rendez-vous.

Cependant, après cette valeur de saut, les pénalités diffèrent. Par exemple, pour les conflits E/E ou S/S, les pénalités pour *Lam/MPI* valent 2 alors que celles de *Mpich* valent 1,5. Cela nous conduit à proposer pour *Lam/MPI* des expériences plus élaborées afin de déterminer si la hiérarchie des pénalités est fonction de l'implantation *MPI*. Ces expériences élaborées montrent que *Lam/MPI* propose la même hiérarchie des pénalités que *Mpich*. En effet, comme lors des expériences sur les commutateurs, il est aussi possible que cette différence n'avait pas d'impact significatif.

En comparant les pénalités des expériences élaborées, nous remarquons que *Lam/MPI* est moins performant que *Mpich*. En effet, les pénalisations induites par la concurrence sont beaucoup plus fortes pour *Lam/MPI*. Cependant, leur hiérarchie reste préservée.

6.3.2 Caractéristiques de *MPI*

Cette section étudie l'impact de certains choix du standard *MPI* sur les pénalités des communications concurrentes.

Objectif : Cette étude concerne l'utilisation du protocole de rendez-vous. Le protocole de rendez-vous permet d'anticiper l'arrivée d'une communication de grande taille. Ainsi, le nœud destinataire peut allouer les tampons mémoires nécessaires et confirmer sa disponibilité pour le début de l'émission. Nous avons remarqué précédemment que les pénalités apparaissent à partir du moment où le protocole de rendez-vous est utilisé. Quel comportement aura la concurrence si ce protocole n'est pas utilisé ?

Contexte expérimental : Pour réaliser des expériences sans ce protocole, nous avons choisi de développer les méthodes *MPI_Send* et *MPI_Recv* avec la librairie *Socket*. En effet, dans nos expériences les données envoyées ne sont pas utilisées par le programme test après leurs réceptions. De ce fait, ces données peuvent être écrasées. Cela nous conduit à utiliser une implantation du *MPI_Recv* avec un tampon mémoire fixe, qui sera écrit de façon cyclique par les données reçues. Ainsi, le protocole de rendez-vous est inutile car il sert essentiellement à préciser la taille à allouer pour le tampon de réception. De même, le temps nécessaire à cette allocation ou l'espace mémoire utilisé sont fortement réduits. Nous justifions l'utilisation de la librairie *Socket* par le fait qu'une telle technique ne serait pas envisageable sans une modification lourde d'une implantation *MPI* telle que *Mpich*. La librairie *Socket* est configurée avec les mêmes options que celles utilisées par *Mpich* et de manière bloquante.

Résultats : Dans la première expérience, nous comparons les temps de communications entre *Mpich* et l'implantation *Socket* sans rendez-vous. Les courbes des temps sont présentées dans la figure 6.14. Nous notons deux particularités :

1/ l'implantation *Socket* donne des latences pour de petits messages très nettement supérieures à *Mpich*. En effet, notre implantation ne propose aucune optimisation des latences à l'inverse de *Mpich*.

2/ nous remarquons aucun effet de saut ni une bande passante plus efficace pour la librairie *Socket*.

Il est évident que sans protocole de rendez-vous il ne se produit pas de saut. De plus, l'utilisation d'un tampon fixe est responsable du gain en bande passante.

Suivant ce contexte, nous présentons les résultats des expériences élaborées mettant en jeu de grande taille de message (> 10Mo). Ces expériences, figure 6.15, montrent un comportement et des pénalités très similaires à celles de *Mpich*. La comparaison des valeurs des pénalités est peu significative car les bandes passantes sont différentes. Cependant, ce comportement similaire indique que le protocole de rendez-vous n'a pas d'influence directe sur la hiérarchie des pénalités.

6.4 Conclusion

Ce chapitre a pour objectif de montrer l'impact de certains composants matériels et logiciels sur les pénalités observées lors de communications concurrentes. Les expériences ont principalement été menées sur le réseau *Gigabit Ethernet* qui propose une plus grande hétérogénéité matérielle.

Au niveau des composants matériels, l'isolation du bus *PCI* a révélé une faible influence sur les pénalités. Lors des expérimentations, il semble fort probable que cette influence soit due au système. En revanche, les commutateurs ont un impact fort. Les valeurs des pénalités varient fortement avec différents commutateurs. Cependant, leurs variations sont identiques. Ainsi, les commutateurs n'influent pas sur la hiérarchie des pénalités.

En se consacrant aux ressources internes du nœud, dont la mémoire, les expériences ont montré des caractéristiques intéressantes. En particulier, des communications intra-nœuds et inter-nœuds ne s'influencent pas mutuellement. En revanche, les conflits mettant en jeu uniquement des communications intra-nœuds dévoilent des comportements particuliers. En effet, il est envisageable

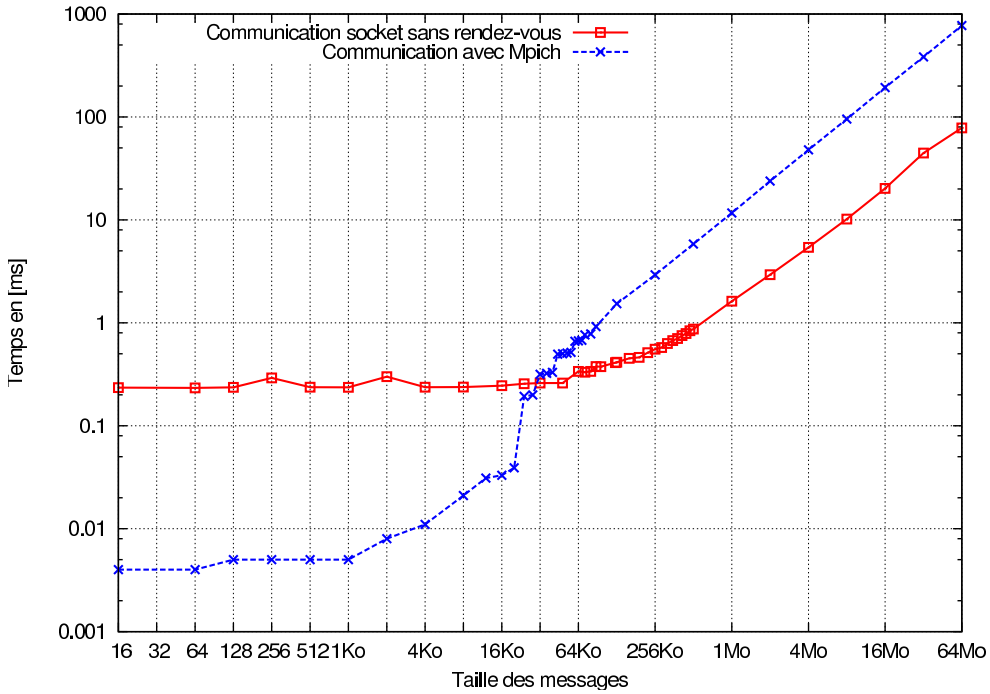


FIG. 6.14 – Comparaison des temps de communications avec et sans protocole de rendez-vous, *Gigabit Ethernet*.

de déterminer des modèles prévoyant les coûts associés à des conflits mémoire. Ces expériences révèlent aussi que les écritures multiples sont plus pénalisantes que des lectures.

Au niveau des composants logiciel, le changement d'implantation *MPI* caractérise les valeurs des pénalités. Cependant, comme pour les commutateurs, la hiérarchie des pénalités n'est pas affectée. De même, le protocole de rendez-vous n'exerce pas de modification sur cette hiérarchie.

En conclusion, l'influence de certains composants est notable sur les valeurs des pénalités. Toutefois, aucun composant testé ne modifie la hiérarchie des pénalités. Pour le réseau *Gigabit Ethernet*, ces deux propriétés confirment la faisabilité d'un modèle prédictif quel que soit la combinaison des composants modélisés. Une approche quantitative basée sur un ensemble de mesures semble être plus appropriée pour construire un tel modèle. En effet, même si les expériences montrent une hiérarchie entre pénalités qui pourrait être modélisée de façon descriptive, ce modèle nécessiterait, de toutes manières, un ensemble de mesures pour évaluer les pénalités.

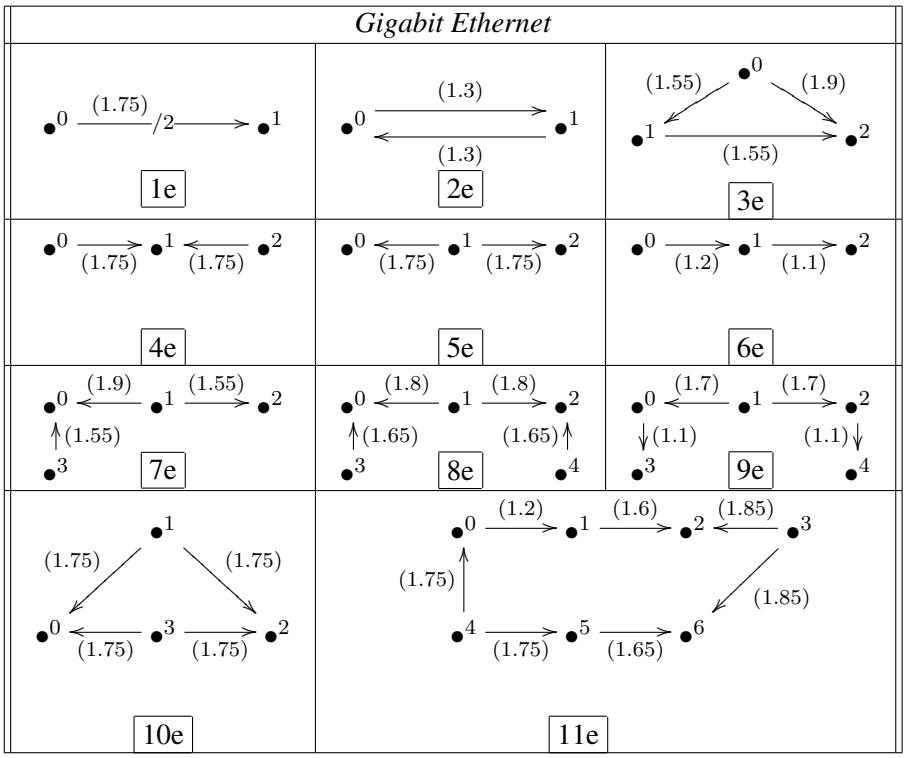


FIG. 6.15 – Conflits complexes sans rendez-vous, *Gigabit Ethernet* et *Socket*.

Evaluation des modèles pour la prédiction 7

7.1 Introduction

Ce chapitre traite de l'évaluation des modèles de communications définis dans le chapitre 5. Pour réaliser cette évaluation, nous comparons les valeurs obtenues par une prédiction basée sur les modèles avec les valeurs obtenues par expériences. Ceci nous permet d'évaluer concrètement la qualité de nos modèles avec la réalité, mais aussi leur intégrabilité dans un processus de simulation et de prédiction.

Pour évaluer, ces modèles nous avons utilisé deux types de graphes :

1/ Des graphes synthétiques, comme des arbres ou graphes complets qui ont des propriétés (degré, diamètre) assez différentes. Nous étudions le comportement des modèles face à ces propriétés.

2/ Des graphes d'applications issus d'applications réelles. Nous montrons comment nos modèles prédictifs se comportent face à ces graphes. Nous étudions le *benchmark HPL* qui est largement utilisé pour établir le classement du TOP500 des plates-formes les plus puissantes. Ces graphes d'applications sont extraits à partir de l'exécution de ce *benchmark*. Un traçage des événements de l'exécution est réalisé et sera plus amplement commenté dans la section 7.5.

Nous présentons, dans la section 7.2, les différentes métriques pour estimer l'erreur entre les valeurs mesurées et les valeurs prédites. Le calcul des valeurs prédites est réalisé grâce à un simulateur décrit dans la section 7.3. Ce simulateur permet de calculer les prédictions suivant différents schémas de communications ou graphes de communications. Pour évaluer les modèles, nous utilisons, dans la section 7.4, des graphes synthétiques puis, dans la section 7.5, des graphes extraits d'applications.

L'analyse des comportements des réseaux montre certains phénomènes inattendus. Les utilisateurs devraient en tenir compte notamment pour obtenir des gains de performances significatifs. Nous montrons, dans la section 7.6, quelques exemples pour appréhender ces phénomènes inattendus.

7.2 Méthode d'évaluation

L'évaluation des modèles se base sur la comparaison entre les temps prédits T_p et les temps mesurés T_m pour différents graphes.

Dans le cas des graphes synthétiques, cette comparaison est effectuée en calculant l'erreur relative pour chaque communication. De plus, pour mesurer l'erreur globale du graphe, nous calculons la moyenne des erreurs absolues des communications du graphes. Ces deux erreurs sont mesurées en pourcentage. Pour un graphe constitué de N communications et une communication notée c_k, les formules calculant ces erreurs sont présentées ci-dessous :

$$E_{rel}(c_k) = \frac{T_p - T_m}{T_m} \times 100$$

$$E_{abs}(G) = \frac{1}{N} \sum_{k=1}^{N} |E_{rel}(c_k)|$$

L'erreur relative permet d'avoir une vision fine de la précision des modèles. Elle permet entre autre de montrer si le modèle a un comportement optimiste (erreur négative) ou pessimiste (erreur positive). A l'inverse, la moyenne des erreurs absolues donne une vision globale des précisions sur le graphe donné. L'utilisation de l'erreur absolue évite des comportements de compensation entre erreurs relatives, et donc une description plus exacte des prédictions. Les mesures sont réalisées par le programme présenté dans le chapitre 4.

Dans le cas de graphe provenant d'applications, nous calculons pour chaque tâche la somme des temps des communications prédites et mesurées. Pour une tâche t_i, ces sommes sont notées $S_m = \sum_{c_k \in t_i} T_m$ pour les communications mesurées et $S_p = \sum_{c_k \in t_i} T_p$ pour les communications prédites. En se basant sur ces valeurs, nous calculons l'erreur absolue pour une tâche t_i :

$$E_{abs}(t_i) = |\frac{S_p - S_m}{S_m} \times 100|$$

Pour obtenir les valeurs prédites, nous utilisons un simulateur qui a été décrit dans le chapitre 5, section 5.4. Dans la section suivante, nous détaillons certains aspects de son implantation et de son utilisation.

7.3 Simulation

Afin de calculer les prédictions des différents modèles, nous avons implanté un simulateur. Ce simulateur prend en entrée différents paramètres :

- une ou plusieurs applications représentées par une séquence d'événements. Les événements sont de deux types : événement de calcul et événement de communication. Un événement de calcul comporte uniquement le temps de calcul alors qu'un événement de communication est défini par le numéro des tâches source et destination et par la quantité de données à transmettre.
- la définition de l'architecture comportant chaque nœud de la grappe en indiquant leur nombre de processeurs. Les machines sont numérotées de manière itérative, en commençant par zéro.

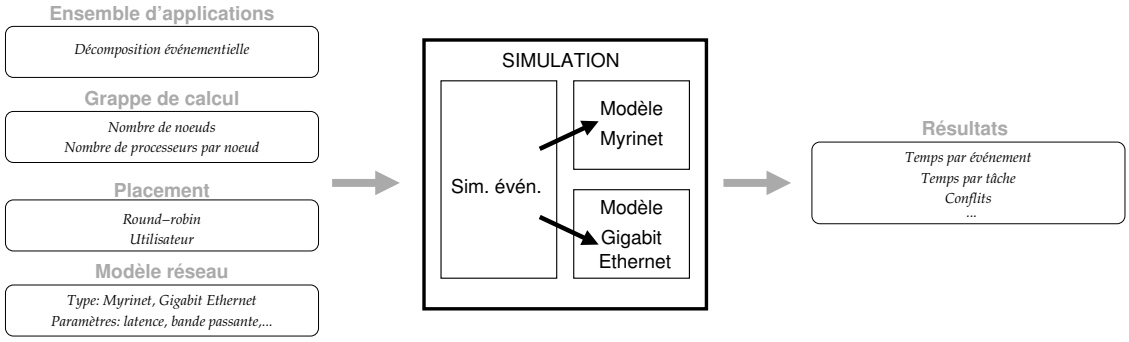


FIG. 7.1 – Architecture du simulateur

– le placement des tâches sur les nœuds. Il peut-être explicite ou pré-défini, comme étant par exemple un placement du type *Round-Robin*. Dans le cas où le placement est décrit par l'utilisateur, il est représenté par une suite de chiffres qui correspondent aux numéros des machines de la grappe. Le premier chiffre correspond au placement de la première tâche, le second chiffre au placement de la seconde tâche, etc.
– la définition du type du modèle et de ses paramètres : la latence, la bande passante, la taille du tampon délimitant les communications concurrentes des communications non concurrentes, etc.

La figure 7.1 schématise les paramètres d'entrée nécessaires, la structure du simulateur et les résultats obtenus.

Pour réaliser une simulation, le choix d'une sémantique de communication et de son implantation sont importants. Cette sémantique décrit le comportement des tâches lorsqu'elles initient une émission ou une réception. La sémantique choisie des communications est une sémantique synchrone. La durée d'une communication comprend le temps de synchronisation entre l'émetteur et le récepteur et le temps de transfert des données. Cette implantation est décrite dans la figure 7.2. Dans le cas de communications intra-nœud sur le réseau *Gigabit Ethernet* associées à *Mpich*, nous proposons une implantation légèrement différente. Cette nouvelle implantation est décrite dans la figure 7.3. *Mpich* utilise la librairie *Socket* pour effectuer l'envoi des données, également dans le cas de communications intra-nœud. L'utilisation de cette librairie dans ce cas implique deux phénomènes :

– une bande passante beaucoup plus faible (environ trois fois moins importante que celle mesurée par l'utilisation d'une fonction *memcpy*) ;
– une modification de la sémantique. Le tampon mémoire d'envoi utilisé dans la fonction *MPI_Send* peut-être réutilisé plus rapidement. Effectivement, si un envoi est initié avant une réception, les données sont copiées dans le tampon de la *Socket* durant la phase d'attente. A partir du moment où la réception est prête, le *MPI_Send* bloquant peut-être relâché.

Le simulateur fournit en sortie différentes mesures telles que la durée de chaque tâche, la durée de chaque événement, le nombre de communications en conflit, le nombre de conflits, la pénalité moyenne entre l'ensemble des conflits ou encore les volumes de communications.

