Ordonnancement et qualité de service pour réseaux rapides

Jérôme Clet-Ortega

Mémoire de Master Recherche

présenté par

Jérôme Clet-Ortega

Ordonnancement et qualité de service pour réseaux rapides

Encadrement : Olivier Aumage et Guillaume Mercier

Février – Juin 2007

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

Table des matières

Chapitre 1

Introduction

1.1 Le calcul intensif

En 1955, des scientifiques issus du laboratoire national de Los Alamos utilisent l'informatique pour simuler un phénomène physique qu'ils ne peuvent observer [8]. La simulation numérique voit le jour. Celle-ci s'appuie sur des modèles théoriques, comme les mécanismes de conduction de la chaleur, pour en donner une représentation réaliste. Beaucoup de domaines sont concernés. En climatologie par exemple, l'atmosphère est découpée en un grand nombre de boîtes élémentaires où sont effectués des calculs paramétrés par les observations réalisées, la taille de ces boîtes déterminant la précision des prévisions.

À l'époque de la première simulation, l'ordinateur utilisé (le « Maniac ») pouvait effectuer quelques dix mille opérations par seconde, ce qui peut sembler dérisoire au regard des centaines de millions d'opérations par seconde qu'effectue n'importe quel ordinateur de bureau actuel. Cependant les applications à exécuter évoluent et se complexifient sans cesse au cours du temps. Certaines nécessitent de contenir le temps d'exécution sous un seuil acceptable, d'autres requièrent plus de précision et de fiabilité dans les résultats. Afin de répondre aux besoins croissants de ces applications, il est nécessaire de disposer de davantage de puissance de calcul. Mais l'apport de cette puissance ne peut pas se faire en s'appuyant uniquement sur l'évolution en fréquence des processeurs. Les besoins sont immédiats et il n'est pas possible d'attendre la prochaine génération de processeurs. Le recours au parallélisme est donc inévitable.

Différentes solutions ont été proposées. Les premières ont été les supercalculateurs, des machines qui, à l'époque, offraient les meilleures performances pour les applications nécessitant de fréquentes communications. Les architectures multiprocesseurs à mémoire commune en font partie par exemple. Initialement dominants dans le monde du calcul parallèle, ces supercalculateurs ont, en raison de leur coût et de leur faible évolutivité, peu à peu laissé entrevoir une alternative : les grappes de calculs. Ces ordinateurs (nœuds) reliés entre eux par des réseaux rapides exhibent en effet un rapport performances/prix permettant de les concurrencer et de s'imposer sur le marché.

Les performances des grappes de calcul sont conditionnées par les performances des réseaux d'interconnexions sous-jacents (Myrinet [14], Infiniband [2], etc.) En effet, ceux-ci affichent des latences de l'ordre de la micro-seconde et un débit de plusieurs gigabits par seconde. De nos jours, l'hétérogénéité des grappes de calcul, que ce soit au niveau des multiples architectures des machines ou au niveau des différentes technologies de réseaux employées,

est un réel problème pour le programmeur de l'application. Aucune des technologies ne s'est réellement démarquée des autres, chacune ayant des caractéristiques bien spécifiques avec ses avantages et ses inconvénients. De plus, pour chacune d'entre elles, l'interface de pilotage fournie y est intrinsèquement liée. Le point essentiel est alors de fournir au programmeur un support logiciel offrant un modèle de communication plus générique pour interagir avec le matériel. Les environnements de programmation, utilisés par les applications en question, s'appuient notamment sur des supports exécutifs offrant une interface homogène. Ces supports ou bibliothèques de communication, doivent assurer la portabilité tout en minimisant le surcoût logiciel, de manière à conserver les performances des réseaux utilisés.

1.2 Les applications et leurs besoins

Malgré les efforts soutenus par les bibliothèques de communication pour offrir des interfaces génériques, il n'est pas possible d'abstraire parfaitement la ressource. En effet, chacune des technologies réseau offre une interface différente, les services fournis par le matériel étant spécifiques. Dans la plupart des cas, une autre interface est fournie, d'un plus haut niveau d'abstraction et offrant des primitives de communication plus standards (par exemple des implémentations du standard MPI citePPP-MPI [13], mais moins efficaces. En pratique, les développeurs de bibliothèques de communication d'un niveau intermédiaire (telles que MADELEINE [3] [23], ou encore Fast Message [17]), font toujours des choix sur les fonctionnalités de ces réseaux pour essayer d'obtenir le meilleur compromis performance/portabilité.

Or, chaque flux de données issu d'une application a des contraintes spécifiques selon la taille de ces données, la nécessité ou non de les transmettre au plus vite, etc. Et les choix réalisés par les bibliothèques de communication ne conviennent pas toujours à ces contraintes. Une réponse judicieuse serait de permettre à l'utilisateur de paramétrer les procédés de transmission selon l'application qui va utiliser la bibliothèque de communication. Les solutions proposées permettent de le faire de façon statique : par exemple, demander à utiliser le processeur pour effectuer des calculs pendant les transferts (recouvrement), ou au contraire attendre l'arrivée des messages avant de continuer le calcul. Mais ce paramétrage est décidé pour toute l'exécution, ce qui peut être un problème si au cours du temps l'application exprime de nouveaux besoins (par exemple plus de réactivité au détriment du recouvrement, etc.)

Actuellement, les bibliothèques, telle que NEWMADELEINE [5], proposent un certain niveau de paramétrage répondant à ce type de besoins. Cependant, les optimisations appliquées ont une portée globale aux flux de données, quel que soit le nombre de communications utilisées. Et l'évolution dans le domaine du calcul intensif, que ce soit au niveau du matériel ou des logiciels, est telle que la multitude de flots excède les possibilités de multiplexage physique. Les applications produisant des flux aux besoins différents sont de plus en plus courantes, autant que les cas d'utilisation conjointe de plusieurs applications, chacune produisant ses propres flux. Citons entre autres l'action de collaboration académique financée par l'ANR : le projet LEGO [1]. Ce projet vise une mise en commun d'outils et d'environnements pour le calcul distribué à grande échelle. Chacun de ces logiciels est axé sur une problématique particulière : les appels de procédure à distance, les mémoires distribuées virtuellement partagées, le contrôle interactif, etc. Autant de paradigmes reposant en grande partie sur les communications entraînent une multiplication du nombre de flux. Le projet EPSN [7] en est une deuxième illustration : la visualisation des résultats intermédiaires y est effectuée au cours de

résolutions numériques d'un schéma dans le but d'interagir avec celui-ci. Qui dit interactions, dit communications, auxquelles on rajoute les communications propres à chaque algorithme (visualisation et calcul).

La transmission de messages de visualisation ou de contrôle ne doit en aucune façon diminuer le temps d'exécution des calculs à surveiller. En outre, le temps d'interruption d'un calcul, en raison de l'attente d'un message nécessaire pour la suite, doit être minime. La scrutation de sa réception requiert donc d'être plus fréquente que celles de données moins urgentes.

Pour répondre aux besoins de ces applications, il faudrait pouvoir cloisonner les choix concernant le schéma de communication, ceci de manière ponctuelle, en fonction des flots de données.

1.3 Vers un traitement adapté à chaque besoin

Les architectures de communication actuelles traitent les flux de données de manière globale. Elle ne permettent pas de les différencier. La solution serait alors d'offrir à chacun des flux des garanties idoines dont les paramètres seraient fournies par l'interface. Notre étude portera donc sur les différentes conditions d'expression des contraintes applicatives caractéristiques aux réseaux rapides (latence, réactivité, recouvrement, etc.) et sur l'intégration du concept de qualité de service au sein d'une bibliothèque de communication.

En vue d'accomplir notre objectif, nous réalisons nos travaux dans le cadre de la bibliothèque de communication NewMadeleine. Son architecture est basée sur un moteur d'optimisation qui assure une progression des communications décorellée de l'application. Le but d'un tel moteur est d'offrir les meilleures performances pour chaque réseau, de manière intransigeante.

En résumé, nous proposons de mettre en œuvre la notion de qualité de service dans un contexte inédit : les réseaux rapides, et ainsi prendre en compte les spécificités de chaque flux de données les traversant.

Dans le premier chapitre de ce mémoire, nous énonçons les conséquences des évolutions dans le monde du calcul intensif. Le chapitre présente alors les différentes solutions proposées par les interfaces de communication actuelles, et notamment sur l'interface NewMadeleine. Ce chapitre pose la problématique et amène un concept vers lequel s'orienter : la qualité de service. Dans le chapitre suivant, nous exposons les contraintes propres aux réseaux des grappes de calculs, puis l'architecture et l'organisation de notre proposition sont décrites. Nous détaillons ensuite quelques aspects importants de l'implémentation de notre proposition au sein de la bibliothèque de communication NewMadeleine. Le chapitre 5 affiche alors les résultats qui prouvent l'efficacité de notre solution lorsque de multiples flux aux besoins différents cohabitent. Enfin, nous concluons sur les perspectives soulevées par ces travaux dans le dernier chapitre.

Chapitre 2

Équité et favoritisme dans les réseaux rapides

Le monde du calcul intensif est en perpétuel mouvement, les architectures physiques et logicielles changent. De plus en plus de flots de communication aux besoins différents s'entremêlent sur les réseaux rapides et les solutions actuelles d'interfaces de communication ne permettent pas de répondre correctement à ces besoins. C'est pourquoi nous proposons de faire également évoluer les mécanismes d'une de ces interfaces vers un schéma de communication à qualité de service afin d'y remédier.