En résumé, la simulation s'articule autour de trois points :

• La trace d'une application décomposée en événements, dont les rangs logiques des commu-

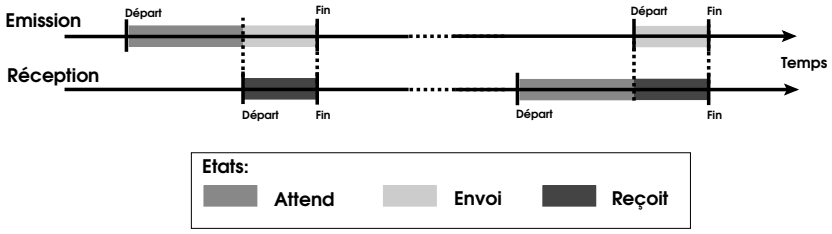


FIG. 7.2 – Implantation de la sémantique synchrone des communications

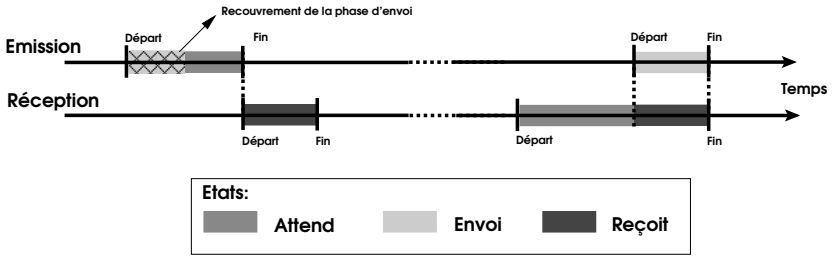


FIG. 7.3 – Implantation de la sémantique synchrone des communications intra-nœud pour *Gigabit Ethernet* et *Mpich*

nications sont transformés en numéros de machines pour un placement donné ;
- Des modèles de répartition de bande passante et d'interactions des communications ;
- le choix d'une sémantique des communications et son implantation.

Le simulateur a été réalisé en *C++* et comporte environ 4200 lignes de code. Sa conception permet d'incorporer facilement de nouveaux modèles grâce à l'utilisation de classes abstraites. Il peut aussi intégrer des modèles de la littérature présentés dans le chapitre 3.

7.4 Evaluation sur des graphes synthétiques orientés

Les graphes synthétiques permettent d'explorer la pertinence des modèles suivant une certaine corrélation entre les conflits générés. Il est intéressant de déterminer l'évolution des erreurs afin d'exhiber pour chaque conflit les faiblesses et les points forts des modèles.

Nous choisissons de focaliser notre étude sur deux familles de graphes : les arbres et les graphes complets. Les graphes complets ont des conflits fortement reliés où chaque communication sortante est en conflit sur le nœud qu'elle atteint. A l'inverse, les arbres proposent des conflits impliquant moins de communications et des communications moins concentrées. Pour obtenir ces graphes, nous construisons des séquences particulières d'événements qui représentent les tâches d'une application. Chacune de ces séquences est composée d'un seul événement de communication. Ainsi, deux tâches, une composée d'un événement en émission et l'autre de l'événement en réception correspondant, forment une communication. En plaçant de manière judicieuses ces tâches sur les machines de l'architecture simulée, nous pouvons créer les familles de graphes utilisées.

Chaque communication de ces graphes débute simultanément et transmet 8Mo de données. A

titre indicatif, les temps d'une communication de 8Mo sont présentés pour chaque réseau dans le tableau 7.1.

Les grappes utilisées pour évaluer les graphes synthétiques sont pour *Myrinet* : la grappe de Sophia et pour *Gigabit Ethernet* : la grappe de Rennes. Les paramètres nécessaires au modèle *Gigabit Ethernet* sont identiques à ceux présentés dans le chapitre 5, sous-section 5.5.2.3. Ces paramètres valent : $\beta = 0,71$, $\gamma_s = 0,115$ et $\gamma_e = 0,036$. Lors de l'évaluation des graphes provenant d'applications, nous utiliserons la grappe Helios. Pour cette grappe, les valeurs du modèle *Gigabit Ethernet* sont : $\beta = 0,99$, $\gamma_s = 0,12$ et $\gamma_e = 0,11$. La différence entre les valeurs des paramètres reflète la différence entre les technologies des composants réseaux des grappes. Des valeurs plus faibles indiquent des composants plus performants.

Taille	*Myrinet*	*Gigabit Ethernet*
8 Mo	0.0355	0.0966

TAB. 7.1 – Temps des communications seules en seconde

7.4.1 Arbres

Les arbres ont la particularité de proposer des corrélations faibles entre conflits. Chaque conflit situé au niveau d'un nœud de l'arbre est relié uniquement aux conflits des nœuds fils et à celui du nœud père. La racine et les feuilles de l'arbre comportent un seul conflit.

Nous utilisons quatre arbres orientés. Les deux premiers arbres A1 et A2 sont opposés suivant l'orientation des communications. Ces arbres ont la particularité d'avoir un grand nombre de conflits E/S simples. De leur côté, les deux arbres A3 et A4 ont un grand nombre de conflits E/E et S/S.

***Myrinet* :**

De très bons résultats sont obtenus sur les arbres MA1 et MA2, figure 7.4. Sur l'exemple MA1 avec les communications (a), (b) et (c) ont constate que la bande passante disponible est équitablement partagée. Mais, cette bande passante disponible est inférieure à la bande passante physique, la différence provenant d'une perte due à la gestion des conflits. Cette propriété ressort clairement des résultats du simulateur. Il en est de même pour l'arbre MA2. Plus précisément, l'arbre MA1 obtient une erreur globale de 4% et l'arbre MA2 d'environ 7%. On notera que les mesures des conflits S/S sont plus stables que celles des conflits E/E. La variabilité des mesures des conflits E/E se retrouvent également dans les valeurs des erreurs de ces conflits.

L'arbre MA3 montre d'excellents résultats de 2.6%. Le modèle semble apte pour modéliser des arbres homogènes en conflits.

Finalement, le dernier arbre MA4 met plus en défaut le modèle avec une erreur de 20%. Cette erreur est principalement due aux communications (c) et (h) qui sont en conflit E/S sur le nœud 3. Il semble que le modèle prédise de façon trop optimiste ce conflit quand il est composé de deux communications. La modélisation d'un tel conflit n'induit pas de ralentissement sur la communication (h). Cette propriété correspond à l'utilisation de lien full-duplex. Toutefois, il semble qu'un ralentissement se produit, dont il est difficile d'isoler l'origine. Sans les fortes erreurs de ces deux communications, l'erreur globale redescendrait aux environs de 10%.

En conclusion, le modèle *Myrinet* s'adapte bien à des schémas de communications arborescents, même si un conflit du type E/S à deux communications est traité de manière trop optimiste.

***Gigabit Ethernet* :**
Les résultats du modèle *Gigabit Ethernet* sont présentés dans la figure 7.5. Sur l'arbre GA1, on observe d'excellents résultats avec un taux d'erreur quasiment nul. Ces résultats confortent la validité du modèle dans le cas d'arbres où les conflits S/S dominent. Lorsque les conflits E/E sont dominants, sur l'arbre GA2, les résultats restent satisfaisants. Cependant, nous noterons que l'erreur sur la communication (b) avoisine les 26%. Le modèle considère la communication (b) au même titre que les communications (a) et (c) alors que le nœud source de (b) ne subit aucun conflit. Il conviendrait certainement d'approfondir le comportement du modèle dans cette situation afin d'affiner les équations du calcul des pénalités.

Les temps des communications de l'arbre GA3 sont prédits avec une erreur globale de 8%. En approfondissant cette analyse, nous remarquons que la communication (e) est prédite de manière optimiste avec une erreur de -23,7%. Nous remarquons que le conflit du nœud 1 de l'arbre GA3 (nœud source de (e)) est identique aux conflits des nœuds 1 et 3 de l'arbre GA2. Dans ces deux conflits, les temps des communications sortantes (a) et (c) sont aussi sous-estimés. La modélisation proposée ne considère pas l'influence entre communication entrante et sortante. Une faible variation des temps se produit qu'il doit être possible d'optimiser en raffinant le modèle.

Finalement, le dernier arbre étudié, GA4, propose des performances globalement satisfaisantes.

De manière générale, le modèle est optimiste dans le cas des arbres. La plupart des erreurs relatives sont négatives. Certains arbres ont montré qu'il est possible d'approfondir la modélisation des conflits E/S. Cependant, l'influence de ces conflits est faible et mineure face à celle des autres types de conflits. Cela a pour effet de la rendre plus difficile à capturer dans un modèle prédictif.

7.4.2 Graphes complets

A l'inverse des arbres, les graphes complets sont utilisés pour étudier les corrélations fortes entre conflits. Dans ces graphes, chaque communication fait partie de deux conflits qui ont chacun le même nombre de communications. Par exemple, pour un graphe complet à 5 nœuds, chaque conflit comporte 4 communications.

Pour évaluer les modèles sur des graphes complets, nous utilisons 3 graphes à 5 nœuds, soit 10 communications par graphe. Le premier graphe K1 comporte toutes les combinaisons possibles de conflits entre communications entrantes et sortantes. En opposition, le second graphe K2 est constitué uniquement de conflits composés de deux communications entrantes et de deux communications sortantes par nœud. Finalement, le dernier graphe K3 est plus hétérogène en terme de conflits.

***Myrinet* :**
Pour le premier graphe complet MK1, on observe des performances satisfaisantes avec une erreur globale de 8,6%. Les temps des communications sont globalement surestimés, en particulier au niveau des conflits à 4 communications entrantes et 4 communications sortantes.

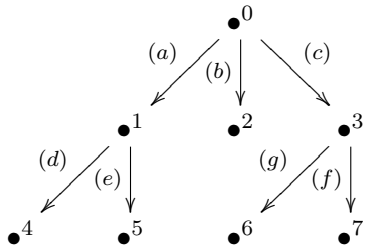
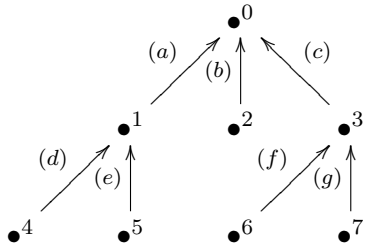
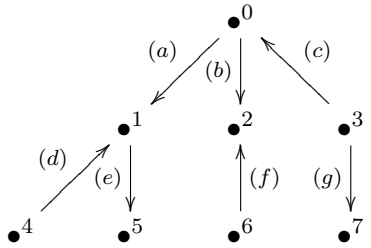
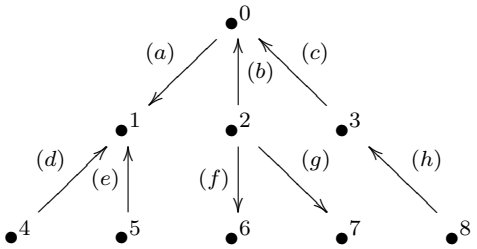
Schémas	Temps des communications			
MA1	com.	T_m	T_p	Erreur (E_{rel})
	a	0.102	0.107	4.9
	b	0.102	0.107	4.9
	c	0.102	0.107	4.9
	d	0.069	0.071	2.9
	e	0.069	0.071	2.9
	f	0.069	0.071	2.9
	g	0.068	0.071	4.4
	Moyenne des erreurs absolues E_{abs} = 4.0			
MA2	com.	T_m	T_p	Erreur (E_{rel})
	a	0.097	0.107	10.3
	b	0.103	0.107	3.9
	c	0.092	0.107	16.3
	d	0.067	0.071	6.0
	e	0.070	0.071	1.4
	f	0.065	0.071	9.2
	g	0.070	0.071	1.4
	Moyenne des erreurs absolues E_{abs} = 6.9			
MA3	com.	T_m	T_p	Erreur (E_{rel})
	a	0.087	0.089	2.3
	b	0.087	0.089	2.3
	c	0.070	0.071	1.4
	d	0.052	0.053	1.9
	e	0.037	0.035	-5.4
	f	0.051	0.053	3.9
	g	0.070	0.071	1.4
	Moyenne des erreurs absolues E_{abs} = 2.6			
MA4	com.	T_m	T_p	Erreur (E_{rel})
	a	0.100	0.107	-8.1
	b	0.121	0.107	-11.6
	c	0.096	0.053	-44.8
	d	0.103	0.107	3.9
	e	0.103	0.107	3.9
	f	0.122	0.107	-12.3
	g	0.121	0.107	-11.6
	h	0.096	0.035	-63.5
	Moyenne des erreurs absolues E_{abs} = 20.0			

FIG. 7.4 – Précision du modèle *Myrinet* : cas des arbres

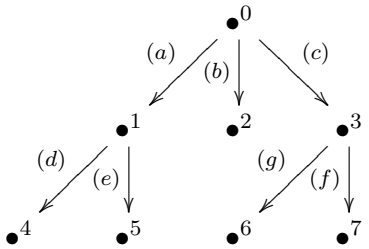
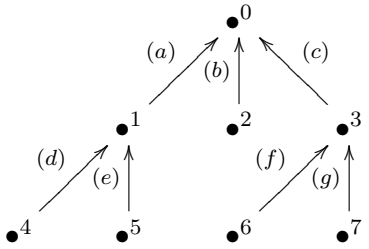
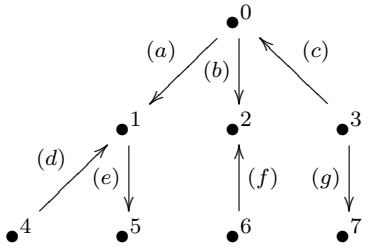
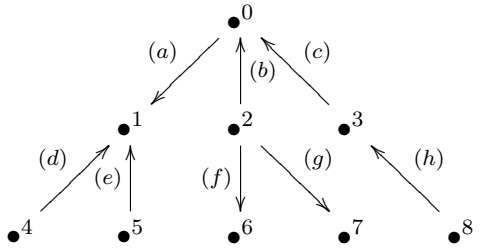
Schémas	Temps des communications			
GA1	com.	T_m	T_p	Erreur (E_{rel})
	a	0.212	0.214	1.0
	b	0.207	0.214	3.4
	c	0.212	0.214	1.0
	d	0.143	0.143	0
	e	0.143	0.143	0
	f	0.143	0.143	0
	g	0.143	0.143	0
	Moyenne des erreurs absolues E_{abs} = 0.08			
GA2	com.	T_m	T_p	Erreur (E_{rel})
	a	0.223	0.214	-4.0
	b	0.170	0.214	25.9
	c	0.226	0.214	-4.0
	d	0.147	0.143	-2.7
	e	0.146	0.143	-2.0
	f	0.147	0.143	-2.7
	g	0.146	0.143	-2.0
	Moyenne des erreurs absolues E_{abs} = 6.0			
GA3	com.	T_m	T_p	Erreur (E_{rel})
	a	0.166	0.147	-11.4
	b	0.164	0.147	-10.4
	c	0.147	0.143	-2.7
	d	0.140	0.137	-2.1
	e	0.131	0.1	-23.7
	f	0.131	0.137	4.6
	g	0.145	0.143	-1.4
	Moyenne des erreurs absolues E_{abs} = 8.0			
GA4	com.	T_m	T_p	Erreur (E_{rel})
	a	0.233	0.214	-8.1
	b	0.205	0.216	5.4
	c	0.132	0.137	3.8
	d	0.205	0.214	4.4
	e	0.209	0.214	2.4
	f	0.213	0.196	-7.9
	g	0.213	0.196	-7.9
	h	0.108	0.1	-7.4
	Moyenne des erreurs absolues E_{abs} = 5.9			

FIG. 7.5 – Précision du modèle *Gigabit Ethernet* : cas des arbres

Pour chaque communication du graphe MK2, les performances observées sont égales. Le modèle de prédiction en fait de même. Cependant, le modèle permet de prédire des temps de communication avec une erreur de 21,1%. Le modèle prédit des pénalités de 2,42 alors que les pénalités mesurées correspondent à des valeurs constantes valant deux.

Sur le dernier graphe MK3, on observe également de bonnes prédictions. L'erreur globale est de 9,5%. Deux communications (e) et (j) ont une erreur plus importante entre 23 et 24%.

Pour conclure, la modélisation est assez efficace pour des graphes complets malgré quelques types de conflits moins bien prédits.

***Gigabit Ethernet* :**

Les temps prédits sur le graphe GK1 sont proches de ceux mesurés avec une erreur globale de 4,3%.

Les valeurs mesurées sur les graphes GK2 et MK2 sont égales. Le modèle permet de prédire des temps identiques mais inférieur d'environ 25%.

Des erreurs de -25,5% et -17,9% sont observées pour les deux communications (e) et (f) sur le graphe GK3. Ces deux communications ont un conflit en réception équivalant aux conflits du graphe GK2. L'erreur absolue globale de ce graphe est de 9,3%.

La conclusion de cette évaluation du modèle *Gigabit Ethernet* est comparable à celle du modèle *Myrinet*. Une seule exception intervient : la modélisation est optimiste pour le graphe GK2 alors qu'elle était pessimiste dans le modèle *Myrinet* pour le graphe MK2.

7.4.3 Graphe de communications hétérogènes

Dans cette section, nous évaluons les modèles sur des graphes de communications hétérogènes en taille. Le graphe est arbitrairement choisi mais comporte des conflits complexes. En effet, il n'est pas possible d'étudier les modèles en fonction de toutes les combinaisons de graphes possibles et de tailles de communications. De plus, le tirage aléatoire des arrêtes entre les nœuds, suivant une loi de probabilité, conduit à des graphes ayant certaines particularités [69][70]. Ainsi, nous avons opté pour un choix arbitraire du graphe, mais en étant conscient qu'un tel choix reste sujet à discussions.

A titre indicatif, nous présentons dans le tableau 7.2 les temps des communications sans conflit en fonction de leurs tailles.

	16 Ko		320 Ko	512 Ko	1 Mo	2 Mo
Myrinet	0.0000031		0.00142	0.00225	0.0044	0.008
Gigabit Ethernet	0.0000337		0.00364	0.00583	0.0116	0.024
	3 Mo	4 Mo	5 Mo	7Mo	8 Mo	16 Mo
Myrinet	0.013	0.017	0.022	0.031	0.036	0.071
Gigabit Ethernet	0.036	0.048	0.056	0.084	0.096	0.193

TAB. 7.2 – Temps des communications seules en seconde

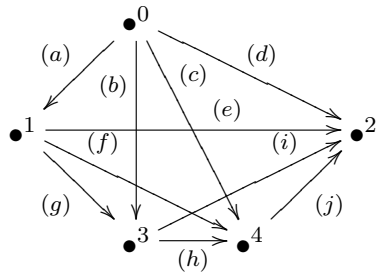
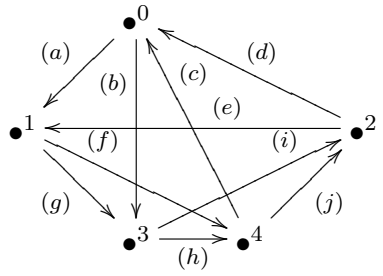
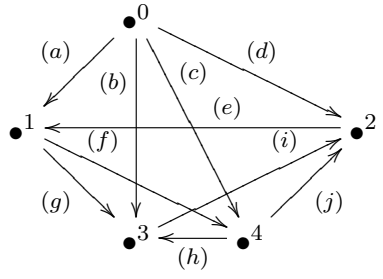
Schémas	Temps des communications			
MK1	com.	T_m	T_p	Erreur (E_{rel})
	a	0.171	0.191	11.7
	b	0.171	0.191	11.7
	c	0.171	0.191	11.7
	d	0.171	0.191	11.7
	e	0.149	0.144	-3.3
	f	0.149	0.144	-3.3
	g	0.149	0.144	-3.3
	h	0.123	0.117	-4.9
	i	0.123	0.117	-4.9
	j	0.083	0.099	19.3
	Moyenne des erreurs absolues E_{abs} = 8.6			
MK2	com.	T_m	T_p	Erreur (E_{rel})
	a	0.071	0.086	21.1
	b	0.071	0.086	21.1
	c	0.071	0.086	21.1
	d	0.071	0.086	21.1
	e	0.071	0.086	21.1
	f	0.071	0.086	21.1
	g	0.071	0.086	21.1
	h	0.071	0.086	21.1
	i	0.071	0.086	21.1
	j	0.071	0.086	21.1
	Moyenne des erreurs absolues E_{abs} = 21.1			
MK3	com.	T_m	T_p	Erreur (E_{rel})
	a	0.164	0.177	7.9
	b	0.164	0.177	7.9
	c	0.164	0.177	7.9
	d	0.164	0.177	7.9
	e	0.043	0.053	23.2
	f	0.086	0.085	-1.2
	g	0.087	0.085	-2.3
	h	0.108	0.101	-6.5
	i	0.108	0.101	-6.5
	j	0.059	0.073	23.7
	Moyenne des erreurs absolues E_{abs} = 9.5			

FIG. 7.6 – Précision du modèle *Myrinet* : cas des graphes complets

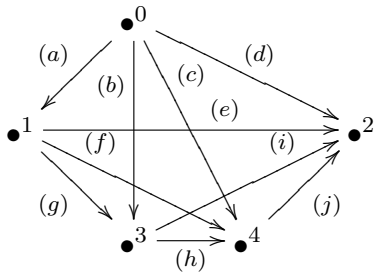
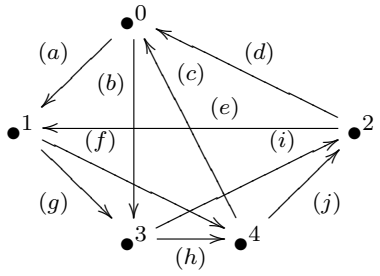
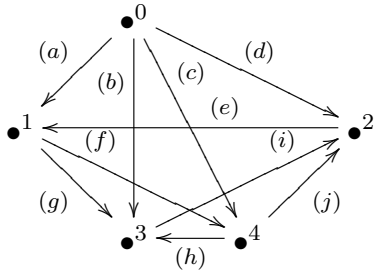
Schémas	Temps des communications			
GK1	com.	T_m	T_p	Erreur (E_{rel})
	a	0.257	0.252	-1.9
	b	0.256	0.252	-1.6
	c	0.270	0.252	-6.7
	d	0.311	0.302	-2.9
	e	0.181	0.189	4.4
	f	0.207	0.206	-0.5
	g	0.274	0.276	0.7
	h	0.182	0.206	13.2
	i	0.258	0.276	7.0
	j	0.287	0.276	-3.8
	Moyenne des erreurs absolues E_{abs} = 4.3			
GK2	com.	T_m	T_p	Erreur (E_{rel})
	a	0.192	0.143	-25.5
	b	0.191	0.143	-25.1
	c	0.191	0.143	-25.1
	d	0.191	0.143	-25.1
	e	0.188	0.143	-23.9
	f	0.190	0.143	-24.3
	g	0.188	0.143	-23.9
	h	0.189	0.143	-24.3
	i	0.188	0.143	-23.9
	j	0.189	0.143	-24.3
	Moyenne des erreurs absolues E_{abs} = 24.5			
GK3	com.	T_m	T_p	Erreur (E_{rel})
	a	0.270	0.285	5.5
	b	0.284	0.304	7.0
	c	0.284	0.304	7.0
	d	0.270	0.285	5.5
	e	0.184	0.137	-25.5
	f	0.167	0.137	-17.9
	g	0.194	0.206	6.2
	h	0.198	0.206	4.4
	i	0.215	0.206	-4.2
	j	0.188	0.206	9.6
	Moyenne des erreurs absolues E_{abs} = 9.3			

FIG. 7.7 – Précision du modèle *Gigabit Ethernet* : cas des graphes complets

Myrinet La figure 7.8 montre que la précision du modèle *Myrinet* est globalement satisfaisante avec une erreur de 14,5%. En outre, nous remarquons que les erreurs relatives les plus importantes correspondent aux communications les plus petites. Ce phénomène s'explique par deux raisons :

- Premièrement, les mesures de l'ordre de quelques millisecondes sont plus sensibles à l'erreur de la mesure. De ce fait, la variabilité des mesures est beaucoup plus importante.
- Deuxièmement, nous utilisons une bande passante (ou gap) fixe comme dans le modèle LogGP. Dans le chapitre 3, nous avons vu qu'un modèle paramétré comme pLoGP est plus efficace dans la prédiction de message de taille moyenne (entre environ 1Ko et 128Ko suivant les réseaux). Ce modèle propose une bande passante fonction de la taille des messages alors que dans nos prédictions, nous appliquons la bande passante maximale obtenue pour de grandes tailles de messages (> 100Mo). Ainsi, une optimisation possible des modèles pourrait s'appuyer sur une nouvelle valeur de la bande passante mesurée pour des messages de taille moyenne.

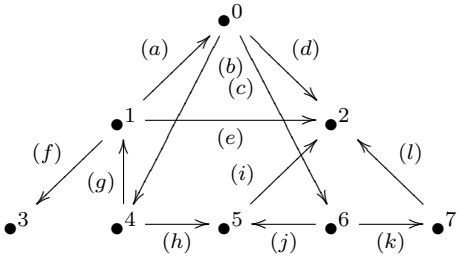
Schémas	Temps des communications			
MH1	com.	T_m	T_p	Erreur (E_{rel})
	a (16Ko)	0.0000068	0.0000075	10.3
	b (2Mo)	0.027	0.029	7.4
	c (4Mo)	0.042	0.047	9.3
	d (8Mo)	0.068	0.069	1.5
	e (512Ko)	0.0050	0.0057	14.0
	f (5Mo)	0.024	0.025	4.2
	g (16Mo)	0.084	0.089	5.9
	h (3Mo)	0.027	0.031	14.8
	i (320Ko)	0.00031	0.00048	54.9
	j (4Mo)	0.030	0.035	16.7
	k (1Mo)	0.0069	0.013	35.4
	l (7Mo)	0.057	0.057	0.1
	Moyenne des erreurs absolues E_{abs} = 14.5			

FIG. 7.8 – Précision du modèle *Myrinet* : graphe de communications hétérogènes

Gigabit Ethernet De manière similaire au modèles *Myrinet*, le modèle *Gigabit Ethernet* propose de bonnes performances avec une erreur globale de 16,6%. Les mêmes constations sur la correspondance entre erreurs importantes et communications de petite taille sont vérifiées. Cependant, *Gigabit Ethernet* étant globalement moins stable que *Myrinet*, ces phénomènes conduisent à une erreur de 150% pour la communication (a). En retirant cette communication, qui par sa petite taille (16ko) n'intervient pas dans la concurrence, l'erreur globale descendrait à 3,8%.

7.4.4 Comparaison avec des repartions simples de la bande passante

Dans cette sous-section, nous comparons les modèles avec des modèles beaucoup plus simples. Les modèles simples calculent les pénalités en fonction du nombre de communications en concurrence. Les degrés des nœuds définis dans le modèle de *Gigabit Ethernet* sont de nouveau utilisés.

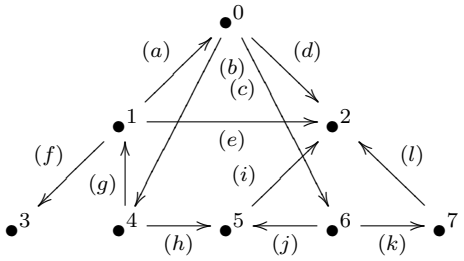
Schémas	Temps des communications			
GH1	com.	T_m	T_p	Erreur (E_{rel})
	a (16Ko)	0.00008	0.0002	150
	b (2Mo)	0.052	0.047	-9.6
	c (4Mo)	0.085	0.079	-7.0
	d (8Mo)	0.158	0.157	0.6
	e (512Ko)	0.0163	0.0156	-4.3
	f (5Mo)	0.066	0.066	0.2
	g (16Mo)	0.216	0.213	-1.4
	h (3Mo)	0.059	0.059	0.1
	i (320Ko)	0.010	0.010	0.3
	j (4Mo)	0.071	0.068	-4.2
	k (1Mo)	0.019	0.016	-15.8
	l (7Mo)	0.121	0.129	6.6
	Moyenne des erreurs absolues E_{abs} = 16.6			

FIG. 7.9 – Précision du modèle *Gigabit Ethernet* : graphe de communications hétérogènes

Nous proposons trois modèles simples pour calculer la pénalité p d'une communication :

- $p = 1$, ce modèle correspond au modèle de Hockney sans concurrence (le modèle logGP donnerait des valeurs similaires) ;
- $p = \Delta_s$, la communication partage de manière proportionnelle la ressource réseau uniquement avec les communications de même origine ;
- $p = \Delta_e$, la communication partage de manière proportionnelle la ressource réseau uniquement avec les communications ayant la même destination.

Ces modèles simples sont intégrés dans le simulateur pour obtenir des prédictions. Nous appliquons ces modèles aux différents graphes synthétiques présentés dans cette section. Le tableau 7.3 montre les erreurs absolues des modèles simples sur ces graphes.

Les fortes erreurs du modèle $p = 1$, sans concurrence, montrent de manière significative l'impact du partage de ressource dans ces réseaux. Toutefois, une modélisation simple, comme celle des deux modèles simples, conduit également à des erreurs importantes. En comparaison, les modèles que nous avons définis sont beaucoup plus précis. Nous remarquons que dans le cas de graphes ayant une certaine régularité dans leurs conflits (graphe MK2 et GK2) ces modèles simples ont de meilleures précisions. Cependant, il s'agit de graphes particuliers, homogènes en conflit. Dans le cas général ces modèles simples sont peu précis.

Cette section montre que les modèles prédisent de manière satisfaisante les communications des graphes synthétiques. Dans la section suivante, nous confrontons les modèles au graphe de l'application *HPL*.

Modèles	$E_{abs}(G)$ sur *Myrinet* [%]						
	MA1	MA2	MA3	MA4	MK1	MK2	MK3
$p = 1$	55.7	54.2	40.7	76.6	74.8	50	62.8
$p = \Delta_s$	4	54.2	15.7	44.7	30.8	0	21.8
$p = \Delta_e$	55.7	6.9	30.7	35.9	33.4	0	31.3
Nos modèles	4	6.9	2.6	20	8.6	21.1	9.5
Modèles	$E_{abs}(G)$ sur *Gigabit Ethernet* [%]						
	GA1	GA2	GA3	GA4	GK1	GK2	GK3
$p = 1$	38.9	42.3	33.7	45.8	60	49.5	55.7
$p = \Delta_s$	37	42.3	26.2	41.7	37.9	1	28.7
$p = \Delta_e$	38.9	37.3	31	35.8	36.4	1	27.4
Nos modèles	0	6	8	5.9	4.3	24.5	9.3

TAB. 7.3 – Erreurs des modèles simples

7.5 Evaluation sur des graphes extraits de l'exécution d'applications

Après avoir évalué les modèles sur des graphes synthétiques, nous considérons des graphes provenant de l'exécution d'applications. Dans cette section, nous utiliserons l'application *benchmark HPL* (présentée dans le chapitre 2) comme base pour l'évaluation des modèles. L'application *HPL* est utilisée pour établir le classement du TOP500. Elle a été amplement étudiée. Par exemple, les auteurs de [18] et les auteurs de [74] proposent un modèle analytique basé sur une multitude de paramètres dont ceux définissant la taille des problèmes. Mais ces modèles ne prennent pas en compte ni le graphe de communication sous-jacent à une exécution ni les effets causés par la concurrence entre communications.

Pour évaluer les modèles avec le graphe *HPL*, nous choisissons d'utiliser la grappe Helios pour les raisons suivantes :

- D'une part, cette grappe comporte des nœuds biprocesseur et bicœur ainsi qu'un réseau *Gigabit Ethernet* et *Myrinet*.
- D'autre part, l'utilisation d'une nouvelle grappe[1] montre la généralité du modèle, en particulier pour le modèle *Gigabit Ethernet*.

Cette section se décompose en deux sous-sections. La première sous-section décrit le graphe *HPL* ainsi que les techniques et outils utilisés pour l'obtenir. La seconde sous-section concerne l'évaluation des modèles sur ce graphe.

7.5.1 Graphes *HPL*

Dans cette sous-section, nous détaillerons deux points :

- Les techniques d'obtention du graphe basés sur le traçage d'événements et leurs mesures ;
- Une analyse détaillée de la structure du graphe obtenu.

[1] Dans l'évaluation des modèles sur des graphes synthétiques, nous avions utilisé la grappe Rennes pour le réseau *Gigabit Ethernet* et la grappe Sophia pour le réseau *Myrinet*

7.5.1.1 Obtention du graphe *HPL*

L'utilisation d'une application ou d'un *benchmark* pour évaluer les modèles réseaux soulève le problème de l'extraction du graphe logique des communications. Ce graphe est caractérisé par deux éléments : les communications (tâche émettrice, tâche destinataire, et taille des messages) et leurs dates de départ. Les dates de départ varient en fonction des calculs et des accès à la mémoire effectués avant la communication. On peut remarquer que le partage de la ressource mémoire a une influence non négligeable dans l'exécution de *HPL*.

Pour extraire le graphe de communications *HPL*, nous traçons l'exécution du programme. Cela permet d'obtenir la suite des événements qui ce sont réellement produits lors d'une vraie exécution. Ce traçage est événementiel autour de deux événements principaux : calcul et communication.

- Un événement de communication est déterminé par l'appel à une procédure *MPI*. Nous considérons uniquement les procédures d'envoi ou de réception bloquante de message d'au moins 1Ko de données. *HPL* peut être paramétré pour obtenir uniquement des communications *MPI* bloquantes.
- Un événement calcul se produit entre deux appels consécutifs à une méthode *MPI*. Un événement de calcul ne comporte que la durée mesurée entre deux événements de communication.

Le traçage d'application *MPI* a déjà fait l'objet de plusieurs études et plusieurs développements. Par exemple, *Intel Trace Analysis and Collector* (*ITAC*) [37] permet d'obtenir des traces uniquement sur des plates-formes à base de processeurs Intel. Un autre exemple est l'outil *Tuning and Analysis Utilities* (*TAU*) [85] qui est principalement dédié à l'analyse et le profilage (*profiling* d'applications. Il contient des outils pour obtenir des traces des appels aux fonctions *MPI* mais ne permet pas de définir un nouveau format de trace (obtention des événements de calcul). Afin de pouvoir obtenir des traces compatibles avec le format des données accepté par le simulateur, nous avons utilisé l'outil *MultiProcessing Environment* (*MPE*) [97] intégré à *Mpich*. Cet outil inclut un mécanisme pour créer un fichier contenant le squelette des fonctions *MPI* encapsulant les appels aux fonctions de communications de *Mpich*. Il est alors possible d'ajouter du code avant et après ces appels afin d'en obtenir une trace. De cette manière les appels aux fonctions *MPI* de l'application exécutent les fonctions définies dans ce nouveau fichier. Cet outil nous a permis d'obtenir les traces escomptées.

7.5.1.2 Exemple de graphe

L'exemple de graphe choisi correspond à un placement aléatoire des tâches *MPI* sur 8 nœuds avec 2 tâches par nœud. Le nombre de tâches *MPI* est de 16, ce qui permet d'avoir au maximum 8 communications bloquantes simultanément. Un processeur exécute une seule tâche. Le réseau utilisé est le réseau *Myrinet* de la grappe Helios.

La taille du problème choisi est un élément déterminant dans la caractérisation du graphe de communications. Nous optons pour une taille de problème[2] de 12500.