2.1 Les évolutions dans le monde du calcul intensif

La dynamique qui anime le domaine du calcul numérique se traduit par une évolution permanente, que ce soit en terme matériel ou en terme logiciel. Ces deux aspects indissociables permettent à l'informatique de perdurer mais apportent leurs lots de complications.

2.1.1 Les évolutions matérielles

Depuis ses débuts, l'informatique n'a cessé de se développer toujours plus vite vers des vitesses de traitement et de calcul de plus en plus élevées. À en croire Gordon Moore : « *Tous les 18 mois, la finesse de gravure augmente d'un facteur 2. Ce qui induit une croissance exponentielle régulière des performances des processeurs* ». Cette célèbre loi s'est constamment vérifiée au cours du temps et permet d'avoir un bon ordre de grandeur des performances des futurs processeurs.

Pourtant il y a peu de temps, le problème de la dissipation thermique rencontré par les constructeurs a eu un effet de ralentissement : en dépit de la taille plus faible des composants, la montée en fréquence ne pouvait s'effectuer sans faire courir de sérieux risques de détérioration aux puces.

Pour le moment, les constructeurs contournent le problème en introduisant du parallélisme au sein des processeurs [11]. Cela se traduit tout d'abord par la duplication de certains circuits pour effectuer une instruction avant même que la précédente soit finie : processeurs superscalaires (par exemple i486). Puis les constructeurs en sont venus à dupliquer l'unité de contrôle qui se charge d'aiguiller les instructions vers les circuits qui les exécutent. C'est ce qu'on appelle les SMT (Simultaneous MultiThreading), comme par exemple l'Hyper

Threading chez Intel (Pentium 4). À côté de cela, des architectures ont évolué vers un schéma parallèle plus prononcé : les SMP (Symetric Multi-Processor) où plusieurs processeurs équivalents partagent la mémoire, le disque, les cartes réseau, etc. Et depuis peu, la tendance reprend à peu près le même schéma mais à l'intérieur d'une même puce : les processeurs multi-cores, bien connus du grand public. Il s'agit ici de processeurs SMP à moindre coût : Intel Pentium D, AMD Opteron, Intel Core 2 Duo, etc.

Face à cette croissance du nombre d'unités de calcul, le nombre de cartes de communication reste sensiblement le même. D'une part, les problèmes physiques, dûs au bus mémoire qui n'est pas de taille infinie et dûs au nombre de connecteurs des cartes mères de ces machines, empêchent les administrateurs de telles grappes d'en doter les machines. D'autre part, le prix de la technologie réseau n'est pas négligeable, en particulier en ce qui concerne les switchs (environ $ 100.000 pour un switch Myrinet 128 ports et $ 700 pour une carte réseau contre $ 4.000 pour le prix d'une machine).

En résumé, un déséquilibre matériel très prononcé apparaît : avec d'un côté de plus en plus de flots de données pouvant être générés en parallèle, et de l'autre côté une capacité comparativement réduite à pouvoir les traiter efficacement en simultané.

2.1.2 Les évolutions logicielles

Afin de faire face à la complexité de programmation des applications au dessus des environnements que sont les grilles de calcul, de nombreux modèles de programmation ont été proposés. La plupart d'entre-eux se contentent d'étendre un unique paradigme de programmation comme par exemple les implémentations pour grilles de MPI (MPICH-G2 [10], etc.), l'appel de fonction à distance (GridRPC [19], etc.)

Or, une application a fréquemment recours à plusieurs paradigmes de programmation. Un exemple courant est le couplage de code : la recherche des meilleurs rendements thermiques dans les moteurs aéronautiques nécessite de faire dialoguer des solveurs d'équations de mécanique des fluides et de thermo-mécanique. Dans ces cas-là, les codes échangent régulièrement des informations sur l'interface de couplage. De surcroît, les applications tirant parti du couplage de code font souvent intervenir plus de deux codes issus de différents milieux. Ce qui introduit de nouveaux paradigmes et d'autant plus de flots de données à gérer. Enfin, pour coordonner tous ces modules et ainsi maîtriser la complexité de l'application et son déploiement, un modèle de programmation par composants est fréquemment utilisé.

Nous constatons que de nombreuses communications (échanges d'informations sur le calcul, transferts de données pour leur stockage, échanges de messages pour la gestion des modules, etc.) s'effectuent. Face à cette multiplication de flots de données, nous devons nous questionner sur le comportement des interfaces de communications actuelles.

2.2 Les réseaux rapides et leurs interfaces

Pendant longtemps, les super-calculateurs ont dominé le monde du calcul hautes-performances (CRAY, SGI, DEC, IBM). Mais le manque d'évolutivité et le coût considérable de ces solutions ont diminué leurs attraits. Les constructeurs ont alors commencé à agglomérer des unités de calcul individuellement moins puissantes mais d'un meilleur rapport performance/prix une fois réunies. Les clusters, ou grappes, se sont donc naturellement présentés comme une alternative aux super-calculateurs. Ainsi, au vu des résultats du der-

nier TOP500 [20], les grappes de calculs constituent désormais 72 % des machines les plus puissantes au monde.

Afin de garantir de telles performances, et de ce fait, de se rapprocher des vitesses de transferts des bus mémoires, de nombreux progrès ont été réalisés dans le développement des réseaux rapides. Citons entre autres le réseau MYRINET qui affiche 2,3 μs de latence et un débit de 1,2 Go/s. Plus récemment, est apparu également le réseau INFINIBAND dont les performances sont similaires.

Disposant de tels réseaux, nous serions tentés d'imaginer que l'utilisation des sockets Unix permettrait un dialogue efficace entre les processeurs des différentes machines. Malheureusement les messages en provenance de l'application devraient traverser un nombre considérable de couches logicielles, celles du protocole TCP/IP, ce qui augmenterait énormément la latence (de 3 μs à 50 μs dans le cas du réseau MYRINET). Un support logiciel rajoute inévitablement un surcoût, mais ce surcoût doit être le plus petit possible pour ne pas faire perdre tout intérêt à l'utilisation du matériel sous-jacent. À cet effet, les constructeurs fournissent des interfaces de programmation (MX/MYRINET [15], ELAN/QSNET[18], SISCI/SCI [9], etc.) dites de bas-niveau, car ce sont les plus proches du matériel. Elles offrent des performances comparables aux capacités théoriques du matériel.

Pour minimiser le surcoût induit, les interfaces de programmation des réseaux rapides mettent en œuvre deux techniques en particulier :

Communications en mode utilisateur L'accès à une carte réseau étant contrôlé par le système d'exploitation, toute émission ou réception de message entraîne un surcoût principalement dû à un appel système. Le procédé pour y déroger est tout simplement de contourner le noyau du système pour dialoguer directement avec la carte réseau. À l'initialisation du programme, l'interface de communication établit avec le noyau des projections mémoires entre la carte et l'espace d'adressage utilisateur : on nomme ce mécanisme les communications en mode utilisateur (Figure 2.1).

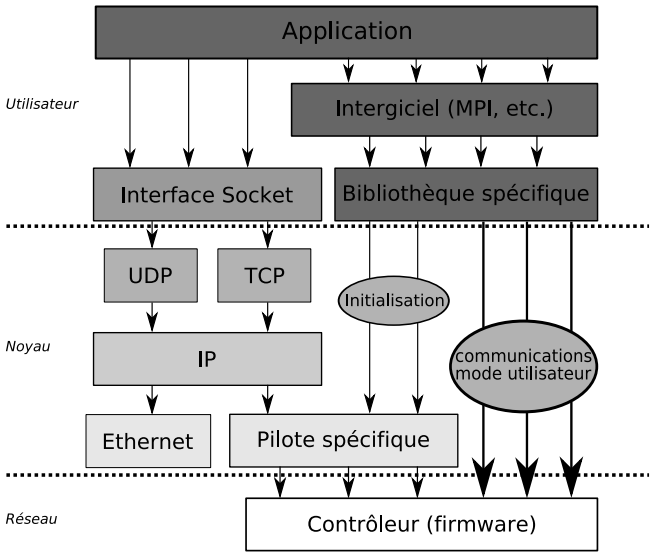
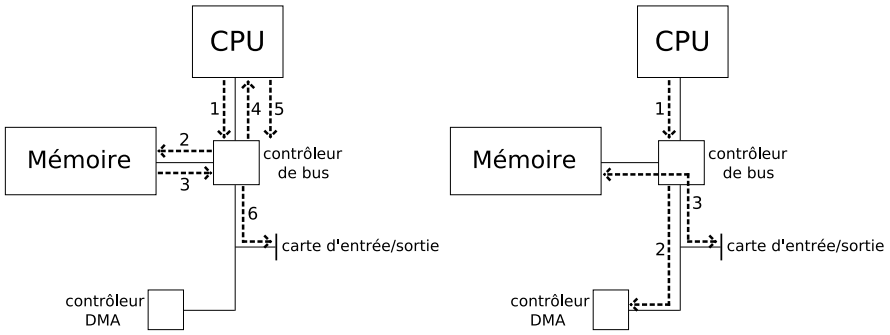


FIG. 2.1 – Les communications en mode utilisateur.