[2]Une explication détaillée de l'algorithme *HPL* et des tailles de problèmes associées est accessible à l'adresse suivante `http://www.netlib.org/benchmark/hpl/algorithm.html`

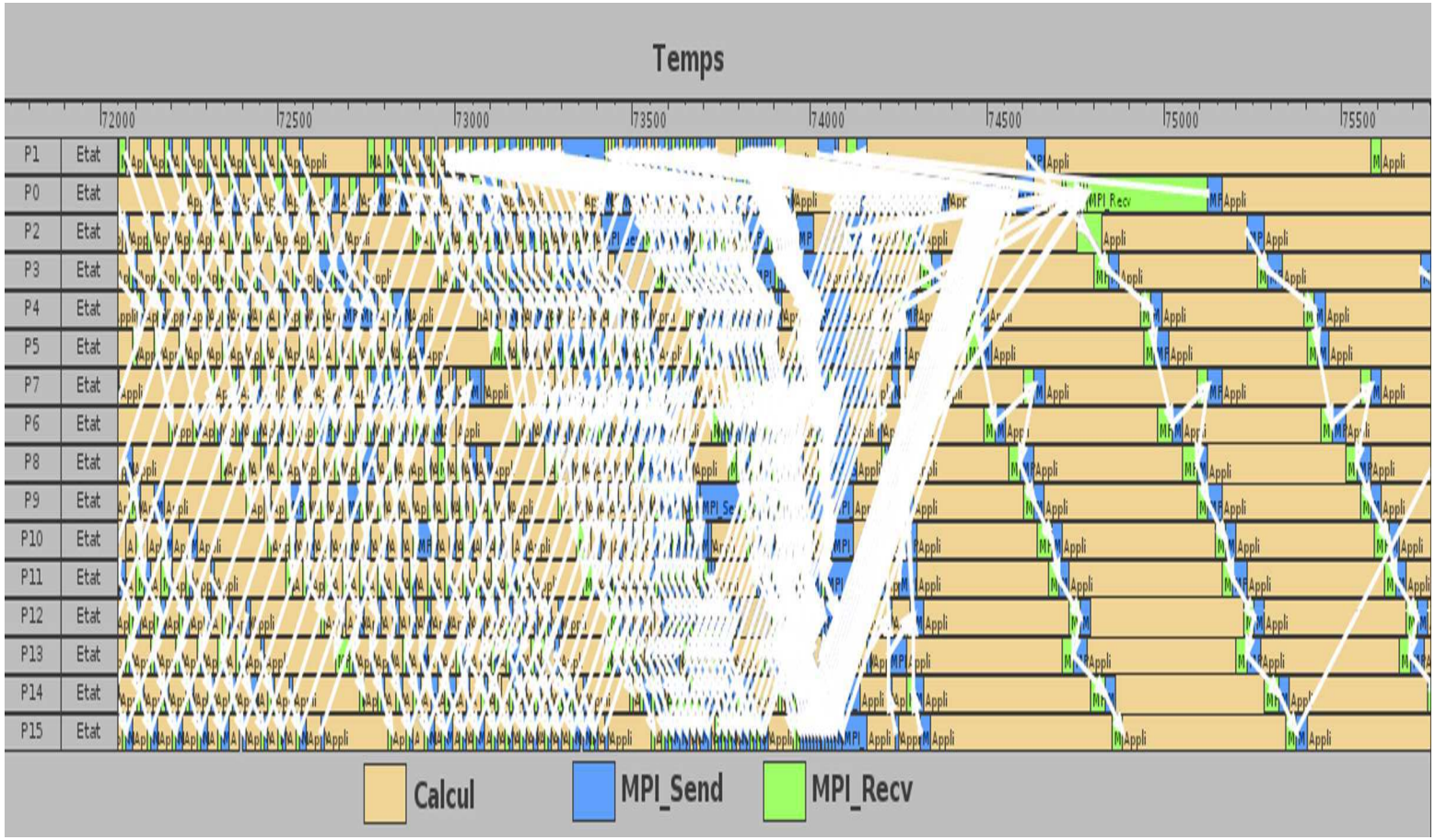


FIG. 7.10 – Visualisation d'une partie du graphe *HPL* de communication entre deux itérations

Le tableau 7.4 affiche des statistiques relatives aux graphes *HPL* de taille 12500 mais aussi d'une taille de problème de 20500. Le graphe, obtenu pour un problème de taille 12500, comporte 9767 événements en moyenne par tâches pour un total cumulé de 156273 événements. Ces événements sont répartis en moyenne entre 6683 événements de communication et 3084 événements de calcul. Le volume cumulé des communications est de 158 Go (soit environ 9,8 Go par tâche). Chaque tâche passe en moyenne 546 secondes en phase de calcul et 104 secondes en phase de communication (comprenant les événements en émission et réception).

Une partie d'un graphe *HPL* est présentée, dans la figure 7.10, sous forme graphique en utilisant l'outil Pajé [8]. Cette figure décrit les communications entre deux itérations. Elle montre une certaine régularité. Le placement des tâches sur les nœuds peut apporter une certaine complexité dans les schémas de communications.

La figure 7.11 montre la matrice de communication entre les tâches. Les communications s'effectuent essentiellement entre tâches de rangs successifs, et pour une quantité de données totale d'environ 10Go. La topologie est une topologie en anneau. Ce graphe intègre aussi d'autres communications entre tâches non voisines mais d'une quantité négligeable d'environ 5Mo.

La durée d'exécution de *HPL* pour une taille de 12500 varie aux alentours de 10 minutes suivant le placement et le réseau. Une exécution avec une taille de 20500 varie aux alentours de 45 minutes. Finalement, le coût induit par le traçage global sur l'exécution de l'application est négligeable comme le montre le tableau 7.5.

Problèmes	Moy. du nb événs. calculs / coms	Temps moy. [s] calculs / coms	Volume moy. des coms. [Go]
12500	3084 / 6683	546 / 104	9.8
20500	5375 / 10901	2416 / 275	26.5

TAB. 7.4 – Statistiques par tâche sur le graphe *HPL*, taille 12500, grappe Helios, réseau *Myrinet*

Problèmes	Temps sans trace	Temps avec trace	Coût	Taille de la trace
12500	675 s	680 s	0.7 %	5.1 Mo

TAB. 7.5 – Coût du traçage pour 16 tâches, grappe Helios, réseau *Myrinet*

7.5.2 Evaluation

Pour évaluer les réseaux *Myrinet* et *Gigabit Ethernet*, pour chaque tâche, nous comparons les temps mesurés et prédits sur les graphes *HPL* (taille 12500 et 20500). Ces exécutions se font autour de 16 tâches. Ces tâches sont placées sur 4 ou 8 nœuds. Le placement des tâches détermine différentes propriétés (nombre de communications en concurrence, nombre de communications intra-nœud) du graphe de communications. Nous choisissons des placements qui conduisent à des graphes couvrant un ensemble représentatif de ces propriétés. L'utilisation de 8 nœuds comportant 2 tâches par nœuds est notée 8x2, l'utilisation de 4 nœuds et 4 tâches par nœuds est notée 4x4.

Un placement définit la correspondance entre les rangs des tâches *MPI* et le numéro des processeurs. Le placement des tâches peut se faire de différentes manières. Une politique de placement permet de définir un ensemble de placements. Nous utiliserons trois politiques de placement :

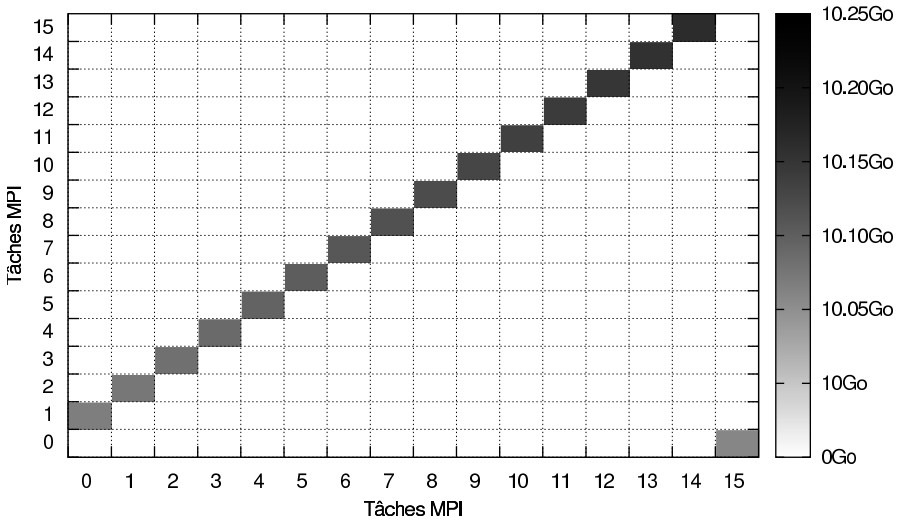


FIG. 7.11 – Matrice de communication, 16 tâches, taille 12500

– politique “Round Robin par Processeur” (RRP) : les tâches sont assignées en remplissant en premier les nœuds ;
– politique “Round Robin par Nœud” (RRN) : les tâches sont assignées de manière cyclique entre les nœuds ;
– politique aléatoire (Aléa) : les tâches sont distribuées aléatoirement sur les nœuds.

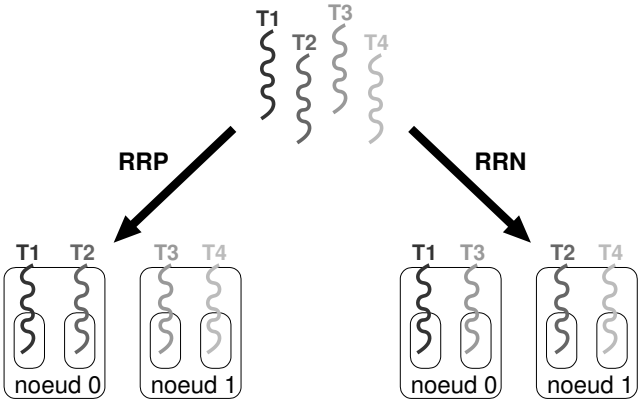


FIG. 7.12 – Politiques de placement

La figure 7.12 décrit un exemple de placement suivant les politiques RRP et RRN. Ainsi, l'utilisation de la politique RRP sur 4 nœuds comportant 4 tâches est notée : RRP(4x4).

Avant de décrire la précision des modèles au travers des valeurs des erreurs, nous présentons un tableau comportant différentes statistiques sur les comportements réseaux prédits des graphes. Ce tableau 7.6 montre les indices statistiques obtenus par la simulation :

– Le nombre de communications suivant trois catégories : les communications intra-nœud (N_a), les communications inter-nœud non concurrentes (N_{nc}) et les communications inter-

nœud concurrentes (N_c). Ces indices quantifient le comportement général des graphes. Par exemple, un graphe ayant beaucoup de communications intra-nœud demande peu de ressource réseau.

– Une série d'indices relative aux conflits et à leurs pénalités permet de caractériser l'impact de la concurrence sur les graphes. Ces indices sont : la pénalité moyenne de l'ensemble des communications en conflit durant la simulation (P_m), le nombre de conflits sachant qu'une communication peut faire partie de plusieurs conflits durant son exécution (N_{conf}), le nombre moyen de communications par conflit (N_{com}), et la pénalité moyenne du conflit le plus pénalisant (P_{Cmax}).

Les indices statistiques nous permettent de classer les expériences en trois catégories :

– Les graphes RRN(8x2) et RRN(4x4) ont peu de communications intra-nœud et un nombre de communications inter-nœud concurrentes significatif. Suivant le réseau, ces graphes génèrent des conflits qui en moyenne ont environ deux communications. Pour ces graphes, le réseau *Myrinet* pénalise plus les communications en conflit que le réseau *Gigabit Ethernet*.

– Les graphes Aléa(8x2) et Aléa(4x4) sont plus homogènes en termes de communications intra-nœud et inter-nœud concurrentes et non concurrentes. Leurs pénalités moyennes sont proches des graphes précédents, mais ils contiennent des conflits plus importants.

– Les graphes RRP(8x2) et RPP(4x4) n'introduisent pas de communications concurrentes. Ces deux graphes nous serviront de référence pour mettre en évidence la validité du choix de l'implantation de la sémantique synchrone des communications, sans être influencé par les erreurs des modèles. Les graphes RRP(8x2) ont environ le même nombre de communications intra-nœud et inter-nœud, alors que les graphes RRP(4x4) a nettement plus de communications intra-nœud.

Problèmes			12500			20500		
			RRN(8x2)	Aléa(8x2)	RRP(4x4)	RRN(4x4)	Aléa(4x4)	RRP(8x2)
Myrinet	Coms.	N_a	54	9990	40140	162	11016	43632
		N_{nc}	41566	34730	13320	68132	59430	43578
		N_c	11840	8740	0	18916	16764	0
	Conflits	P_m	1.67	1.69	0	1.82	1.80	0
		N_{conf}	8152	6324	0	14525	12955	0
		N_{coms}	2.02	2.07	0	2.15	2.19	0
		P_{Cmax}	2.0	3.0	0	4.0	4.0	0
Gigabit Ethernet	Coms.	N_a	54	9990	40140	162	11016	43632
		N_{nc}	46853	29941	13320	56647	45756	43578
		N_c	6553	13529	0	30401	30438	0
	Conflits	P_m	1.29	1.29	0	1.35	1.35	0
		N_{conf}	5813	10812	0	24684	23919	0
		N_{coms}	2.03	2.08	0	2.11	2.19	0
		P_{Cmax}	1.42	1.42	0	2.18	3.16	0

TAB. 7.6 – Statistiques des graphes *HPL*

Les temps présentés correspondent à la somme des temps passés en phase d'émission. Ils sont

confrontés à la somme des temps prédits durant ces mêmes phases. Nous utilisons une approche globale pour évaluer les prédictions des milliers de communications.

***Myrinet* :**

Les figures 7.13 et 7.14 montrent les résultats obtenus sur le réseau *Myrinet*. Les graphes RRP(4x4) et RPP(8x2), avec aucune communication concurrente, permettent de valider le processus de simulation en inhibant les erreurs produites par les modèles prédictifs. Les erreurs obtenues sont faibles et restent globalement inférieures à 5%. Cependant, certaines tâches ont des erreurs plus importantes entre 10% et 15%. Ces tâches contiennent uniquement des communications intra-nœuds. Nous pouvons supposer que cette augmentation de l'erreur est due à des effets de congestion mémoire provoqués par les phases de calcul des autres tâches sur le même nœud. La congestion mémoire implique une dégradation de la bande passante allouée aux communications intra-nœud. Cette augmentation touche plus de tâches dans le graphe RRP(4x4) qui regroupe 4 tâches par nœud. Cependant, nous observons que la simulation a un comportement très proche de la réalité confortant notre choix d'implantation de la sémantique synchrone.

Les graphes RRN(8x2) et RRN(4x4) comportent peu de communications intra-nœuds et un nombre de communications en concurrence significatif. La simulation et les modèles utilisés montrent qu'ils sont performants dans la prédiction du comportement réseau *HPL*. Aucune tâche n'a une erreur supérieure à 10%. Dans le cas du graphe RRN(8x2), les temps de chaque tâche sont équivalents, mais ils deviennent beaucoup plus hétérogènes dans le graphe RRN(4x4) (avec un écart maximum d'environ 1 minute). Le modèle reproduit très précisément ces deux comportements.

Finalement, la conjonction entre communications intra-nœuds et communications concurrences montre des résultats aussi très satisfaisants. Toutefois, comme précédemment, les communications intra-nœuds sont moins bien prédites avec pour certaines tâches des erreurs approchant les 20%. Ces effets sont probablement dûs à la congestion mémoire qui n'est pas prise en compte dans la simulation.

Nous concluons que le modèle pour *Myrinet* est globalement précis et les erreurs légèrement plus importantes sont causées par la congestion mémoire.

***Gigabit Ethernet* :**

Les figures 7.15 et 7.16 présentent les prédictions du modèle *Gigabit Ethernet*.

Les graphes RRP(4x4) et RRP(8x2) montrent des erreurs satisfaisantes. Ces deux graphes ne contiennent pas de communications concurrentes. Il est intéressant de noter que la modification de l'implantation de la sémantique synchrone dans le cas de communications intra-nœuds s'avère justifiée.

Lorsque les communications intra-nœuds ont un impact négligeable, comme pour les graphes RRN(8x2) et RRN(4x4), nous remarquons que l'erreur est également satisfaisante. Les erreurs varient aux environs de 10%.

Lorsque les graphes sont hétérogènes entre communications, graphes Aléa(8x2) et Aléa(4x4), l'erreur devient très irrégulière et plus importante (parfois supérieure à 20 %). Cette déviation entre prédictions et mesures est amplifiée par la propagation de l'erreur[10] et la modification des conflits. Effectivement, il suffit que le modèle réduise ou augmente le temps d'une communica-