Transferts zéro-copie Les transferts locaux de données entre une carte et la mémoire d'un PC peuvent se faire selon 2 méthodes (2.2) : soit en réalisant un transfert bloc par bloc initié par le processeur et transitant dans une zone mémoire réservée au système d'exploitation (mode PIO : Programmed Input/Output), soit en mode DMA(Direct Memory Access), où les transferts se font en continu et directement vers l'emplacement de la carte ou vers la mémoire, en utilisant le processeur uniquement pour initier le transfert(plus précisément pour initialiser le contrôleur DMA). Le gain se réalise alors lorsque l'on distingue la taille des messages à émettre. En effet, le temps d'initialisation du mode PIO étant plus court, les performances seront meilleures pour de petits messages qu'avec le mode DMA, que l'on favorisera pour les messages de taille plus importante. Le mode DMA libère ainsi le processeur et autorise l'application à l'utiliser pendant les transferts de données : ce qui est appelé un recouvrement.



(a) mode PIO (b) mode DMA

Fig. 2.2 – Transmissions locales de données.

Ces interfaces sont donc les plus efficaces en terme de performances mais, en contrepartie, l'abstraction réalisée reste minime et les primitives offertes sont étroitement liées à la technologie (Myrinet, QSNet, etc.) Cela pose un problème de portabilité : si une application est développée pour utiliser les méthodes fournies par une interface de ce niveau, il faudra obligatoirement la modifier pour qu'elle puisse utiliser un autre type de réseau. Pour remédier à cela, des interfaces de plus haut niveau d'abstraction ont été conçues.

2.3 Les solutions actuelles à paramétrage statique

L'objectif principal des interfaces de haut niveau est de faciliter la programmation des applications nécessitant des communications. D'une part, elles permettent à ces applications de ne pas se soucier du type de réseau sous-jacent. D'autre part, elles offrent un panel fonctionnel relativement riche pour effectuer de nombreuses opérations.

Les multiples implémentations du standard MPI en sont de bons exemples. Ce standard définit un ensemble de spécifications et de fonctionnalités qui ne fait aucune supposition sur le matériel réseau sous-jacent. Cela garantit une portabilité totale pour le développeur d'applications.

Néanmoins, les performances offertes par ce type d'interface sont moins bonnes que celles d'interfaces bas-niveau. Ces interfaces font des compromis sur les paradigmes utilisés pour offrir leur interface générique. L'application ne peut pas spécifier ses contraintes , sauf dans

certaines implémentations de MPI qui autorisent à modifier le seuil PIO/DMA. Mais ce n'est pas modifiable pendant l'exécution de l'application, c'est-à-dire qu'il n'est pas possible de stipuler dynamiquement ses choix à la bibliothèque de communication.

Les interfaces de communication haut niveau font souvent appel à des supports exécutifs qui fournissent une abstraction moins poussée et offrent une portabilité des performances (l'application obtient les meilleures performances pour chaque réseau). En se rapprochant du matériel, ce type d'interface permet alors de spécifier beaucoup plus de contraintes applicatives.

2.4 Le paramétrage dynamique de NewMadeleine

La bibliothèque NewMadeleine est développée au sein de l'équipe RUNTIME pour le support exécutif PM2 [16]. Contrairement aux bibliothèques de communication présentées précédemment, elle présente l'avantage de pouvoir appliquer dynamiquement, au cœur de son moteur d'optimisation, des stratégies d'ordonnancement et d'optimisation génériques. NewMadeleine déclenche des opérations en fonction de l'état des cartes réseau, de l'état de la machine (lorsqu'un processeur devient inoccupé, etc.), à la liste des requêtes en cours, à ce que l'application demande (recouvrement ou attente), pour que les communications progressent de la meilleure façon possible.

Dans la suite, nous allons exposer l'architecture et décrire le fonctionnement de NewMadeleine.

2.4.1 Architecture

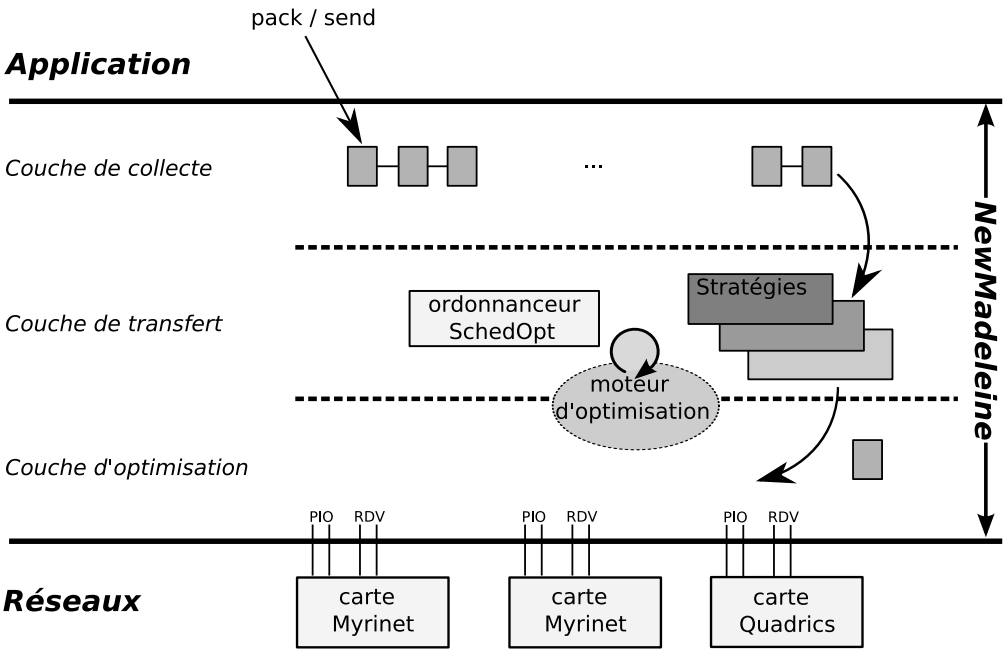


Fig. 2.3 – Architecture de NewMadeleine.

L'architecture de NewMadeleine (2.3) se décompose en trois couches qui correspondent aux différentes étapes de l'évolution d'un paquet.

La couche de collecte La couche de collecte est en charge de récupérer les données fournies par les différents flux. Elle les encapsule et rajoute les informations servant à leur identification (identifiant de l'expéditeur, numéro de séquence, etc.) Les données collectées sont alors placées soit sur une liste spécifique au type de réseau renseigné par l'application, soit sur une liste générale dans le cas par défaut, pour permettre à l'ordonnanceur de répartir les données sur toutes les cartes réseau disponibles.

La couche d'ordonnancement et d'optimisation La couche intermédiaire est chargée de l'ordonnancement des données et de l'optimisation de leur transfert. Elle s'organise pour former des paquets à fournir aux cartes réseaux disponibles à partir de ces données tout en respectant les contraintes applicatives et en garantissant de bonnes performances. C'est cette couche qui assure la mise en place des protocoles réseaux assurant le bon déroulement des communications (rendez-vous, etc.)

La couche de transfert La couche de transfert est la plus proche du matériel. Elle se charge de vérifier l'état d'occupation des cartes réseaux et de signaler à la couche supérieure la disponibilité de celles-ci pour l'envoi de nouveaux paquets. Elle transmet également les données en réception à la couche d'ordonnancement-optimisation.

Une fenêtre de travail se constitue donc tout naturellement, dans laquelle vont s'accumuler les données lorsque les cartes réseau sont déjà occupées. Puis lorsqu'une carte devient libre, le moteur d'optimisation de NewMadeleine va extraire de cette fenêtre une requête, ou en générer une à partir de plusieurs, afin d'approvisionner les cartes réseau.

2.4.2 Ordonnancement et stratégies

Au cœur de cette mécanique se trouve donc le moteur d'optimisation de NewMadeleine. C'est en son sein que se déroule la fonction chargée de sélectionner (ou de générer, par exemple avec une accumulation des données) le prochain paquet à soumettre à chacune des unités inactives. De nombreux arguments potentiels paramètrent cette fonction : la taille de la fenêtre de travail, les informations propres à chacun des paquets, les caractéristiques nominales et fonctionnelles du réseau, ainsi que la politique d'ordonnancement des paquets.

Tous les flux applicatifs vont être projetés sur les unités de multiplexage logiques selon ces différents paramètres. Les messages courts seront par exemple envoyés directement sur le réseau, alors que les messages dont la taille excède un certain seuil vont être stockés et transférés plus tard en attendant que le récepteur confirme qu'il est prêt à le recevoir (protocole de type rendez-vous).

Face à autant de paramètres, il est possible d'user de plusieurs tactiques d'optimisations, mais combiner de façon efficace ces tactiques demeure un problème difficile. Les développeurs ont alors choisi de proposer un ordonnanceur programmable : SchedOpt. La fonction d'optimisation globale est alors définie au travers de l'écriture de stratégies. Chacune de ces stratégies implémente une liste d'opérations élémentaires organisant le comportement global de l'ordonnancement selon un objectif particulier.

2.4.3 Description du rôle d'une stratégie dans l'optimiseur SchedOpt

Concrètement les stratégies réunissent un certain nombre de méthodes appelées au cœur de l'ordonnanceur. L'objectif de ces stratégies est de définir un schéma d'ordonnancement optimal en fonction de la politique attribuée. C'est notamment le rôle d'une stratégie de mettre en œuvre le protocole rendez-vous et de définir le seuil associé.

Une des fonctions des stratégies est de traiter l'envoi d'un paquet. C'est à cet endroit qu'est prise la décision d'envoyer une demande de rendez-vous avant d'émettre les données ou bien de placer les données dans la liste des paquets prêts à être transmis.

Lorsqu'une des cartes réseau est disponible, l'optimiseur fait appel à la stratégie en cours pour qu'elle effectue sa passe d'optimisation (dans laquelle elle peut agréger certains paquets avec d'autres pour former la requête) et ainsi obtenir le prochain paquet à transmettre à la carte libre.

2.5 Problématique

L'augmentation du nombre de modules composant une application de calcul scientifique entraîne, jointe à l'essor du nombre de processeurs dans les machines, une multitude de flots qui dépasse de loin les capacités de multiplexage physique. Ces capacités sont très réduites car le multiplexage occasionne une consommation des ressources mémoire embarquées sur ces cartes. Les cartes ne sont pas prévues pour desservir autant de flux : seulement 3 unités de multiplexage par défaut sur une carte de communication Myrinet avec MX, par exemple.

De plus, ces flux n'ont pas tous les mêmes besoins quant à leur utilisation du réseau. Certains demandent par exemple à être transmis au plus vite, d'autres seront de nature moins urgente, et d'autres encore voudront "recouvrir" les communications (mode asynchrone) par du calcul plutôt que d'attendre la bonne réception du message.

Un multiplexage logiciel doit donc être mis en place pour desservir tous ces flots. Par ailleurs, le module d'ordonnancement de la bibliothèque de communication doit fournir des garanties à chacun de ces flux et ainsi répondre à leur besoins.

2.6 La qualité de service

Fournir des garanties, répondre aux besoins correspondent au concept de la qualité de service. Nous proposons ici de décrire ce concept, propre au réseaux "classiques", au travers de ses différentes implémentations en vue de s'inspirer des travaux déjà réalisés, choisir ceux coïncidant avec nos attentes, et ainsi les adapter aux réseaux rapides.

2.6.1 Principe

De nos jours, l'Internet achemine des paquets en provenance de toutes sortes d'applications : des applications d'accès à un serveur (Mail, FTP, etc.), des applications reliant plusieurs utilisateurs (téléphonie, etc.), et bien d'autres. Quelles qu'elles soient, ces applications détiennent des marges de valeurs tolérées par rapport aux délais de transmission, aux variations de ces délais, à l'intégrité des données transportées. C'est sous ces différents aspects que se traduit la qualité de service. Elle consiste donc à fournir des garanties (latence, débit, taux d'erreur, etc.) aux différentes applications selon les besoins qu'elles auront exprimés.

Un mécanisme de qualité de service devra donc être capable de fournir des garanties à une application tout en assurant que les besoins exprimés par les autres applications soient toujours satisfaits.

2.6.2 Techniques

De nombreux efforts de recherche sur la qualité de service ont été réalisés. Beaucoup d'entre eux portent par exemple sur les réseaux IP et le domaine de l'Internet [12].

Les approches

À l'origine, le protocole IP n'était pas prévu pour prendre en compte les nombreux aspects liés à la qualité de service. Le seul service fourni, appelé Best Effort, était minimaliste. Concrètement, son objectif était de maximiser l'utilisation des ressources et de simplifier l'utilisation le fonctionnement des équipements d'interconnexions. Cependant, les débits des réseaux ont grandement évolué, et les applications de l'Internet également : de nouveaux besoins sont apparus (voix sur IP, etc.) et nécessitent une réelle qualité de service.

Deux approches ont été proposées par l'IETF (Internet Engineering Task Force) pour gérer la bande passante et fournir une qualité de service de bout en bout. La première, IntServ, propose une garantie stricte : après négociation avec les gestionnaires de qualité de service déterminant si le service peut être garanti, les ressources sont réservées, offrant un lien de communication à qualité constante en terme de débit et de latence. Ce schéma, bien que performant, se heurte au problème du passage à l'échelle. En effet chaque élément d'interconnexion doit garder en mémoire l'état des flux qui le traversent, ce qui est inconcevable en raison de la multitude de flux circulant sur Internet. Afin de résoudre ces problèmes, l'IETF a proposé une autre solution : l'architecture DiffServ [4]. Ce modèle propose de séparer le trafic par flots mais par classes de flots (Behaviour Aggregate). Ce qui limite ainsi le nombre de comportements au niveau de chaque nœud. De cette manière, chaque paquet est doté d'une étiquette qui déterminera le traitement (Per Hop Behaviour) que celui-ci obtiendra du réseau (règles de mise en file d'attente, d'ordonnancement, lissage).

Les algorithmes d'ordonnancement

Quelle que soit la politique de qualité de service à laquelle on a recours, le traitement s'effectue toujours au niveau des listes des paquets en attente. Dans cette partie nous allons décrire plusieurs algorithmes d'ordonnancement utilisés dans les routeurs [21] [6].

Fisrt In First Out (FIFO) C'est le plus simple algorithme. Les paquets sont délivrés suivant leur ordre chronologique d'arrivée. Il ne permet pas de différencier les paquets entre eux mais il a l'avantage d'être simple et donc rapide.

Priority Queuing Une priorité est affectée à chaque file d'attente du routeur. Chaque paquet est réparti selon sa priorité et le routeur desservira en premier lieu les paquets de plus haute priorité. Cela permet de gérer le trafic critique mais pour les flux de basse priorité se pose alors un problème de famine (lorsque le trafic haute priorité est important).

Round robin Pareillement à PRIORITY QUEUING les paquets sont répartis dans un certain nombre de files suivant leur classe, ensuite, un tourniquet alterne les paquets à servir parmi les files présentes.

Weighted Fair Queuing Cet algorithme est une émulation du ROUND ROBIN bit à bit. Les paquets sont donc répartis dans des files d'attente et sont ensuite servis bit à bit, et de façon circulaire. Chaque file peut être pondérée pour donner plus ou moins de bande passante à chaque classe de flots. Weighted Fair Queuing est relativement coûteux en ressources car le routeur doit calculer à chaque émission combien de paquets de chaque source seront émis.

La gestion des files d'attente

Un autre point relatif aux files d'attente des routeurs est la gestion de ces files, notamment en ce qui concerne la prévention de la congestion. Essentiel dans le domaine de l'Internet, il n'est pas utile d'en détailler le fonctionnement dans notre cas, les réseaux rapides ne souffrant pas de ce genre de problème.

2.7 Conclusions

Les évolutions dans le monde du calcul scientifique ont engendré une multiplication des flux de communication concurrents au sein des application. D'une part, ces évolutions sont liées au matériel où la croissance du nombre d'unités de calcul par machine (architectures parallèles) s'est opposée à la limitation du nombre de cartes réseau sur chacune de ces machines. D'autre part, elle sont également liées aux logiciels qui se complexifient continuellement pour à la fois exploiter efficacement le matériel sous-jacent, mais également pour correspondre à de nouveaux modèles de programmation. Le nombre de flux logiciels devient bien supérieurs aux possibilités de multiplexage matériel.

Face à cette multiplication de flux, plusieurs solutions s'offrent au développeur d'applications pour interagir avec le réseau rapide. Les premières d'entre-elles maximisent les performances mais sont étroitement liées à un réseau particulier. Pour le second cas, la portabilité est privilégiée mais aux dépens des performances. Dans la dernière catégorie, les interfaces tentent de garantir la portabilité des performances pour chaque type de technologie réseau. Certaines de ces solutions, comme c'est le cas pour NEWMADELEINE, autorisent l'application de diverses optimisations. Elles réalisent un mixage des flux et optimise les communications de manière globale. Mais ces solutions ne prennent pas en considération les besoins propres à chaque flux.

Le concept de qualité de service offre une réponse attrayante aux problèmes de multiplexage logiciel soulevés. Des divers travaux accomplis jusqu'alors, a résulté une formidable quantité de mécanismes destinés à satisfaire les besoins des flux de l'Internet. Mais ces méthodes, aussi performantes soient-elles, conviennent elles aux technologies des réseaux rapides ?

Chapitre 3

Contribution

Dans ce chapitre, nous allons exprimer les besoins caractéristiques des différents flux parcourant les réseaux hautes performances. Puis nous allons dépeindre une proposition d'architecture logicielle intégrée au cœur de NewMadeleine et réalisant un mixage dynamique répondant aux contraintes exprimées par les flux applicatifs.

3.1 Réflexions préliminaires sur le contexte particulier des réseaux rapides

Attachons nous tout d'abord à décrire toutes les particularités qui entourent le domaine des réseaux rapides. Nous obtenons ainsi une vision claire de la démarche à entreprendre pour répondre aux besoins adéquats des applications ciblées.

3.1.1 Les caractéristiques physiques des réseaux rapides

Les réseaux rapides utilisés à l'intérieur des grappes de calcul n'ont pas pour objectif de relier des machines séparées de plus de quelques mètres. Les constructeurs ont donc développé des technologies capables de transmettre une information entre deux nœuds en quelques microsecondes et offrant des débits bien supérieurs aux technologies d'interconnexion classiques. Les réseaux utilisés ont comme caractéristique une faible latence, ce qui induit que lorsque l'application désire envoyer un message, elle va passer les données à la bibliothèque de communication qui devra les remettre à la carte réseau dans un laps de temps très court. Le temps de décision (ordonnancement, agrégation, etc.) doit effectivement être le plus court possible sous peine de voir chuter les performances inhérentes aux communications. Par déduction, les algorithmes d'ordonnancement qui vont être implantés n'auront pas les mêmes ordres de grandeur que ceux issus des études portées sur la qualité de service dans les réseaux classiques. Par ailleurs, dans notre contexte, les communications sont considérées comme fiables.