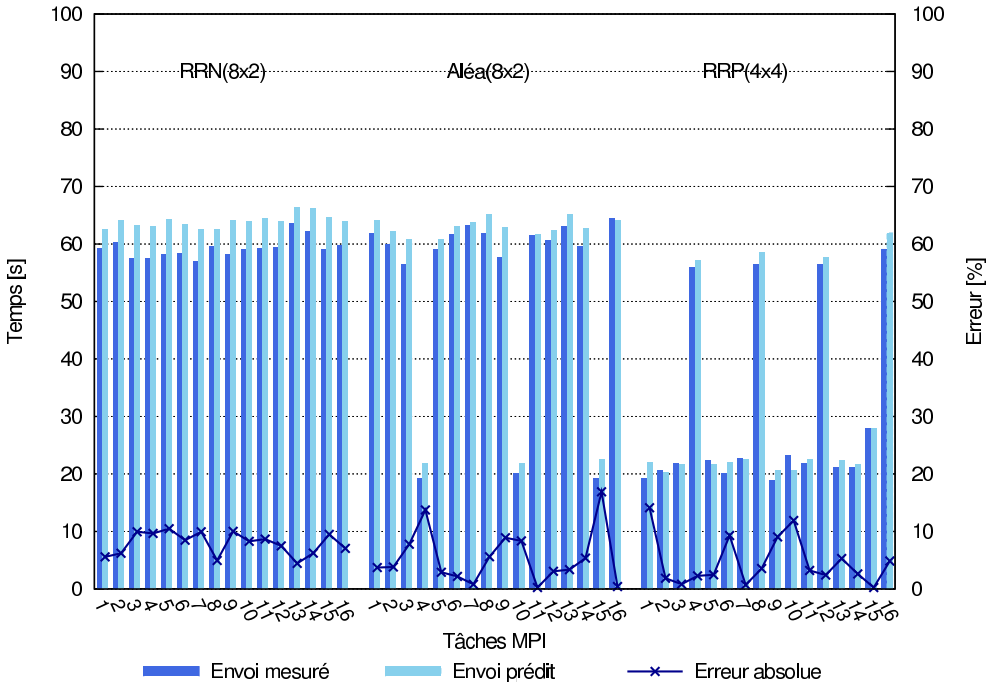


FIG. 7.13 – Evaluation du modèle *Myrinet* sur des graphes *HPL*, taille 12500

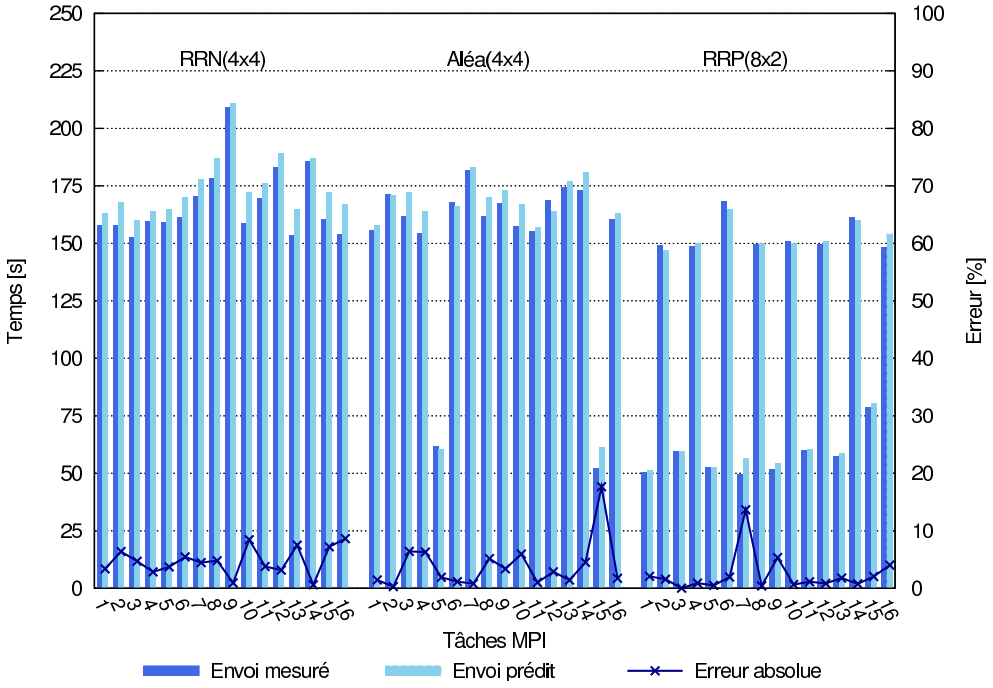


FIG. 7.14 – Evaluation du modèle *Myrinet* sur des graphes *HPL*, taille 20500

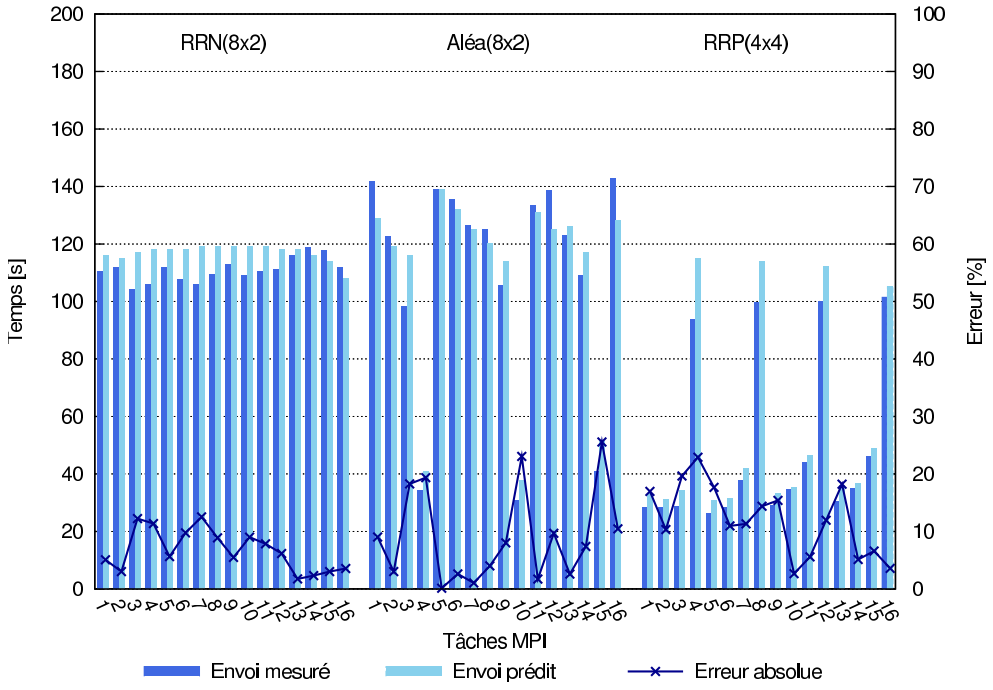


FIG. 7.15 – Evaluation du modèle *Gigabit Ethernet* sur des graphes *HPL*, taille 12500

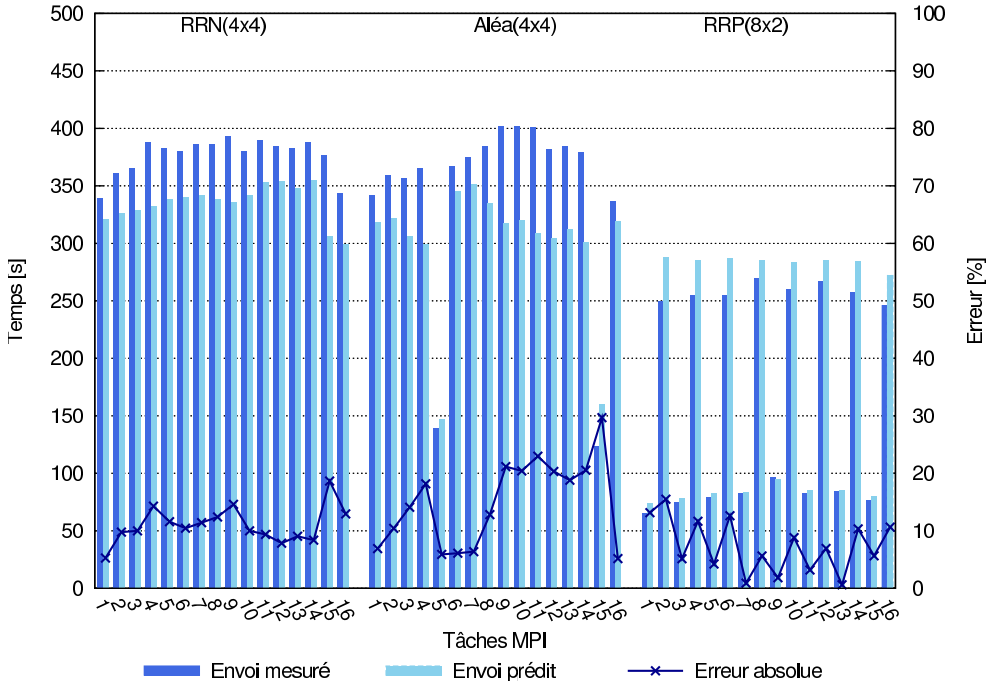


FIG. 7.16 – Evaluation du modèle *Gigabit Ethernet* sur des graphes *HPL*, taille 20500

tion pour ainsi décaler dans le temps tous les événements suivants. Ce décalage peut provoquer l'apparition ou la disparation de conflits par rapport à ceux qui sont réellement produits dans le graphe mesuré. Cette déviation a pour conséquence directe d'amplifier les valeurs des erreurs. Dans le cas de *HPL*, chaque tâche communique essentiellement avec la tâche suivante, une forte erreur sur une tâche affectera les conflits avec ces tâches voisines en augmentant ou diminuant les temps prédits. On peut trouver une explication vraisemblable des erreurs du graphe Aléa(4x4) dans les effets de la propagation de l'erreur. Les quatre premières tâches ont une erreur croissante, puis une tâche mieux prédite avec une majorité de communications intra-nœuds réduit cette erreur. Les deux tâches suivantes ont également une faible erreur, la propagation de l'erreur n'intervient que faiblement. A partir des tâches numéro 8 et 9, l'erreur augmente. Ces tâches moins bien prédites réamorcent le phénomène de propagation de l'erreur qui influencera l'ensemble des tâches restantes (à l'exception de la dernière tâche).

Les résultats pour *Gigabit Ethernet* sont légèrement moins précis que pour *Myrinet*. Ces écarts peuvent s'expliquer par la plus grande variabilité des comportements sur le réseau *Gigabit Ethernet*. A l'inverse de *Myrinet*, *Gigabit Ethernet* n'utilise pas des techniques telles que des accès *RDMA* ou *OS-bypass*. Ces techniques ont pour effets de limiter, par exemple, l'influence du système d'exploitation dans la gestion des communications.

7.6 Optimisations

En analysant les expériences sur les communications concurrentes et l'évaluation des modèles, nous nous sommes aperçus de certains effets sur les performances. Cette section traite de l'analyse de ces effets afin d'augmenter, à moindre coût, certains critères de performance. Dans une première sous-section, nous décrivons comment il est possible d'optimiser des communications sur des réseaux haute performance. Dans la seconde sous-section, nous étudierons les effets relatifs au placement des tâches *MPI* sur la grappe.

7.6.1 Découpage de message

Le découpage de message (*message stripping*) consiste à la décomposition d'une communication de grande taille en plusieurs communications de taille inférieure. Par exemple, un appel à la fonction *MPI_Send* avec 8Mo de données se transforme en 256 appels de cette même fonction avec des messages de 32Ko. Nous appelons segment la taille des messages une fois découpés (le segment est de 32Ko dans l'exemple précédent). Cette technique a été introduite dans [7] au début des années 1990. Les auteurs de [17] l'ont repris en l'incluant comme une option lors du processus de compilation (de manière similaire au déroulage de boucle). Ils comparent les résultats obtenus sur les réseaux *Quadrics* et *Myrinet* et montrent un gain de performance non négligeable. Toutefois, ils discutent peu de la probable origine d'un tel phénomène.

Intuitivement, on peut envisager qu'une telle découpe est contre performante : chaque nouveau message additionne au temps total une latence réseau et un temps de découpe. Néanmoins, pour certains réseaux comme *Myrinet*, nous avons vu que le protocole par rendez-vous de *MPI* provoque une forte augmentation du temps de communication (confère chapitre 4, figures 4.10, 4.11 et 4.12). Cette augmentation peut être évitée en envoyant des messages de taille plus petite.

Le découpage de message apporte une contrainte supplémentaire. Pour découper des données, il est nécessaire qu'elles soient contiguës en mémoire. Il est obligatoire pour des données dispersées en mémoire de faire appel à des méthodes qui copient les données dans des tampons contigus. Le standard *MPI* définit de telles méthodes : *MPI_Pack* et *MPI_Unpack*. Lors de l'appel à ces méthodes la question du typage des données est résolue en allouant des tampons contigus de type *MPI_BYTE* ou *MPI_PACKED* (type unitaire codé sur 8 bits). Ainsi, le coût du découpage inclut également le temps pour copier les données vers un espace contigu en mémoire. De ce fait, il ne permet pas d'utiliser le recouvrement des communications par du calcul.

Nous étudions, dans une suite d'expériences, le gain obtenu par le découpage suivant deux axes : l'influence de la taille du segment sur le gain de performance, et l'impact du découpage sur les communications concurrentes. Ces expériences utilisent un réseau *Myrinet*.

La figure 7.17 montre le gain (E_{rel}) obtenu en fonction de la taille du segment lors de l'envoi d'une communication seule de 8Mo. Le temps mesuré pour les communications découpées inclut le temps de découpage. Les données sont préalablement contiguës en mémoire (tableau d'entiers). Le gain est relativement important entre 10 et 20 %. Ce gain augmente avec la taille du segment jusqu'à la limite de 32 Ko, au-delà il y a surcoût. Cette taille limite correspond au changement de protocole sur *Myrinet* couplé à *Mpich*. Ainsi la taille de segment optimale est de 32Ko. Nous utilisons une segmentation suivant cette taille dans la suite des expériences.

Nous abordons l'influence du découpage sur des communications en concurrence. Le modèle de concurrence de *Myrinet* stipule que les communications de tailles inférieures à 32 Ko ne subissent pas de pénalités lors de conflits. Ainsi, en découpant les communications concurrentes suivant cette taille, le gain obtenu devrait suivre celui obtenu pour une communication seule. Pour mesurer ce gain, nous utilisons un conflit S/S à deux et quatre communications. Nous évaluons la moyenne des temps de communication du conflit pour calculer le gain.

La figure 7.18 décrit le gain en fonction de la taille des communications et des conflits. Nous remarquons que ce gain diminue. Il atteint 95% pour une taille de message de 512Ko. Les gains obtenus pour des communications en conflit présentent également ce caractère décroissant conformément aux propositions du modèle. Le gain diminue avec l'augmentation de la taille des messages. Le coût du découpage et le nombre de messages deviennent alors trop importants provoquant la perte des effets bénéfiques.

Nous avons aussi fait la même étude sur *Gigabit Ethernet*, mais sur ce réseau le découpage n'apporte aucun gain. En effet, le protocole *TCP* découpe les messages en trame d'environ 1500o. Le découpage n'apporte dans ce cas-ci qu'un coût supplémentaire.

Pour avoir un aperçu sur les performances gagnées, nous modifions notre programme de traçage pour incorporer cette technique. La figure 7.19 compare les performances (nombre de *Gflops*) de *HPL* avec et sans cette optimisation pour deux placements des tâches : RRN et aléatoire. A l'inverse des expériences précédentes, les données de *HPL* ne sont pas contiguës en mémoire. Pour les deux placements, les expériences montrent que cette technique montre une augmentation significative des performances de *HPL*. Toutefois, à partir d'une taille de problème supérieure à 11000 le gain est quasiment nul, et à partir de 15500, cette technique diminue les performances de *HPL*. Il s'avère que les tailles des communications augmentent avec la taille des problèmes. Comme précédemment, cela a pour effet de contre balancer le gain jusqu'à ralentir l'application.

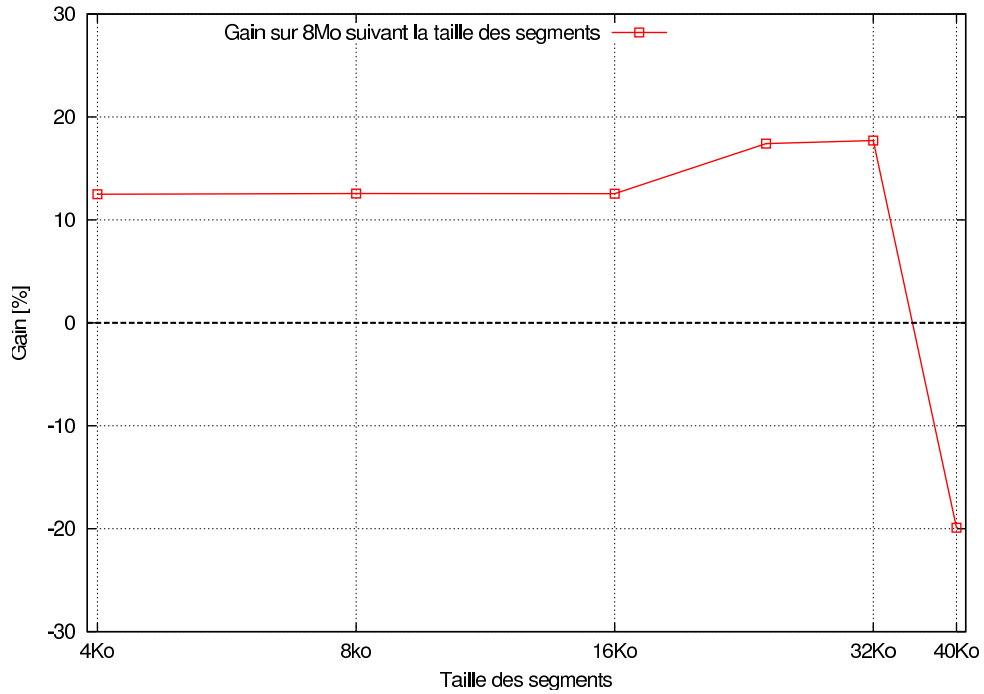


FIG. 7.17 – Variation des gains sans conflit, communications de 8Mo, *Myrinet*

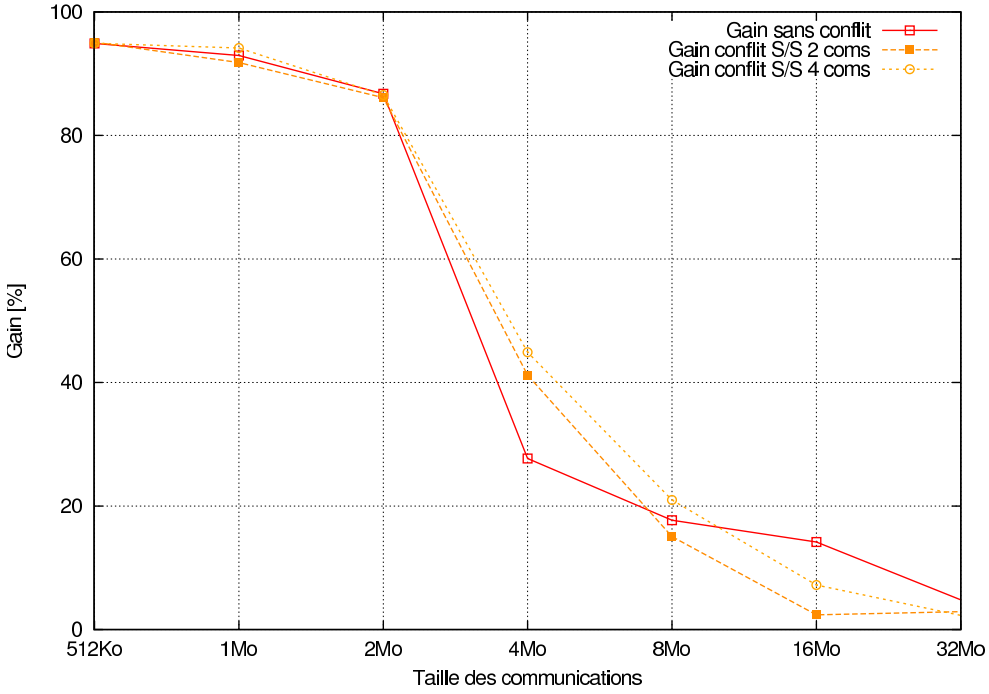


FIG. 7.18 – Variation des gains en fonction des conflits, taille des segments de 32Ko, *Myrinet*

Finalement, il serait intéressant d'analyser les performances d'une telle option dans une libraire de communication comme [67], en particulier pour *Myrinet*. Il conviendrait alors d'ajouter aux mécanismes d'agrégation de petits messages des mécanismes de découpage pour des messages de taille moyenne. Une telle librairie convergerait dans une certaine mesure (en fonction de la taille des données à transmettre ou le temps de découpe) à envoyer des messages de tailles constantes quel que soit les requêtes des applications.