3.1.2 Le trafic dû au calcul intensif

Les difficultés (ou même impossibilités) qu'éprouvent certains mécanismes à être mis en place dans le domaine d'Internet ont le plus souvent comme origine la masse conséquente de flux le traversant (par exemple le problème de mise à l'échelle de IntServ). Or, ce problème ne se pose pas dans notre environnement, où tout au plus quelques dizaines de flots de

données viendront se disputer les liens de communications. La bibliothèque NewMadeleine offre actuellement un multiplexage de 255 flots différents au maximum, ce qui est amplement nécessaire. C'est pourquoi, nous nous autorisons à garder certaines informations sur chaque flux parcourant l'architecture de communication.

3.1.3 L'absence de congestion

Dans la même optique de simplification, notre solution ne nécessite pas d'inclure les processus servant à prévenir la congestion. Ce phénomène intervient pour ainsi dire rarement dans les réseaux rapides, en raison des spécificités précédemment citées(faible latence, peu de flux). Pour ces mêmes motifs, les techniques de lissage de trafic sont inutiles en ce qui nous concerne.

3.1.4 L'absence de garanties quantitatives

La quantité précise du débit qui sera octroyée à un flux ne peut être garantie. De même en ce qui concerne le temps exact d'attente maximum qu'aura un paquet de ce flux. Contrairement à un routeur qui peut ordonnancer les paquets de ses files selon une échelle temporelle précise, une bibliothèque de communication s'exécute sur un ou plusieurs processeurs qu'elle partage avec les processus applicatifs. Dans ces conditions, offrir une garantie de partage quantitatif de bande passante ou de temps de latence maximum serait un système caduc. Nous pouvons uniquement fournir des garanties à un flux relativement aux autres flux : garantir une latence plus élevée, un débit plus important, une priorité globale, une meilleure réactivité, etc.

3.1.5 Le recouvrement

Beaucoup d'applications obtiennent de sérieux gains en terme de performances quand elles effectuent des calculs pendant les transmissions de messages, au lieu de laisser le processeur inoccupé. À l'inverse, il existe des applications où les machines ont rapidement besoin de recevoir certaines données en provenance des autres nœuds, afin de pouvoir continuer leur propre calcul (par exemple lors d'un couplage de code, le résultat d'une équation mécanique devant être transmis au plus vite à un des nœuds de la grappe qui exécute le code thermique). Il doit donc être possible de répondre aux besoins de chaque flux applicatif à ce niveau là également, par l'usage des techniques explicitées précédemment (PIO et DMA 2.2).

À partir de toutes ces différentes spécificités des réseaux rapides, nous pouvons établir une solution répondant aux besoins des applications.

3.2 Vers une intégration de la qualité de service dans les bibliothèques de communication pour réseaux rapides

Nous proposons d'aller plus profondément dans l'exploitation des informations fournies en amont et d'offrir aux applications la possibilité d'exprimer clairement leurs contraintes. C'est-à-dire garantir une qualité de service dans un environnement où il faut prendre en considération de nombreux paramètres comme la latence, la réactivité, les priorités. Pour cela,

nous pouvons envisager d'implanter au cœur de la bibliothèque de communication NewMadeleine un nouveau module d'optimisation chargé d'orienter chaque flux applicatif vers la tactique d'ordonnancement correspondant au service associé. Le module se charge de mixer tous ces flux de façon cohérente, c'est-à-dire en adéquation avec leurs besoins.

3.2.1 L'interface utilisateur

L'objectif de notre mémoire est de réaliser un mixage dynamique approprié aux contraintes des flux, au sein de l'ordonnanceur de communication NewMadeleine. Ces contraintes ont besoin de s'exprimer pour être prises en compte par l'application. Nous proposons donc d'enrichir l'interface d'échanges par messages de NewMadeleine de plusieurs méthodes. La première de ces méthodes associe la politique de qualité de service au flux passé en paramètres. L'application passe en paramètres, d'une part le choix de sa politique d'ordonnancement et d'autre part l'étiquette du flux correspondant. L'autre primitive définit le degré de priorité affecté aux paquets de certains flux, dans les cas d'utilisation de la politique d'ordonnancement par file de priorité.

3.2.2 Mise en place d'une stratégie pour la différenciation de services : Stratos

Le principe de notre solution s'inspire de celui de la manipulation des flux. Différents besoins sont exprimés par les flux applicatifs, notamment en terme de latence, de débit, de priorité, et l'objectif est de diriger ces flux vers les mécanismes qui fournissent une réponse à leurs besoins.

Ce sont les stratégies qui implémentent les méthodes décidant et appliquant les optimisations à effectuer compte tenu des caractéristiques des données. De plus, l'emplacement qu'occupe les stratégies dans la bibliothèque NewMadeleine leur permet d'obtenir toutes les informations nécessaires à l'arbitrage des flots de données (identificateur du flux, taille des données, etc.)

Il est donc logique que l'intégration d'un système de garanties attribuées aux flux de communication passe par l'écriture d'une nouvelle stratégie dont le but est de rajouter la notion de qualité de service dans NewMadeleine.

L'idée est d'utiliser les étiquettes (ou tags) qui identifient chaque flux par rapport aux autres. Toutes les données qui appartiennent à un flux, ou même à un groupe de flux, sont associées à un traitement particulier (agrégation, placement dans des listes, etc.), et c'est ce traitement qui détermine le service alloué aux flots.

En définitive, la stratégie en question doit « aiguiller » les flux vers des files d'attente. Chaque file possède alors sa propre politique d'ordonnancement. Notre solution peut s'interpréter comme l'ajout d'un degré de récursivité au principe de fonctionnement actuel de NewMadeleine : la stratégie principale (dénommée Stratos) va appliquer une autre stratégie parmi celles dont elle dispose sur les paquets que l'on lui transmet.

Sur la figure 3.1, nous voyons le parcours entrepris par les flux de données. Chaque requête postée par le niveau applicatif est placée dans les listes de la couche de collecte qui va se charger de fournir à la couche d'ordonnancement. Le moteur interne de la stratégie Stratos va alors retrouver la politique qu'il faut appliquer à la requête (selon son étiquette) et la transmettre à cette même politique. Cette dernière se charge ensuite d'effectuer ses optimisations et de fournir un paquet à émettre (si disponible) lorsque Stratos en fera la demande.

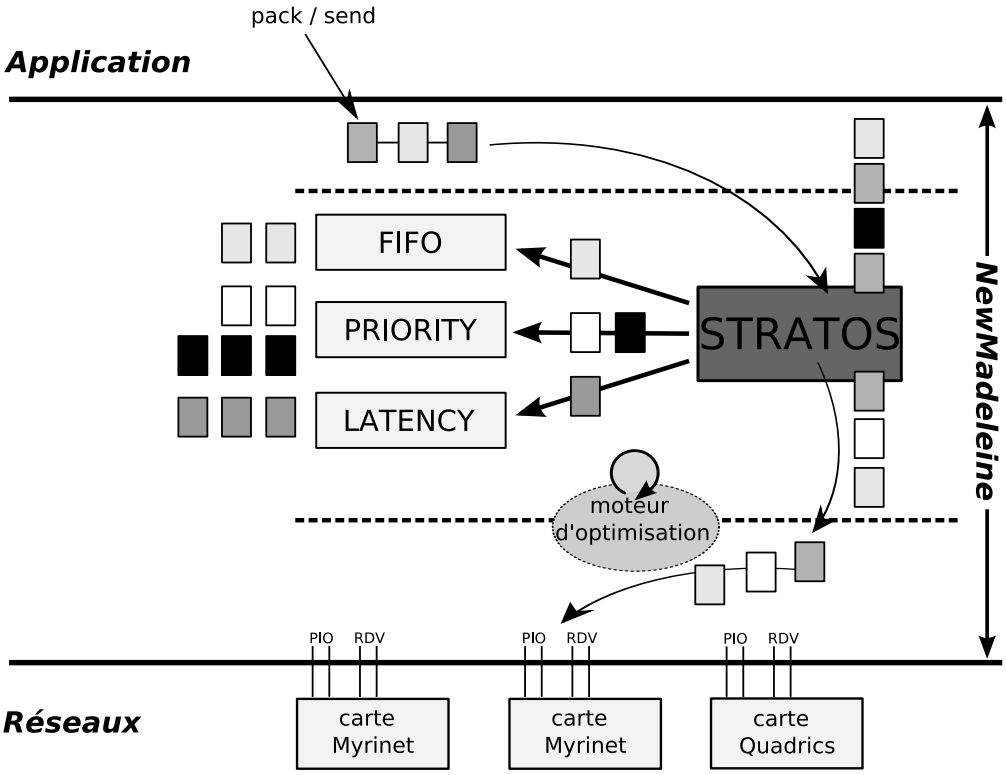


FIG. 3.1 – Architecture de la stratégie à différenciation de services

3.2.3 Les politiques d'ordonnancement

Une politique d'ordonnancement a la charge de fournir des garanties aux paquets qu'elle gère. Elle dispose d'un ensemble de méthodes (empaquetage des données, optimisation, etc.) pour arriver à ses fins. La politique effectue toute seule la gestion des paquets que l'on veut bien lui fournir. Lorsque des données sont transmises par l'application (indirectement), la politique est appelée si elle a la charge du flux correspondant. Elle peut alors décider d'empaqueter immédiatement les données dans une structure prête à être transmise, ou bien de les agréger à d'autres requêtes, ou encore d'envoyer une demande de synchronisation avec le récepteur et de placer les données dans une liste temporaire (protocole rendez-vous). Par la suite lorsqu'une carte réseau devient libre, et que STRATOS décide de lui « donner la main », elle va exécuter sa fonction d'optimisation pour décider de la requête qu'elle transmet à la carte. Son choix peut très bien ne pas être trivial (le premier de la liste) et consister à transmettre le premier paquet dont la taille est inférieure à un certain seuil, ou bien le premier paquet de la liste la plus prioritaire.

C'est cette facilité d'implémentation qui va nous autoriser à ajouter plusieurs politiques aisément, à écrire ses propres algorithmes . Il suffit pour cela de remplir les quelques fonctions d'initialisation de structure, et les deux méthodes explicitées précédemment.

Cette optique ressemble un peu au concept des stratégies. Actuellement, ce n'est pas encore mis en place mais le but des stratégies est, à terme, de pouvoir réaliser une simulation d'optimisation qui renvoie alors un score, permettant d'apprécier les performances obtenues. Le moteur de l'optimiseur choisit ainsi la stratégie qui obtient le meilleur score. De la même façon, la stratégie implémentée dans ce mémoire dispose de plusieurs politiques d'ordonnancement, chacune accomplissant diverses actions. La différence provient du fait que ce sont les processus applicatifs qui choisissent la politique à appliquer parmi le panel disponible. Les politiques s'assimilent donc à des « mini-stratégies ».

En conséquence, c'est à l'intérieur de ces politiques que l'on va mettre en place les algorithmes inspirés des travaux sur la qualité de service.

3.3 Fonctionnement de Stratos

La mécanique de notre implémentation se fonde sur le fonctionnement des stratégies de NEWMADELEINE. Chacune des politiques le composant utilise un algorithme différent mais toujours dans un objectif d'ordonnancement.

3.3.1 Les algorithmes des différentes politiques

Nous décrivons ici les algorithmes utilisés dans les différentes politiques d'ordonnancement.

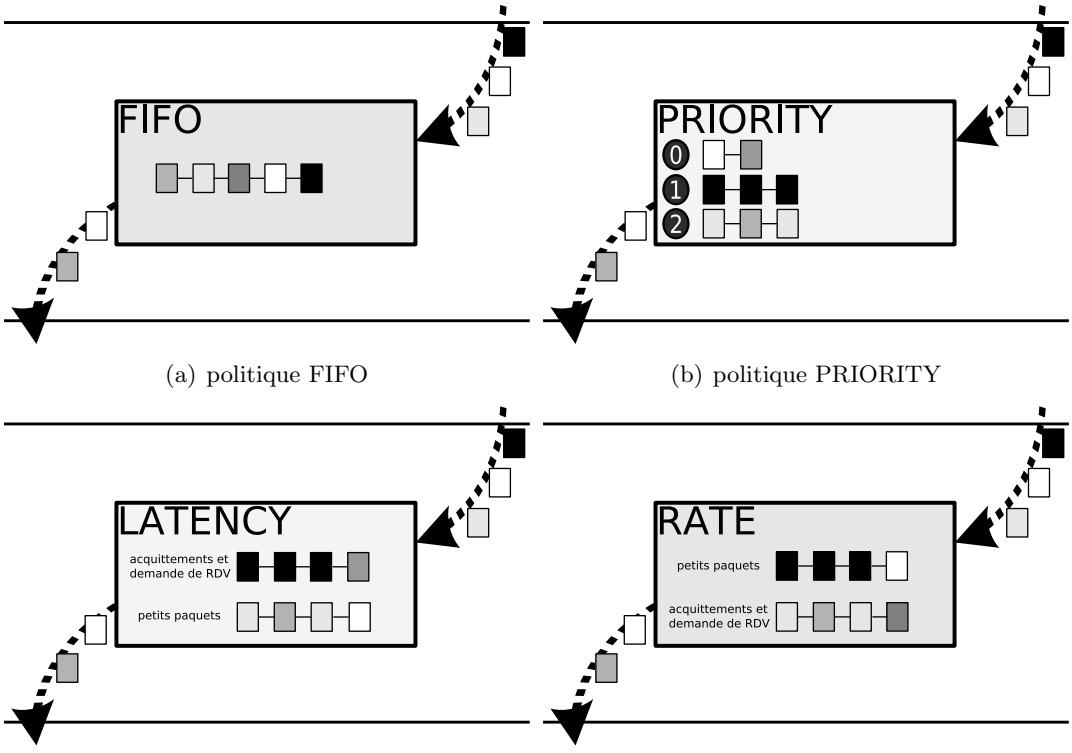
FIFO Aucun traitement particulier n'est appliqué dans cette politique. Tous les paquets sont placés dans une même liste et le premier arrivé est le premier transmis. C'est le fonctionnement de base de la stratégie par défaut de NEWMADELEINE. Cette politique est utilisée pour tous les flux qui n'ont pas spécifié de politique particulière.

PRIORITY Dans cette politique, plusieurs files d'attente sont disponibles (à l'heure actuelle trois), et correspondent chacune à un niveau de priorité. Les flux de cette politique sont

alors placés dans la file de la priorité qui leur a été affectée. De ce fait, lorsque la fonction d'optimisation est appelée par la stratégie, les paquets de plus haute priorité sont desservis en premier.

LATENCY Cette politique n'est pas directement issue des travaux sur la qualité de service puisque son objectif n'est pas de garantir une latence minimum, mais plutôt de privilégier les flux qui délivrent des paquets de petite taille. On retrouve une notion de priorité mais non plus entre chaque flux mais qui peut être au sein d'un même flux. En géneral, les applications délivrant les flux de ce type vont vouloir se transmettre des petites données en urgence.

RATE Il s'agit de la politique inverse à la précédente étant donné que ce sont les paquets nécessitant une demande de rendez-vous qui vont être privilégiés. En effet, ce type de paquet va occuper beaucoup plus de bande passante que ceux transportant moins de données. Pour élaborer cet algorithme, les demande de rendez-vous sont délivrés en priorité. Ainsi, le récepteur pourra émettre son acquittement plus rapidement que dans le cas où aucune distinction n'est faite.



(a) politique FIFO

(b) politique PRIORITY

(c) politique LATENCY

(d) politique RATE

FIG. 3.2 – Politiques d'ordonnancement de la stratégie Stratos.

3.3.2 Sélection du prochain paquet à émettre

À l'heure actuelle, lorsque une des cartes réseau est inoccupée, le moteur d'optimisation demande le prochain paquet à émettre à la stratégie. Si cette stratégie est STRATOS, celle-ci

sélectionne une politique parmi celles qui sont actives, c'est-à-dire celles dont les files ne sont pas vides. Le choix de la politique se fait selon une stratégie de tourniquet.

Au premier abord, ce choix pourrait ne pas sembler viable. En effet, prenons l'exemple de la politique LATENCY qui a vraisemblablement besoin d'avoir le plus souvent possible accès aux cartes pour transmettre au plus vite les petits paquets de ces flux, dans cette optique constante de garanties. Le fait de donner le même « quantum » d'accès à la carte à LATENCY qu'à une autre politique telle que FIFO peut surprendre. Un ordre strict entre les politiques pourrait être instauré dans ce genre de cas. Par exemple : tant que la politique LATENCY a des paquets à fournir la stratégie n'invoque pas la politique FIFO. Cependant, ce type de conflit intervient dans des cas des cas où une congestion se forme et nous sommes toujours dans le contexte des réseaux rapides. Or, dans ce type de réseau les phénomènes de congestion sont inexistants.

3.4 L'adéquation qualité de service et optimisations

Lorsque différents flux de communication, dont la politique d'ordonnancement n'est pas la même, progressent de manière concurrente, leurs paquets sont répartis dans différentes files d'ordonnancement. Dans la version courante de NewMadeleine, les paquets sont placés dans une unique liste, ce qui permet d'accumuler plusieurs segments de données à l'intérieur d'une même requête et de réaliser un seul envoi, là où il en aurait nécessité plusieurs. Mais dans notre cas, les paquets sont traités distinctement selon leur flux respectif. Par exemple, dans la politique d'ordonnancement LATENCY, les requêtes de demande de rendez-vous ne sont plus agrégées avec les requêtes de données (de petite taille). Cependant, notre solution autorise l'agrégation au sein de flux ayant des garanties identiques. L'objectif étant de ne pas s'éloigner de l'orientation "performances" de NewMadeleine.

Chapitre 4

Éléments d'implémentation

Dans ce chapitre, nous revenons avec plus de détails sur la mise en œuvre de certains aspects de notre solution.

4.1 Compromis sur l'implémentation des politiques

Au tout début de nos travaux, la question de la récursivité dans l'utilisation des stratégies a été le point de départ. L'idée de base étant d'appliquer un traitement d'ordonnancement/optimisations sur les flux, les stratégies déjà implémentées dans NewMadeleine se sont avérées correspondre exactement avec ce schéma. Pourtant, nous nous sommes astreints à introduire cette récursivité au sein d'une seule stratégie.

Par ailleurs, il a fallu faire un choix entre utiliser directement les structures de données de la bibliothèque NewMadeleine ou alors mettre en œuvre des structures de données plus complexes au sein desquelles on peut intégrer les informations sur les garanties propres aux données transportées. Le nombre de flux étant limité dans le domaine des réseaux rapides, il était moins coûteux de prendre uniquement en compte, de manière globale, les informations de contraintes au niveau de la bibliothèque plutôt que d'associer une politique donnée à chaque requête.