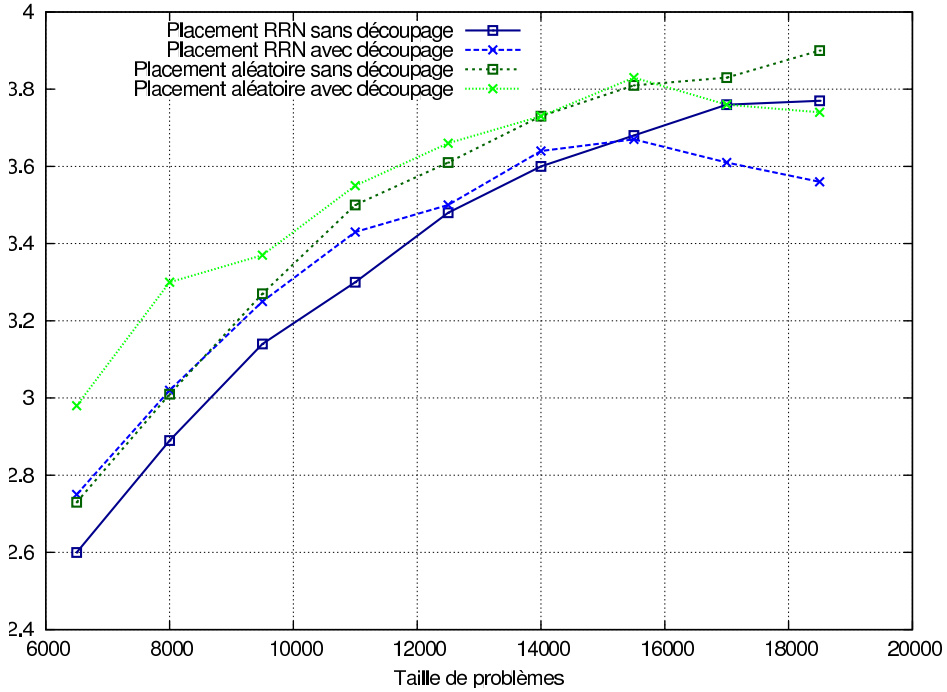


FIG. 7.19 – Performance de *HPL* (*Gflops*) avec et sans découpage, *Myrinet*

7.6.2 Placement des tâches

Nous avons remarqué précédemment que le placement des tâches sur les nœuds a un impact significatif sur les performances. Ce phénomène a aussi été observé par les auteurs de [89]. Dans leurs articles, ils étudient l'influence du placement des tâches de l'application *HPL*. Ils aboutissent à la conclusion qu'il est préférable de placer sur le même nœud des tâches ayant des communications de grandes tailles entre elles. Ils montrent que cette stratégie est nettement plus efficace que le regroupement des tâches ayant une forte fréquence de communications de petites tailles. En utilisant les modèles, nous confortons leurs conclusions. En effet, des communications intra-nœuds ne sont pas génératrices de conflits concurrents. De plus, la bande passante intra-nœud est plus grande et pour optimiser les temps de l'application devrait être le plus souvent utilisée. Il s'avère que dans *HPL* le volume total des communications de grandes tailles est plus important que celui des communications de petites tailles.

Il est possible de généraliser cette étude en définissant un ou plusieurs critères de performance pour un placement. Par exemple, un critère pourrait tenir dans l'utilisation de la bande passante

intra-nœud et un placement efficace optimiserait au maximum son utilisation. Les modèles interviennent parce qu'ils permettent de calculer les valeurs de tels critères pour des placements données. Nous retenons deux critères :

- le critère V_{int} : le volume de communication intra-nœud ;
- le critère V_{pen} : le volume de données équivalant à la quantité de données qui aurait pu être envoyée durant les phases d'attentes induites par les pénalités sur les communications en conflit.

Un placement efficace est donc un placement qui maximise V_{int} et minimise V_{pen}. Toutefois, il est à noter que les phénomènes de congestion mémoire peuvent pénaliser les communications intra-nœuds et diminuer le gain obtenu en cherchant à maximiser V_{int}.

Même si par les modèles on peut déterminer les valeurs de ces deux critères pour un placement donné, nous ne pouvons raisonnablement pas déterminer le placement optimal. En effet, le nombre de placement possible pour p tâches est $p!$ et une approche exhaustive n'est pas envisageable pour de grandes valeurs de p.

A titre d'exemple, nous présentons dans le tableau 7.7 la corrélation entre les temps d'exécution *HPL* et le nombre de *Gflops* observé et les critères de performance des placements. Nous rappelons qu'un graphe *HPL* de taille 12500 génère un volume total de 158 Go. Les variations des critères choisies s'accordent avec les performances.

Placement RRP/4x4		Placement RRN/4x4		Placement Aléatoire/4x4	
V_{int}	V_{pen}	V_{int}	V_{pen}	V_{int}	V_{pen}
75 % (118 Go)	0 %	0.01 % (15 Mo)	31 % (49 Go)	25 % (39 Go)	15 % (23 Go)
Temps	*Gflops*	Temps	*Gflops*	Temps	*Gflops*
610 s	3.9	693 s	3.4	680	3.55

TAB. 7.7 – Corrélations entre placement et performance de *HPL*, réseau *Myrinet*, taille 12500

Nous concluons en précisant que ces critères peuvent être appliqués à l'évaluation d'un algorithme d'opérations collectives, comme cela a été proposé par [28]. Par exemple, une diffusion (*MPI_Broadcast*) utilisant un arbre binomial a des performances variables selon où se situe la racine de l'arbre. Il devient aussi intéressant d'évaluer l'opération d'échange total (*MPI_AlltoAll*) car elle génère énormément de communications concurrentes.

7.7 Bilan

Ce chapitre a pour objectif de démontrer la précision des prédictions obtenues par les modèles. Dans un premier temps, nous avons vu que les modèles prédisent de façon satisfaisante les temps de communications prises dans des conflits formés par des graphes synthétiques. Dans un second temps, nous avons étendu cette analyse aux graphes provenant d'applications. Pour extraire des applications les graphes de communications, nous avons présenté une méthode par traçage et explicité un graphe de l'application *HPL*. Cette méthode met en évidence deux problèmes : l'influence de la ressource mémoire et la propagation de l'erreur. Ces deux problèmes sont responsables de la perte de précision des modèles sur des graphes extraits d'applications. Cependant, les prédictions sous ces conditions restent satisfaisantes.

Conclusion et perspectives 8

Cette dernière partie clôt ce document en rappelant les objectifs et la démarche proposée dans cette thèse.

8.1 Objectifs de la thèse

Nos travaux se sont intéressés à l'étude de la congestion réseau et, particulièrement, au comportement des communications concurrentes sur des réseaux haute performance. Le problème de la congestion réseau est fréquemment abordé de manière globale. Une approche globale est intéressante pour l'étude de réseaux avec une large couverture où la dépendance entre communications est difficile à observer. Ce type de réseau correspond au réseau *WAN* ou Internet qui comporte plusieurs acteurs émettant des communications sans une coordination volontaire.

Dans le cadre de réseaux dédiés, comme les réseaux haute performance, nous abordons ce sujet avec une approche plus fine qui se traduit par l'étude locale de la concurrence entre communications. Nous évoquons en particulier la notion de conflits réseaux et de communications concurrentes à la place de la notion générique de congestion réseau. Suivant cette orientation, nous proposons une étude de la concurrence qui nous a mené à la définition de modèles prédictifs.

Dans un contexte industriel, il est intéressant de pouvoir prédire les comportements concurrents des applications suivant plusieurs architectures réseaux. La définition de plusieurs modèles réseaux permet de comparer les performances des applications entre architectures réseaux et d'aider l'industriel à proposer une solution adaptée. Dans cette thèse, nous proposons un simulateur, support pour le calcul des prédictions de modèles, et deux modèles réseaux. Nous avons également montré que la prédiction des phénomènes concurrents est un élément important dans la prédiction de performance. D'une part, elle permet d'identifier les pertes de performances des applications causées par les conflits, les pénalités, ou par le placement des tâches sur les machines. D'autre part, elle apporte une meilleure compréhension du phénomène de concurrence. Elle permet de corréler les pertes de performances à certains détails des mécanismes réseaux (lien full-duplex, contrôle de flux, carte réseau, commutateur, etc).

8.2 Démarche proposée

La démarche proposée s'articule suivant trois parties.

La première partie consiste en une étude générique des dispositifs influençant les performances des applications sur les grappes. La description à la fois matérielle et logicielle de ces composants met en évidence la complexité pour en prédire les comportements résultant de leurs utilisations. Cette partie a été approfondie en incluant les différentes approches définissant des modèles prédictifs caractérisant les comportements réseaux.

La seconde partie est dédiée, plus particulièrement, à l'étude du partage de la ressource réseau entre tâches d'une même application. Elle consiste en une analyse approfondie des comportements réseaux. Cette analyse est réalisée à partir d'un ensemble d'expérimentations sur différentes architectures réseaux. L'objectif de ces expériences est la formulation de modèles prédisant les effets de la concurrence entre communications.

La dernière partie évalue les modèles proposés et introduit leurs utilisations dans la prédiction des comportements réseaux des applications réelles. Cette évaluation propose également une étude préalable sur la généralité des modèles face aux différents éléments constituant les réseaux d'une grappe. Le caractère complet de cette évaluation est obtenu par la comparaison entre valeurs mesurées et valeurs prédites pour des graphes de communications synthétiques et des graphes provenant d'applications. L'utilisation des modèles dévoile également certaines optimisations qui augmentent les performances réseaux des applications.

8.3 Travaux réalisés

Les principaux travaux de cette thèse consistent en l'expression de phénomènes réseaux concurrents et en leurs modélisations. Un ensemble d'expérimentations complexes a mis en évidence le caractère inhérent au partage de ressources des réseaux *Quadrics*, *Myrinet* et *Gigabit Ethernet*. Ces expérimentations montrent que chaque architecture réseau partage la bande passante entre communications concurrentes de manière différente. Dans le chapitre 4, nous décrivons une partie des expériences menées qui nous ont permis d'aboutir à la définition de modèles prédictifs. Plusieurs de ces expériences ont été effectuées pour mettre en évidence les influences de certains composants réseaux. Le chapitre 6 expose les résultats produits par les expériences les plus pertinentes. Les conclusions de ces expériences ont permis une compréhension plus fine des effets des conflits réseaux sur les communications. Cette étude a abouti à la modélisation des phénomènes de partage suivant deux modèles.

La modélisation ainsi obtenue s'organise suivant deux modèles.

Un premier modèle spécifique est défini pour chaque architecture réseau. Il détermine le partage de la bande passante réseau entre communications par le calcul de coefficients de ralentissement que nous avons appelé pénalités. La prédiction des pénalités peut être obtenue en décrivant les phénomènes responsables de leurs apparitions. Il est également envisageable de les calculer en quantifiant les effets provoqués par le partage de ressource. La fiabilité et les caractéristiques de contrôle de flux de *Myrinet* permettent une modélisation descriptive. Dans le cas du réseau *Quadrics*, un modèle basé également sur une approche descriptive est envisageable. Il conviendrait d'étudier plus en détails le comportement du réseau *Quadrics* en ayant, par exemple, la possibilité de réaliser un plus grand nombre d'expériences sur des grappes expérimentales. Le réseau *Gigabit Ethernet* est un réseau, qui peut être utilisé dans des architectures non dédiées, ce qui l'oblige à mettre en œuvre des mécanismes complexes pour assurer la fiabilité des communications. Ces mé-

canismes étant difficiles à étudier, un modèle quantifiant leurs effets a été proposé pour ce réseau.

Pour intégrer l'aspect dynamique des communications entre tâches d'une application, un second modèle a été proposé. Ce modèle, commun à chaque architectures réseaux, permet d'appliquer le partage de la bande passante de manière temporelle en coordination avec les évolutions des besoins des applications. Plus précisément, il décrit une modélisation des effets sur les temps de communication. Ces effets résultent de la modification des conflits réseaux avec le commencement ou la terminaison d'une communication.

Les modèles proposés prédisent avec une erreur faible d'environ 10% les temps des communications concurrentes.

Afin d'employer ces modèles sur des applications scientifiques et de permettre ainsi une analyse fine de leurs performances réseaux, nous avons implanté ces modèles dans un simulateur. Ce simulateur est basé sur une approche par simulation à événements discrets. Une application est décomposée en événements de calcul et de communication. Les événements de communication sont injectés dans un simulateur qui, suivant les communications concurrentes créées, applique le modèle de partage de bande passante sous-jacent à l'architecture réseau modélisée. L'utilisation du simulateur ajoute un élément important : le choix de la sémantique des communications et l'implantation de cette sémantique. L'exécution de plusieurs applications simultanément peut-être simulée. Il représente environ 4200 lignes de code.

8.4 Perspectives

Nous exposons les champs d'investigation à explorer à partir de nos travaux.

8.4.1 Approfondir la modélisation réseau

Lors de la définition et de l'évaluation des modèles, il a été montré que certaines caractéristiques peuvent donner lieu à des améliorations concernant les prédictions. Le modèle descriptif pour le réseau *Myrinet* se fonde sur certaines hypothèses comme l'utilisation du minimum des coefficients d'émission des communications sortantes d'un même nœud ou l'utilisation de la moyenne de ces coefficients sur le nombre d'états possibles. Bien que de tels choix offrent de bonnes performances dans une majorité des cas, il existe certains cas particuliers pour lesquels ils provoquent des erreurs de prédictions de l'ordre de 25% (confère schéma MK3, chapitre 7, sous-section 7.4.2). Une étude plus approfondie de ces cas particuliers permettrait d'améliorer les valeurs prédites.

La modélisation du réseau *Quadrics* est une autre voie à approfondir. Nous avons introduit certaines particularités du partage de la bande passante du réseau *Quadrics* qu'il serait intéressant d'explorer plus en avant afin de définir un modèle pour ce réseau. Cette étude nécessitera la possibilité d'effectuer un grand nombre d'expériences sur le réseau *Quadrics*.

Finalement, l'évolution des réseaux converge vers la technologie *10 Gigabit Ethernet* telle que présentée en conclusion du premier chapitre de ce document. Quelles modifications impliquent cette nouvelle technologie sur le partage de la bande passante et les modèles proposés ? Quelles sont les modèles qui régissent les architectures *Myri-10G* ou *10 Gigabit Ethernet* ?

8.4.2 Caractérisation des réseaux haute performance

Un autre point mis en avant par les expérimentations tient à la diversité des effets du partage de la bande passante. Par exemple, *Myrinet* et *Gigabit Ethernet* obtiennent des pénalités totalement différentes suivant le schéma de communication et, par conséquent, selon les conflits réseaux entre communications. Une technique largement répandue pour l'évaluation des performances réseaux se base principalement sur deux paramètres : la latence et la bande passante. Ces valeurs ne sont pas suffisantes pour comparer les réseaux entre eux. En effet, *Myrinet* a une latence et une bande passante nettement meilleures que *Gigabit Ethernet*. Pourtant, il résulte de la comparaison avec *Gigabit Ethernet* que la répartition de la bande qu'il propose est très pénalisante sur les performances. Dans ce sens, *Gigabit Ethernet* est plus efficace que *Myrinet* ou *Quadrics*. Il a été introduit également la mesure de la bande passante de bissection[1], mais cette mesure, bien que plus représentative du partage de la bande passante, est loin d'être exhaustive.

Ainsi, il serait possible, en utilisant les expérimentations menées, de définir un *benchmark* capable de comparer équitablement l'efficacité des réseaux à partager leurs ressources. Un tel *benchmark* pourrait être basé sur l'évaluation de plusieurs schémas de communications différents tels que des arbres ou des graphes complets et la comparaison des temps obtenus avec des bornes théoriques (aucun partage et partage avec le nombre maximum de communications en concurrence). Ce *benchmark* a un intérêt prépondérant dans les architectures multiprocesseurs.

8.4.3 Dimensionnement

Le dimensionnement de grappes de calcul a pour objectif de déterminer une configuration matérielle et logicielle permettant d'obtenir et d'exploiter au mieux un ensemble d'applications. Le problème des constructeurs est de pouvoir dimensionner une configuration de grappe qui soit adaptée aux besoins des clients et à son ensemble applicatif (et qui soit compatible avec son budget). Les architectures de grappes peuvent être qualifiées de modulaires par le fait que l'architecture est construite par agrégation de composants, un module pouvant être un nœud de calcul, avec au niveau de ce nœud un ou plusieurs processeurs, un espace mémoire plus ou moins étendu. La difficulté est alors de proposer une solution qui ne soit ni sous-dimensionnée ni sur-dimensionnée par rapport aux besoins des organisations et de leurs différentes applications.

Définir une méthodologie pour dimensionner une architecture est un exercice difficile. En effet, le dimensionnement consiste à prédire les performances des applications en ayant une connaissance abstraite de l'architecture (type de réseau, nombre de nœuds, nombre de processeurs par nœuds, etc). Dans les travaux de cette thèse, nous avons montré qu'il est possible de répondre à la problématique du choix du type de réseau pour une grappe. Un autre élément d'une grappe qu'il serait intéressant de dimensionner est la taille des nœuds en nombre de processeurs. En utilisant le simulateur, nous nous sommes posés la question de savoir s'il n'était pas également possible de dimensionner cet élément ? Pour tenter de répondre à cette question, nous avons cherché à prédire les performances d'une application en regroupant ces tâches sur des nœuds contenant différents nombres de processeurs. Pour ce faire, nous avons tracé une exécution de *HPL* en plaçant une tâche par machine. En utilisant cette trace, nous avons simulé le comportement de *HPL* sur des

[1] La bande passante de bissection correspond à la capacité bidirectionnelle du réseau entre deux partitions égales de nœuds.

nœuds ayant 2 puis 4 processeurs. Les performances prédites sont décevantes par rapport aux performances réelles mesurées. Cette perte de précision provient de la différence des durées des événements de calcul et, par conséquent, de la congestion mémoire provoquée par le regroupement des tâches sur les nœuds. De plus, la propagation de l'erreur augmente significativement l'erreur des prédictions. Ces expériences montrent que pour dimensionner le nombre de processeurs, des modèles réseaux et des modèles mémoire précis sont nécessaires.

Pour traiter le problème du dimensionnement, la société *Bull* et le laboratoire *LIG* (*ID-IMAG*) poursuivent leur partenariat en proposant un nouveau sujet de thèse qui s'inscrit dans la continuité de nos travaux recherche.

8.4.4 Propagation de l'erreur dans les simulations

La décomposition événementielle est une technique prometteuse pour étudier et analyser la corrélation entre les besoins des applications et les ressources disponibles des grappes. Elle permet d'injecter dans un simulateur, qui modélise le partage de ressources, les besoins des applications au cours du temps d'exécution. Toutefois, cette décomposition temporelle introduit le problème du contrôle de l'erreur et de sa propagation entre les événements [10]. En effet, il serait intéressant d'évaluer l'impact du cumul des erreurs par événements sur les prédictions finales des performances. Cette étude pourrait à la fois reposer sur des approches théoriques, en proposant par exemple des bornes ou des valeurs moyennes, et sur des constatations pratiques en utilisant le simulateur proposé.

8.4.5 Evaluation et prédiction

D'un point de vue plus industriel, il serait concevable d'intégrer les modèles prédictifs dans des logiciels d'évaluation de performances [39]. De tels outils permettraient d'évaluer, de comparer, de visualiser et d'indiquer les pertes de performances dues à certaines caractéristiques telles que les conflits générés ou le placement des tâches. Ces outils ajouteraient des fonctionnalités aidant significativement à la fois les constructeurs dans la proposition de grappes que leurs acquéreurs en identifiant leurs besoins en ressources. En éliminant l'influence de la ressource mémoire sur les nœuds multiprocesseurs, et en plaçant une tâche par nœud, ces outils permettraient également d'évaluer l'impact des sémantiques de communication sur les performances des applications.

Sur un plan académique, l'intégration de ces modèles dans un simulateur reconnu, tel que le simulateur de grille SimGrid[6], offrira de nouvelles perspectives à leur utilisation.

IV

Annexes

Configurations des grappes A

Nous présentons, dans le tableau A.1, les différentes caractéristiques matérielles des grappes utilisées lors des expériences menées pendant cette thèse. La plus grande partie des grappes appartiennent à la grille grid'5000. Les versions du noyau Linux sont supérieures au noyau 2.6.13.

Grappe	Commutateur	Carte réseau	Bus PCI	Processeurs
Helios	Foundry EdgeIron 48G (48 ports) / Myrinet 2000 (128 ports)	BCM5704 / Myrinet 2000	PCI-X AMD-8131	AMD Opteron 275 bicœurs 2.2 Ghz
Sophia	Cisco 3750 (24 ports) / Myrinet 2000 (128 ports)	BCM5704 / Myrinet 2000	PCI-X AMD-8131	AMD Opteron 246 1.9 Ghz
Lille	3Com SuperStack3 3870 (48 ports)	BCM5704	PCI-X AMD-8131	AMD Opteron 248 2 Ghz
Nancy	HP_ProCurve 34000cl 48G (48 ports)	BCM5721	PCI-X AMD-8132	AMD Opteron 246 2 Ghz
Rennes	Cisco 6509 (336 ports)	BCM5721	PCI AMD-7459	AMD Opteron 246 2 Ghz
Teratec	Quadrics QS2-64U64D (48 ports)	Quadrics Elan4	PCI Intel 82801	FAME-Bull 8016, Intel Itanium 2 1.6 Ghz

TAB. A.1 – Configuration matérielle des grappes de calcul.

Expérimentations introductives et élaborées : cas *Gigabit Ethernet* avec Lam/MPI B

Dans une première partie, cette annexe présente les expériences introductives pour l'implantation Lam/MPI. Dans une seconde partie ces expériences sont complexifiées.

B.1 Expérimentations introductives

Nous rappelons que ces expériences introductives concernent trois conflits élémentaires : le conflit E/E, le conflit S/S et le conflit E/S.

Conflit E/E (Gigabit Ethernet, Lam/MPI)

Le comportement général de ce conflit est proche de ceux observés dans le chapitre 4. Un même effet de saut se produit impliquant une pénalité pour des communications de taille supérieures à la valeur seuil du saut. Cette valeur se produit aussi lorsque MPI utilise le protocole avec rendez-vous. Pour Lam/MPI, ce changement se produit pour 128Ko, valeur seuil du saut. Au-delà de ce seuil, les temps des communications doublent. Notons une faible augmentation des temps de communications pour de très petites tailles de messages (< 1 Ko).

Conflit S/S (Gigabit Ethernet, Lam/MPI)

Nous remarquons dans ce conflit une particularité. En effet, une pénalité s'applique pour toutes les tailles de messages. Cette pénalité vaut approximativement 2 pour des valeurs inférieures à 128Ko, et 2 pour des valeurs supérieures.

Conflit E/S (Gigabit Ethernet, Lam/MPI)

Ce dernier conflit confirme bien qu'il s'agit d'un conflit peu pénalisant. En effet, les deux communications en concurrence ne sont que très peu ralenties quelque soit la taille des messages.

Finalement, le tableau B.1 résume les courbes précédantes en proposant certaines valeurs observées.

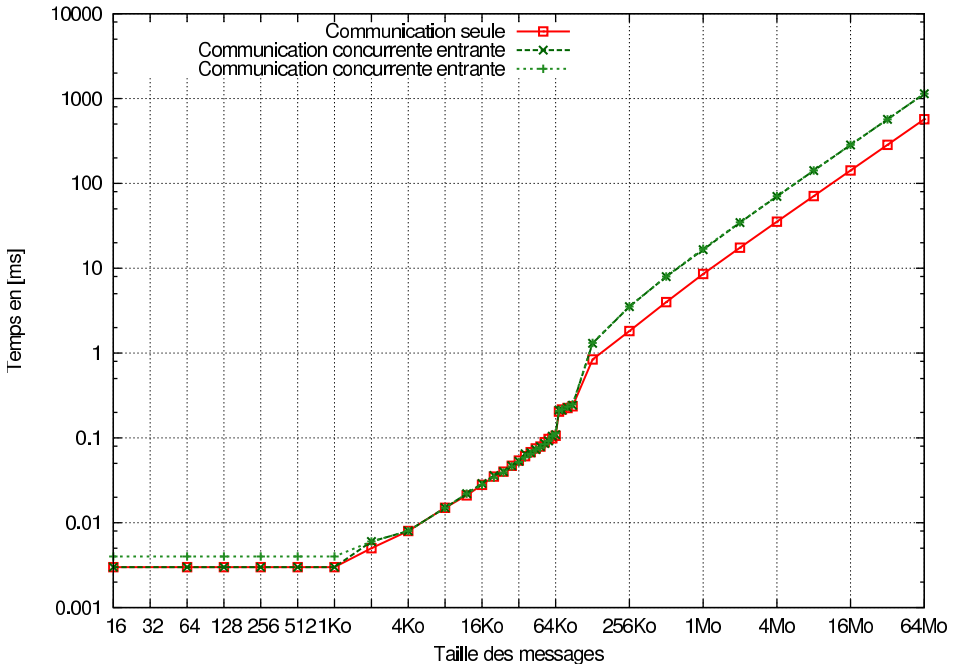


FIG. B.1 – *Gigabit Ethernet*, Lam/MPI, Conflit E/E

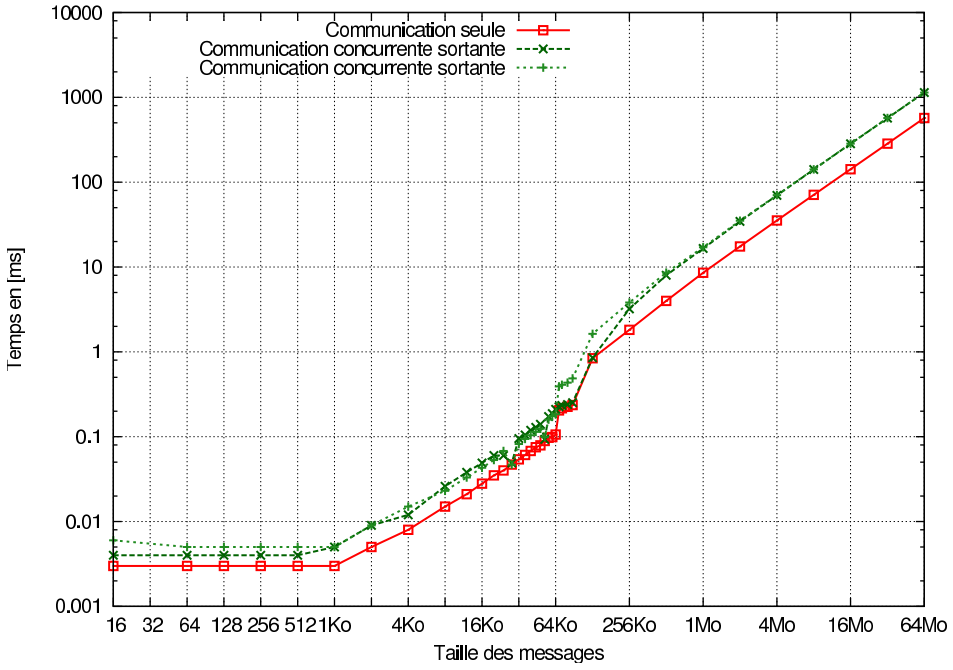


FIG. B.2 – *Gigabit Ethernet*, Lam/MPI, Conflit S/S

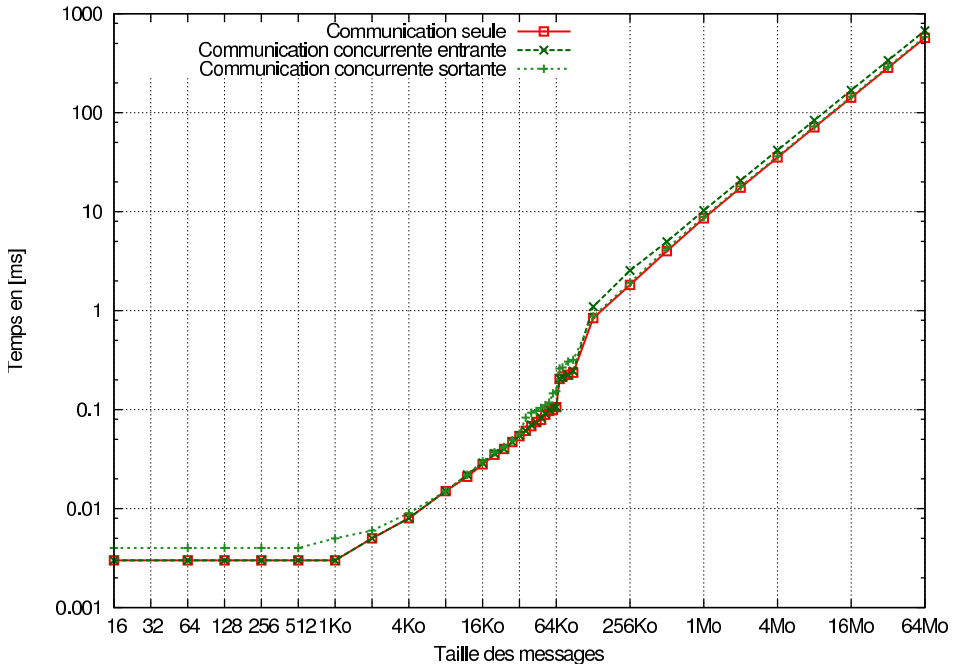


FIG. B.3 – *Gigabit Ethernet*, Lam/MPI, Conflit E/S

		16o [μs]					16Ko [ms]					16Mo[s]				
		Moy	Med	ET	Min	Max	Moy	Med	ET	Min	Max	Moy	Med	ET	Min	Max
Sans conflit		2.7	2.7	0.1	2.4	6.4	0.023	0.022	0.001	0.021	0.049	0.142	0.142	0	0.142	0.146
Conflit E/E	Com1	2.7	2.7	0.1	2.5	4.8	0.022	0.022	0	0.021	0.031	0.283	0.283	0	0.278	0.286
	Com2	3.3	3.3	0.9	2.9	3.5	0.023	0.023	0	0.022	0.032	0.283	0.283	0	0.279	0.286
Conflit S/S	Com1	4.0	3.9	0.5	3.3	8.9	0.049	0.049	0.002	0.044	0.061	0.284	0.284	0	0.279	0.285
	Com2	5.7	5.6	0.3	4.9	8.7	0.043	0.043	0.003	0.037	0.058	0.284	0.284	0	0.276	0.285
Conflit E/S	Com1	2.9	2.9	0.4	2.6	8.4	0.023	0.022	0.001	0.021	0.039	0.150	0.150	0.001	0.149	0.154
	Com2	3.7	3.8	0.4	2.9	8.0	0.034	0.034	0.001	0.028	0.050	0.146	0.146	0	0.146	0.153

TAB. B.1 – Indices statistiques des temps de communications des conflits E/E, S/S, E/S pour le réseau *Gigabit Ethernet* et Lam/MPI.

B.2 Expérimentations élaborées :

Les expériences élaborées montrent que de manière similaire à *Mpich*, les pénalités observées suivent une hiérarchie. En particulier, le schéma 9e montre que les communications sortantes des nœuds 0 et 2 ne sont pas influencées par le conflit du nœud 1. Les conflits du schéma 7e montrent une compensation entre les pénalités des deux communications soumises à un seul conflit et celles soumises à deux conflits. D'une manière générale, les pénalités observées sont proches de celles des conflits élémentaires.

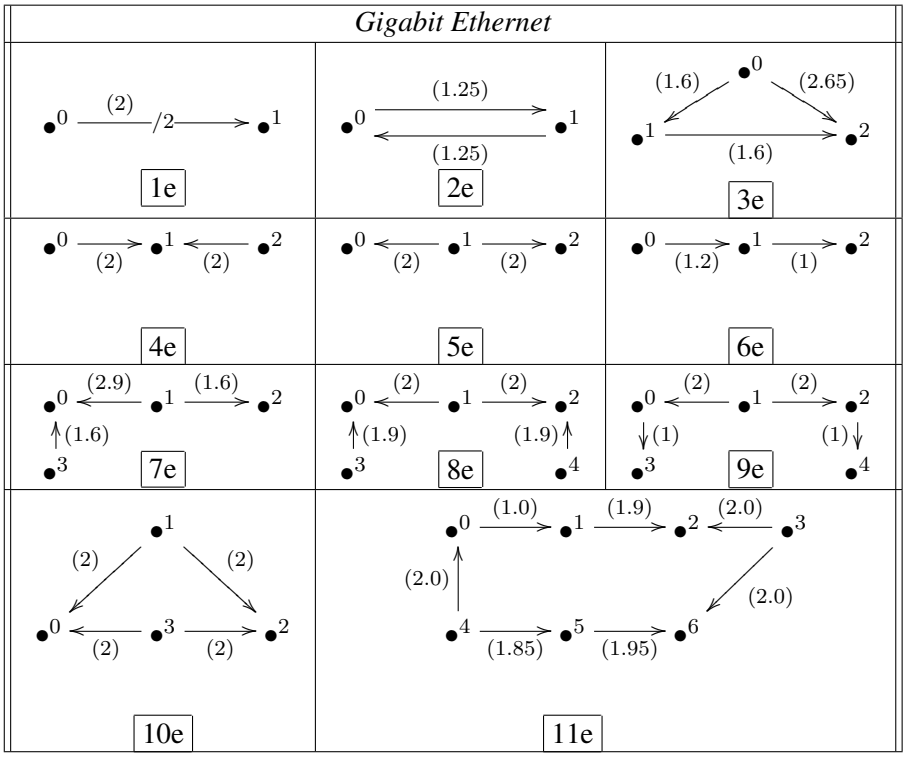


FIG. B.4 – Conflits complexes, *Gigabit Ethernet* et Lam/MPI.

Bibliographie

[1] A. Alexandrov, M. Ionescu, K. Schauser and C. Scheiman. LogGP : Incorporating Long Messages into the LogP model for Parallel Computation. *Journal of Parallel and Distributed Computing*, 44(1) :71–79, 1997.

[2] A. D. George and R. A. Vanloon. High-fidelity modelling and simulation of myrinet system area networks. *International journal of modelling and simulation*, 21(1) :40–50, 2001.

[3] A. Dusseau, D. E. Culler, K. E. Schauser, and R. P. Martin. Fast Parallel Sorting Under LogP : Experience with the CM-5. *IEEE Transactions on Parallel and Distributed Systems*, 7(8) :791–805, 1996.

[4] A. Erramilli, M. Roughan, D. Veitch and W. Willinger. Self-Similar Traffic and Network Dynamics. *Proceedings of the IEEE*, 90(5) :800–819, 2002.

[5] A. Harwood. *High Performance Interconnection Networks*. PhD thesis, Computer Science, Griffith University, July 2002. 160 pages.

[6] A. Legrand, L. Marchal and H. Casanova. Scheduling Distributed Applications : the SimGrid Simulation Framework. In *CCGRID '03 : Proceedings of the 3st International Symposium on Cluster Computing and the Grid*, page 138. IEEE Computer Society, 2003.

[7] A. Wakatani and M. Wolfe. Effectiveness of Message Strip-Mining for Regular and Irregular Communication. In *7th International Conference on Parallel and Distributed Computing Systems - PDCS'94 , Las Vegas,* , October 1994.

[8] B. De Oliveira Stein. *Visualisation interactive et extensible de programmes parallèles à base de processus légers*. PhD thesis, Université Joseph-Fourier Grenoble I, October 1999. 148 pages.

[9] B. Goglin. *Réseaux rapides et stockage distribué dans les grappes de calculateurs : propositions pour une interaction efficace*. PhD thesis, École normale supérieure de Lyon, October 2005. 194 pages.

[10] B. P. Zeigler, T. G. Kim and H. Praehofer. *Theory of Modeling and Simulation*. Academic Press, Inc., Orlando, FL, USA, 2000.

[11] B. Ramakrishna Rau and J. A. Fisher. Instruction-level parallel processing : history, overview, and perspective. *The Journal of Supercomputing*, 7(1-2) :9–50, 1993.

[12] B. Tourancheau. High Speed Networks for Clusters, the BIP-Myrinet Experience. *Lecture Notes in Computer Science*, 1908 :9, 2000.

[13] C. A. Moritz, and M. I. Frank. LoGPC : Modeling Network Contention in Message-Passing Programs. *IEEE Transactions on Parallel and Distributed Systems*, 12(4) :404–415, 2001.

[14] C. Dubnicki, A. Bilas and K. Li. Design and Implementation of Virtual Memory-Mapped Communication on Myrinet. In *IPPS '97 : Proceedings of the 11th International Symposium on Parallel Processing*, page 388. IEEE Computer Society, 1997.

[15] C. E. Leiserson. Fat-trees : universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, 34(10) :892–901, 1985.

[16] C. E. Spurgeon. *Ethernet : The Definitive Guide*. O'Reilly and Associates, 2000.

[17] C. Iancu, P. Husbands and W. Chen. Message Strip-Mining Heuristics for High Speed Networks. In *VECPAR'04, 6th International Meeting. High Performance Computing for Computational Science , Valencia, Spain*, pages 424–437, June 2004.

[18] C-Y. Chou, H-Y. Chang, S-T. Wang and C-H. Wu. A Semi-Empirical Model for Maximal LINPACK Performance Predictions. *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, 0 :343–348, 2006.

[19] C.A.R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, 21(8) :666–677, aug 1978.