4.2 Priorité dans les acquittements

Tout d'abord, présentons le déroulement précis des opérations lorsqu'un paquet de données à émettre est de taille trop importante pour être contenu dans les tampons mémoire du récepteur (destinés à accueillir les paquets inattendus).

Dans ce cas-là, l'émetteur envoie une demande de rendez-vous (en-tête de données qui peut être agrégé avec d'autres paquets), et attend que le nœud distant soit prêt à recevoir les données en question. Le récepteur, de son côté, prend en considération la demande de rendez-vous et la traite : c'est-à-dire que soit il place un paquet d'acquittement dans la file d'attente des paquets prêts, soit il reporte l'acquittement pour le moment où il disposera de l'espace mémoire censé les accueillir.

Au moment où un paquet est reçu, une fonction est appelée pour itérer sur toutes les entêtes qui le composent. Selon le type d'entête (entête de données, acquittement, demande de rendez-vous), une méthode différente est appelée. Dans le cas des acquittements, la méthode

recherche les données correspondant à l'acquittement et les « poste » sur la liste d'envoi de la carte réseau.

Le récepteur peut très bien avoir plusieurs demandes de rendez-vous à la suite. En conséquence, l'émetteur traite parfois plusieurs acquittements (de par l'agrégation réalisée lorsque des paquets sont placés dans les files). Et ces paquets peuvent ne pas avoir la même priorité de traitement. La garantie de priorité de service n'est plus valable car les paquets vont uniquement être traités dans leur ordre d'agrégation. Le schéma 4.1 expose ce problème.

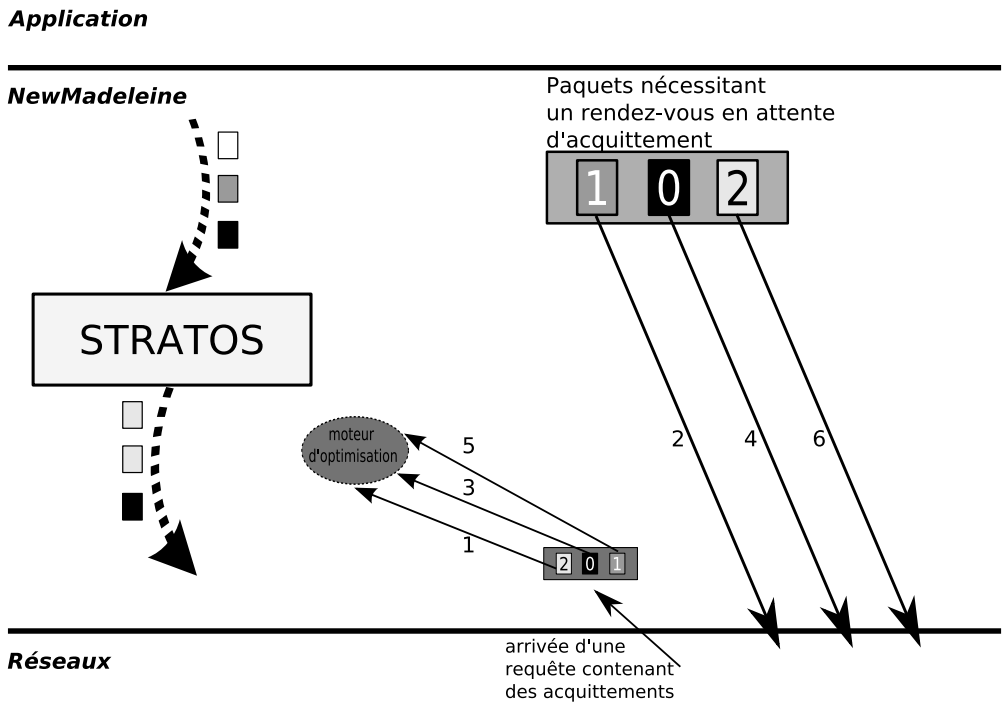


FIG. 4.1 – Traitement des acquittements sans gestion des priorités.

Après ce constat, il est évident que le traitement des acquittements doit se faire différemment, en prenant en compte la notion de priorités. Une méthode enrichit le panel des fonctionnalités de la stratégie STRATOS : cette méthode se substitue à celle appelée lors de l'itération sur les entêtes.

Au lieu de directement transférer les données à chaque fois qu'un acquittement est décelé, l'acquittement est stocké dans une liste correspondant à la priorité des données qui lui sont liées, puis les données sont émises dans l'ordre de leure priorité. On court-circuite ainsi le processus d'itération et les garanties restent valables pour les gros volumes de données. Cette solution est représentée à la figure 4.2.

4.3 Garantie du niveau de réactivité

Une autre notion qu'il est important d'aborder est la gestion de la réactivité. En effet, la mise en place de cette dernière a nécessité d'intégrer le module PIOMAN (PM2 IN/OUT MANAGER [22]), qui est un gestionnaire d'entrées/sorties interagissant avec NEWMADELEINE

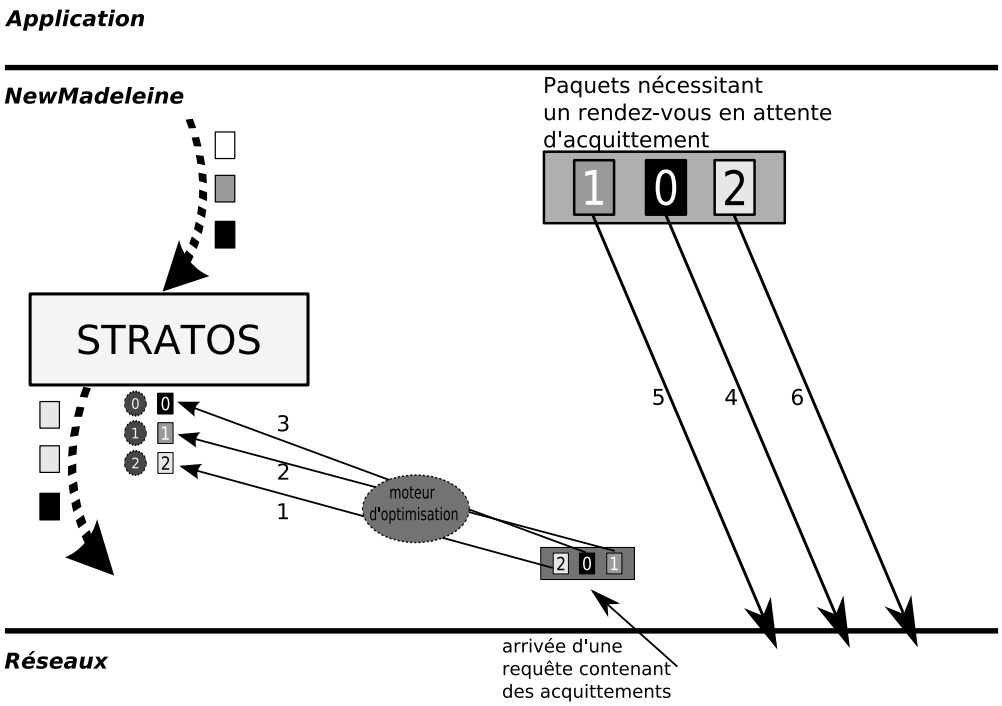


FIG. 4.2 – Gestion de la priorité lors de la réception d'acquittements multiples.

et MARCEL, la bibliothèque de threads du support exécutif PM2. L'objectif de PIOMAN est d'offrir aux bibliothèques de communication un service de scrutation garantissant un temps de réaction indépendant de l'activité des threads de calcul. Le principe de cette architecture logicielle (4.3)se base sur un serveur d'événements asynchrones qui se charge de choisir la meilleure méthode à appliquer pour détecter une communication réseau selon les préférences de l'application. C'est à dire soit attendre l'événement avec un appel bloquant qui permet des temps de réaction très courts (mécanisme d'interruption) mais induit un léger surcoût, soit utiliser une méthode de scrutation qui peut s'avérer plus efficace dans certains cas (processeur libre). Dans le deuxième cas, la scrutation est entièrement gérée par le serveur d'événements.

Pour permettre à l'application de choisir le niveau de réactivité des flots de communication, nous proposons d'intégrer la notion de priorité dans les requêtes traitées par PIOMAN. Dans chaque requête, un degré de priorité définit la fréquence de scrutation de la détection de terminaison. Lorsque plusieurs événements sont détectés le serveur va réveiller le thread de communication de la requête de plus haute priorité. Cette méthode garantit une réactivité accrue pour les requêtes des flux concernés. La modification s'intègre complètement à l'environnement de notre stratégie et n'implique pas l'écriture d'une politique particulière.