[20] Cruz. A Calculus for Network Delay, Part I : Network Elements in Isolation, Part II : Network Analysis. *IEEETIT : IEEE Transactions on Information Theory*, 37, 1991.

[21] D. B. Gustavson. The Scalable Coherent Interface and Related Standards Projects. *IEEE Micro*, 12(1) :10–22, 1992.

[22] D. Culler, R. Karp, D. Patterson, A. Sahay, E. Santos, K. Schauser, R. Subramonian and T. von Eicken. LogP : a practical model of parallel computation. *Commun. ACM*, 39(11) :78–85, 1996.

[23] D. J. Lilja. Cache Coherence in Large-Scale Shared-Memory Multiprocessors : Issues and Comparisons. *ACM Computing Surveys*, 25(3) :303–338, 1993.

[24] D. M. Tullsen, S. J. Eggers and H. M. Levy. Simultaneous multithreading : maximizing on-chip parallelism. In *ISCA '98 : 25 years of the international symposia on Computer architecture (selected papers)*, pages 533–544. ACM Press, 1998.

[25] D. Sundaram-Stukel and M. K. Vernon. Predictive analysis of a wavefront application using LogGP. *ACM SIGPLAN Notices*, 34(8) :141–150, 1999.

[26] D. Turner and X. Chen. Protocol-Dependent Message-Passing Performance on Linux Clusters. In *CLUSTER '02 : Proceedings of the IEEE International Conference on Cluster Computing*, page 187, 2002.

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

[28] E. Gabriel, J. Pjesivac-Grbovic, T. Angskun, G. Bosilca, G. E. Fagg and J. Dongarra. Performance Analysis of MPI Collective Operations. In *IPDPS '05 : Proceedings of the 19th*

IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 15, page 272. IEEE Computer Society, 2005.

[29] F. Petrini. *Communication Performance of Wormhole Interconnection Networks*. PhD thesis, Università degli Studi di Pisa, February 1997. 165 pages.

[30] F. Petrini, J. Beecroft, D. Addison and M. McLaren. QsNet-II : An Interconnect For Supercomputing Applications. In *Proceedings of Hot Chips '03*, August 2003.

[31] F. Petrini, W. chun Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The quadrics network : High performance clustering technology. *IEEE Micro*, 22(1) :46–57, January 2002.

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

[33] G. Rodríguez, R. M. Badia and J. Labarta. Generation of Simple Analytical Models for Message Passing Applications. In *Euro-Par 2004 Parallel Processing, 10th International Euro-Par Conference*, pages 183–188, 2004.

[34] H. Casanova and L. Marchal. A Network Model for Simulation of Grid Application. Research Report 4596, INRIA, October 2002.

[35] I. Stoica, F. Sultan and D. Keyes. A Hyperbolic Model for Communication in Layered Parallel Processing Environments. *Journal of Parallel and Distributed Computing*, 39(1) :29–45, 1996.

[36] InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.0, October 2000. `http://www.infinibandta.org`.

[37] Intel Corporation. *Intel Trace Analyzer 7.0 Reference Guide*. `http://cache-www.intel.com/cd/00/00/32/75/327516_327516.pdf`.

[38] J. Backus. Can programming be liberated from the von Neumann style ? : a functional style and its algebra of programs. *Communications of the ACM*, 21(8) :613–641, August 1978.

[39] J. Dongarra, S. Browne and K. London. Review of Performance Analysis Tools for MPI Parallel Programs. *National HPCC Software Exchange (NHSE) Review*, 3(1), 1998.

[40] J. Duato, P. López, J. M. Martínez and F. Petrini. On the Reduction of Deadlock Frequency by Limiting Message Injection in Wormhole Networks. In *PCRCW '97 : Proceedings of the Second International Workshop on Parallel Computer Routing and Communication*, volume 1417, pages 295–307. Springer-Verlag, 1997.

[41] J. Gibson, R. Kunz, D. Ofelt, M. Horowitz, J. Hennessy and M. Heinrich. FLASH vs. (Simulated) FLASH : closing the simulation loop. In *ASPLOS-IX : Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, volume 35, pages 49–58. ACM Press, 2000.

[42] J. Handy. *The cache memory book*. Academic Press Professional, Inc., San Diego, CA, USA, 1993.

[43] J. Liu, B. Chandrasekaran, J. Wu, W. Jiang, S. Kini, W. Yu, D. Buntinas, P. Wyckoff and D K. Panda. Performance Comparison of MPI Implementations over InfiniBand, Myrinet and Quadrics. In *SC '03 : Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 58, 2003.

[44] J. Padhye and V. Firoiu and D. Towsley and J. Krusoe. Modeling TCP Throughput : A Simple Model and its Empirical Validation. *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 303–314, 1998.

[45] J. Postel. User datagram protocol. Internet Requests for Comments (768), August 1980. RFC 768, `http://www.ietf.org/rfc/rfc768.txt`.

[46] J. Postel. Transmission Control Protocol. USC/Information Sciences Institute, September 1981. RFC 793, `http://www.ietf.org/rfc/rfc793.txt`.

[47] J. Turner and R. Melen. Multirate Clos Networks. IEEE Communications Magazine, 41, no. 10 : 38–44., 2003.

[48] J.-Y. Le Boudec. Application of network calculus to guaranteed service networks. *IEEE Transaction on Information theory*, 44(3), May 1998.

[49] J.-Y. Le Boudec and P. Thiran. *Network Calculus : A Theory of Deterministic Queuing Systems for the Internet*, volume 2050. Springer-Verlag, 2001.

[50] K. Berkbigler, G. Booker, B. Bush, K. Davis, and N. Moss. Simulating the Quadrics Interconnection Network. *HPC2003 : High Performance Computing Symposium 2003, Advance Simulation Technologies Conference 2003*, 2003.

[51] K. Park and W. Willinger. *Self-Similar Network Traffic and Performance Evaluation*. John Wiley & Sons, Inc., New York, NY, USA, 2000.

[52] L. A. Steffenel, M. Martinasso and D. Trystram. Total Exchange Performance Modelling Under Network Contention. In *Proceedings of the 2nd International Conference on Grid and Pervasive Computing (GPC'2007)*, LNCS, May 2007. To be published.

[53] L. F. Pollacia. A survey of discrete event simulation and state-of-the-art discrete event languages. *SIGSIM Simulation Digest*, 20(3) :8–25, 1989.

[54] L. Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, 1975.

[55] L. Massoulie and J. Roberts. Bandwidth Sharing : Objectives and Algorithms. *IEEE/ACM Transactions on Networking*, 10(3) :320–328, 2002.

[56] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control, April 1999. RFC 2581, `http://www.ietf.org/rfc/rfc2581.txt`.

[57] M. Martinasso. Étude et modélisation de la congestion réseau sur des architectures de grappes modulaires. In *16ème Rencontres Francophones du Parallélisme - Renpar'16*, pages 149–158, April 2005.

[58] M. Martinasso. Modéles de communications concurrentes sur des grappes SMP. In *Perpi'2006, Actes de la conférences Renpar'17*, pages 132–139, Canet en Roussillon, October 2006.

[59] M. Martinasso and J-F. Méhaut. Prediction of communication latency over complex network behaviors on SMP clusters. In *Proceedings of the 2nd European Performance Engineering Workshop, EPEW 2005*, volume 3670 of *Lecture Notes in Computer Science*, pages 172–186, Versailles, September 2005. Springer-Verlag.

[60] M. Martinasso and J.-F. Méhaut. Analysis and model of network contentionover SMP clusters. In *International Meeting on Grid and Parallel Computing*. American University of Beiruth, January 2006.

[61] M. Martinasso and J.-F. Méhaut. Model of conccurent MPI communications over SMP clusters. Technical Report 00071352, HAL-INRIA, May 2006.

[62] Message Passing Interface Forum. MPI : A message-passing interface standard. *International Journal of Supercomputer Applications*, pages 165–414, 1994.

[63] Myricom. Myrinet Express (MX) : A High-Performance, Low-Level,Message-Passing Interface for Myrinet, July 2005. `http://www.myri.com/scs/MX/doc/mx.pdf`.

[64] Myricom Inc. GM : A Message-Passing System For Myrinet Networks, 2003. `http://www.myri.com/scs/GM-2/doc/html/`.

[65] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic and W.-K. Su. Myrinet : A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1) :29–36, 1995.

[66] N. T. Spring, R. Wolski and J. Hayes. The network weather service : a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6) :757–768, 1999.

[67] O. Aumage, L. Bougé and R. Namyst. A Portable and Adaptative Multi-protocol Communication Library for Multithreaded Runtime Systems. *Lecture Notes in Computer Science*, 1800 :1136+, 2000.

[68] O. Maquelin, G. R. Gao, H. H. J. Hum, K. B. Theobald and X.-M. Tian. Polling watchdog : combining polling and interrupts for efficient message handling. In *ISCA '96 : Proceedings of the 23rd annual international symposium on Computer architecture*, pages 179–188. ACM Press, 1996.

[69] P. Erdös and A. Rényi. On the Evolution of Random Graphs. *Math. Inst. Hungar. Acad. Sci.*, 5 :17–60, 1960.

[70] P. Erdös and A. Rényi. On the strength of connectedness of random graphs. *Acta. Math. Sci. Hung*, 12 :216–267, 1961.

[71] P. H. Worley. Impact of Communication Protocol on Performance. In *Proceedings of the Second International Workshop on Software Engineering and Code Design in Parallel Meteorological and Oceanographic Applications*, pages 277–288. NASA Conference Publication 1998-206860, 1998.

[72] P. Kermani and L. Kleinrock. Virtual Cut Through : A New Computer Communication Switching Technique. *Computer Networks*, 3 :267–286, 1979.

[73] P. López, E. Baydal and J. Duato. A Family of Mechanisms for Congestion Control in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 16(9) :772–784, 2005.

[74] P. Wang, G. Turner, D. A. Lauer, M. Allen, S. Simms, D.Hart, M. Papakhian and C. A. Stewart. LINPACK Performance on a Geographically Distributed Linux Cluster. In *IPDPS'04 : Proceedings of the 18th International Parallel and Distributed Processing Symposium*, volume 15, page 245, 2004.

[75] Q. Snell, A. Mikler and J. Gustafson. Netpipe : A Network Protocol Independent Performace Evaluator. In *IASTED International Conference on Intelligent Information Management and Systems*, June 1996.

[76] R. Martin A. Vahdat , D. Culler and T. Anderson. Effects of communication latency, overhead, and bandwidth in a cluster architecture. *Proceedings of the 24th annual international symposium on Computer architecture*, pages 85–97, 1997.

[77] R. Morris. TCP behavior with many flows. In *ICNP '97 : Proceedings of the 1997 International Conference on Network Protocols (ICNP '97)*, volume 00, page 205. IEEE Computer Society, 1997.

[78] R. W. Hockney. The Communication Challenge for MPP : Intel Paragon and Meiko CS-2. In *Parallel Computing, North-Holland*, volume 20, pages 389–398, 1994.

[79] S.-A. Reinemo, B. Dobinson, S. Haas, O. Lysne, B. Martin and T. Skeie. Topologies and Routing in Gigabit Switching Fabrics. In *Proceedings of 2nd International Conference on Communications in Computing*, pages 142–149, June 2001.

[80] S. C. Kim and S. Lee. Measurement and Prediction of Communication Delays in Myrinet Networks. *Journal of Parallel and Distributed Computing*, 61(11) :1692–1704, 2001.

[81] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh and A. Gupta. The SPLASH-2 programs : characterization and methodological considerations. In *ISCA '95 : Proceedings of the 22nd annual international symposium on Computer architecture*, pages 24–36, 1995.

[82] S. H. Rodrigues, T. E. Anderson and D. E. Culler. High-Performance Local-Area Communication With Fast Sockets. In *USENIX Annual Technical Conference*, pages 257–274, January 1997.

[83] S. Owicki and A. Agarwal. Evaluating the performance of software cache coherence. In *ASPLOS-III : Proceedings of the third international conference on Architectural support for programming languages and operating systems*, pages 230–242. ACM Press, 1989.

[84] S. Pakin, M. Lauria and A. Chien. High Performance Messaging on Workstations : Illinois Fast Messages (FM) for Myrinet. In *Supercomputing '95 : Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, page 55, 1995.

[85] S. S. Shende and A. D. Malony. The Tau Parallel Performance System. *International Journal of High Performance Computing Applications*, 20(2) :287–311, 2006.

[86] S. Srbljic, Z. G. Vranesic, M. Stumm and L. Budin. Analytical Prediction of Performance for Cache Coherence Protocols. *IEEE Transactions on Computers*, 46(11) :1155–1173, 1997.

[87] T. Bonald and L. Massoulie. Impact of fairness on Internet performance. In *SIGMETRICS/Performance*, pages 82–91, 2001.

[88] T. Kielmann and H. E. Bal and K. Verstoep. Fast Measurement of LogP Parameters for Message Passing Platforms. In *IPDPS '00 : Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing*, pages 1176–1183, 2000.

[89] T. Leng, R. Ali, J. Hsieh, V. Mashayekhi and R. Rooholamini. Performance Impact of Process Mapping on Small-Scale SMP Clusters - A Case Study Using High Performance

Linpack. In *IPDPS'02 : Proceedings of the 16th International Parallel and Distributed Processing Symposium*, page 263, 2002.

[90] T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake and C. V. Packer. BEOWULF : A Parallel Workstation for Scientific Computation. In *Proceedings of the 24th International Conference on Parallel Processing*, pages 11–14, 1995.

[91] T. von Eicken, D. E. Culler, S. C. Goldstein and K. E. Schauser. Active messages : a mechanism for integrated communication and computation. In *ISCA '92 : Proceedings of the 19th annual international symposium on Computer architecture*, pages 256–266. ACM Press, 1992.

[92] The Network Simulator project. The Network Simulator ns-2. `http://nsnam.isi.edu/nsnam/index.php/Main_Page`.

[93] V. Jacobson. Congestion avoidance and control. In *SIGCOMM '88 : Symposium proceedings on Communications architectures and protocols*, pages 314–329. ACM Press, 1988.

[94] W. Amme, P. Braun, W. Lowe and E. Zehendner. LogP modelling of list algorithms. In *In Proceedings 11th Symposium on Computer Architecture and High Performance Computing*, pages 157–63, 1999.

[95] W. Feng and P. Tinnakornsrisuphap. The failure of TCP in high-performance computational grids. In *Supercomputing '00 : Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*, page 37. IEEE Computer Society, 2000.

[96] W. Gropp. MPICH2 : A New Start for MPI Implementations. In *Proceedings of the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, page 7. Springer-Verlag, 2002.

[97] W. Gropp and E. Lusk. User's Guide for MPE : Extensions for MPI Programs. `http://www.mcs.anl.gov/mpi/mpich1/docs/mpeman.pdf`.

[98] W. Gropp, E. Lusk, N. Doss and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6) :789–828, September 1996.

[99] W. J. Dally. Virtual-Channel flow control. *IEEE Transactions on Parallel Distributed Systems*, 3(2) :194–205, 1992.

[100] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5) :547–553, 1987.

[101] W. Loewe, J. Eisenbiegler and A. Wehrenpfennig. On the Optimization by Redundancy Using an Extended LogP Model. *Advances in Parallel and Distributed Computing Conference (APDC '97)*, 00 :149, 1997.

[102] W. Willinger, V. Paxson and M. S. Taqqu. Self-similarity and Heavy Tails : Structural Modeling of Network Traffic. *A Practical Guide to Heavy Tails : Statistical Techniques and Applications*, 1998.

Résumé

Analyse et modélisation des communications concurrentes
dans les réseaux haute performance.

La croissance des capacités de calcul des processeurs se poursuit, non plus par l'augmentation des fréquences d'horloge, mais par la multiplication d'unités de traitement (cœur) au sein des chips. Cette augmentation du nombre de cœurs par processeur induit un partage des autres composants de la machine entre les différentes requêtes. En outre, les applications parallèles provenant du calcul scientifique s'efforcent d'exploiter au maximum les ressources d'une architecture parallèle. La compréhension du partage de ressource devient un des défis majeurs pour utiliser efficacement ces architectures.

Ces nouveaux comportements de partage de ressources, ainsi produits, sont difficiles à interpréter et à prédire. Dans cette thèse, nous avons étudié le problème du partage du réseau. Les grappes de calcul utilisent des réseaux dédiés tels que *Gigabit Ethernet*, *Myrinet* ou *Quadrics*. L'exécution simultanée des tâches d'une application entraîne des accès concurrents sur la ressource réseau. Leurs effets conduisent à une perte de performance qui découle du partage de la bande passante réseau entre communications.

Suivant ce contexte, nous présentons une analyse fine des comportements concurrents sur les architectures : *Quadrics*, *Myrinet* et *Gigabit Ethernet*. Cette analyse conduit à la définition de modèles prédictifs basés sur la notion de partage de la bande passante. En outre, nous montons que l'intégration de ces modèles dans une simulation permet de prédire les impacts dûs à la concurrence entre communications *MPI* résultantes de l'exécution d'applications scientifiques. La prédiction des comportements concurrents donne lieu, au niveau scientifique et technique, à une meilleure connaissance des besoins des applications et, au niveau industriel, à la proposition de solutions de grappes adaptées aux besoins de leurs clientèles.

Mots-clés : Communications concurrentes, congestion réseau, *MPI*, réseaux haute performance, modélisation et simulation, évaluation de performance.

Abstract

Analysis and modelling of concurrent communications
over high performance networks.

Improving processor capabilities by increasing the number of cores produces sharing effects over computer components among core requests. Furthermore, parallel applications from scientific fields attempt to fully exploit the resources of parallel architectures. Therefore, understanding resource-sharing phenomenon becomes one major issue for actual parallel architectures.

These new behaviours of resource sharing are hard to analyse and to predict. In this work, we study the problem of network sharing. Clusters use dedicated high performance networks such as *Gigabit Ethernet*, *Myrinet* or *Quadrics*. Simultaneous executions of application tasks create concurrent access over the network. Their effects lead to performance leaks because of bandwidth network segmentation among communications.

In this context, we present a deep analysis of concurrent behaviours for *Quadrics*, *Myrinet* and *Gigabit Ethernet*. This analysis aims to the definition of predictive models based on the concept of bandwidth sharing. In addition, we integrate such models in a simulation process for predicting concurrent communication impacts resulting of scientific application runs. Predicting concurrent behaviour leads, in scientific and technologic areas, to a better knowledge of application needs and, in industrial context, to advance cluster solutions adjusted to customer requirements.

Keywords : Concurrent communications, network contention, *MPI*, high performance networks, modelling and simulation, performance evaluation.