L'implémentation sera évaluée dans les prochains mois. La politique est complètement mise en œuvre et fonctionne. Il sera facile d'apprécier l'efficacité d'une telle politique en effectuant un test simple entre deux machines :

1. Plusieurs threads de calculs sont exécutés tout le long de l'algorithme sur le premier nœud.

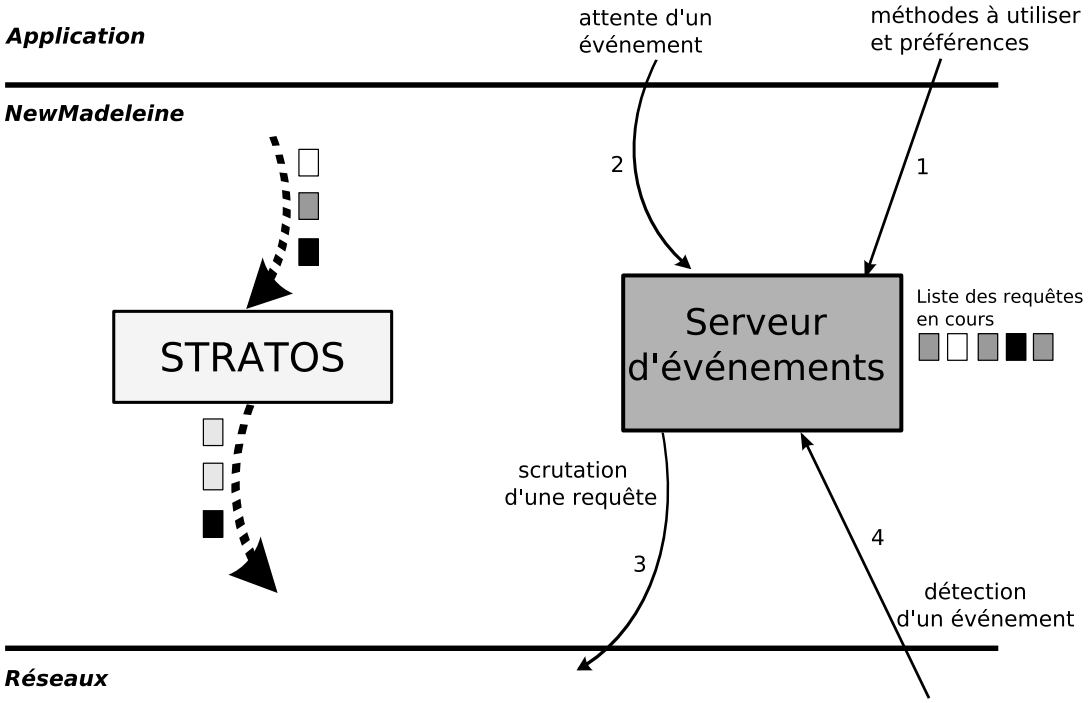


FIG. 4.3 – Fonctionnement du serveur d'événements.

2. Un autre thread est créé sur le premier nœud et envoie un message `a` pour signaler qu'il est prêt à émettre des données.
3. Sur le deuxième nœud un thread en attente du message `a` envoie un message `b` pour signifier qu'il veut recevoir les données.
4. Le thread du premier nœud poste quant à lui une requête au serveur d'événements pour attendre le message `b` et pouvoir ainsi envoyer les données.

Il suffit ensuite de calculer le temps qui s'écoule, du côté récepteur, entre la demande d'envoi et l'arrivée effective de ces données, en faisant varier leur taille. Les valeurs obtenues se comparent alors entre une version de l'expérience utilisant une réactivité plus importante pour le thread de communication que pour les threads de calcul et une version sans priorité particulière.

Ces quelques détails sur la mise en œuvre de notre stratégie confirme qu'il est possible d'intégrer le concept de qualité de service au sein des réseaux rapides. Au chapitre suivant, nous allons mettre à l'essai les différentes politiques de garanties pour les comparer aux stratégies actuelles de NEWMADELEINE.

Chapitre 5

Évaluation

Nous avons réussi à exprimer les différentes contraintes que peuvent manifester les flux applicatifs des réseaux rapides. Nous pouvons donc nous appliquer à évaluer les performances non plus « brutes » mais relatives aux besoins propres à chaque types de flux. La majorité des tests présenté dans ce chapitre a pour objectif de comparer les performances entre le mixage des flux opéré par la stratégie d'agrégation de NewMadeleine (qui a déjà prouvé son efficacité) et celui opéré par les politiques d'ordonnancement proposés dans ce mémoire.

5.1 Environnement de travail

Le tests des sections suivantes on été effectués sur les machines de la grappe de l'équipe RUNTIME : les Dalton. Nous utilisons en particulier les nœuds William et Averell qui sont des bi-Xeon 2,6 GHz disposant chacun de 1 Go de mémoire vive et de 512 ko de mémoire cache. Le système d'exploitation utilisé est Linux 2.6. Chacun des nœuds dispose de carte Myrinet-2000.

5.2 Non régression

Nous commencons par vérifier que les mécanismes et traitements mis en place comme la recherche de la politique associée à une requête ou la gestion des acquittements pour le cas de priorités, n'affectent pas les performances « brutes » de NewMadeleine. Ainsi nous effectuons ici un simple test de type « ping-pong » qui consiste en une série de 2000 allers-retours sur le réseau (afin de lisser les éventuelles fluctuations), avec des données de taille croissante (de 4 octets à 8 Mo). Pour une taille donnée, le temps moyen d'un transfert est obtenu à partir du temps écoulé entre le premier envoi et la dernière réception divisé par deux et par le nombre de tours de boucles effectués.

La courbe 5.1 compare les performances en terme de latence entre la principale stratégie utilisée par le moteur d'optimisation de NewMadeleine et notre stratégie qui oriente les flux vers la politique par défaut, FIFO. Nous observons que les performances sont sensiblement équivalentes. De même en ce qui concerne la courbe 5.2, comparant les débits. Pour les flux ne spécifiant pas de contraintes spécifiques, on garde ainsi les mêmes performances qui font de NewMadeleine une bibliothèque de communication adaptée aux réseaux rapides.

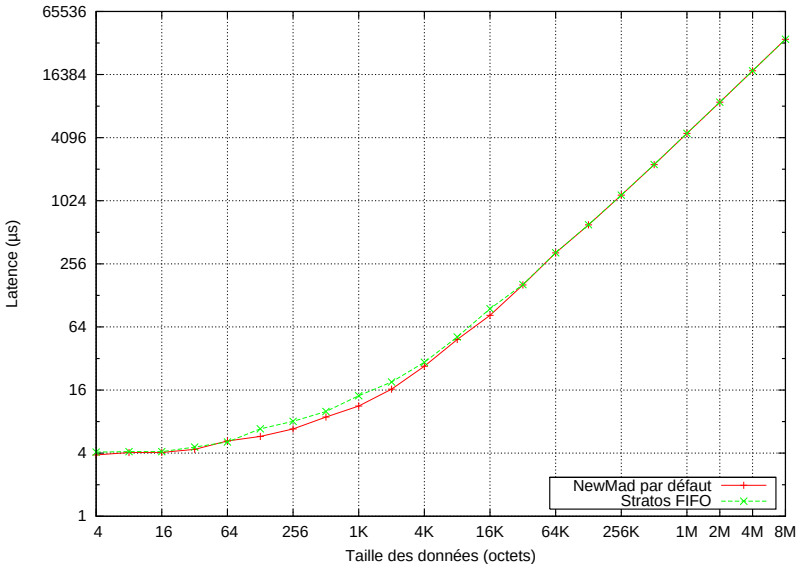


FIG. 5.1 – Ping-Pong (latence) - NewMadeleine par défaut vs Stratos FIFO.

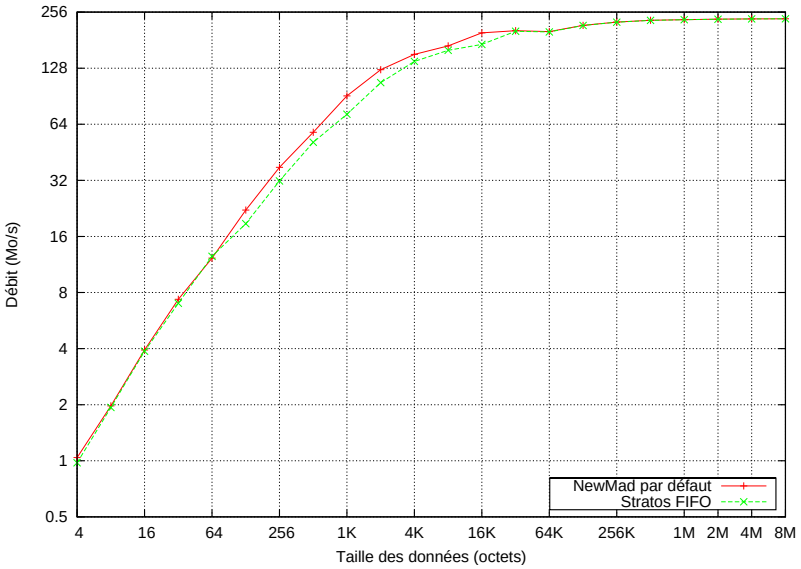


FIG. 5.2 – Ping-Pong (débit) - NewMadeleine par défaut vs Stratos FIFO.

5.3 Priorité

Le second test de performance concerne la garantie de priorité d'un (ou même plusieurs) flux par rapport aux autres flux transitant par l'interface de communication. Les performances des applications utilisant NewMadeleine ne doivent pas être altérées en raison de la présence de flux moins prioritaires. Le test évalue la latence et le débit pour différentes tailles de messages (de 4 octets à 8Mo). Le déroulement des opérations est le suivant :

1. La priorité maximale est donnée au flux `0` et les autres flux ont la priorité la plus basse.
2. Chacun des flux sous-prioritaires envoie une requête d'une taille donnée (les envois sont réalisés en mode non bloquant, c'est-à-dire que l'on attend pas que les données soient réellement envoyées sur le média physique, elles sont uniquement transmises à la bibliothèque de communication qui se chargera de les envoyer).
3. Le flux `0` envoie une requête de la même taille et attend le retour du nœud distant (un message de même priorité).

Nous comparons ensuite le temps mis entre l'envoi du message du flux `0` et la réception du second message prioritaire.

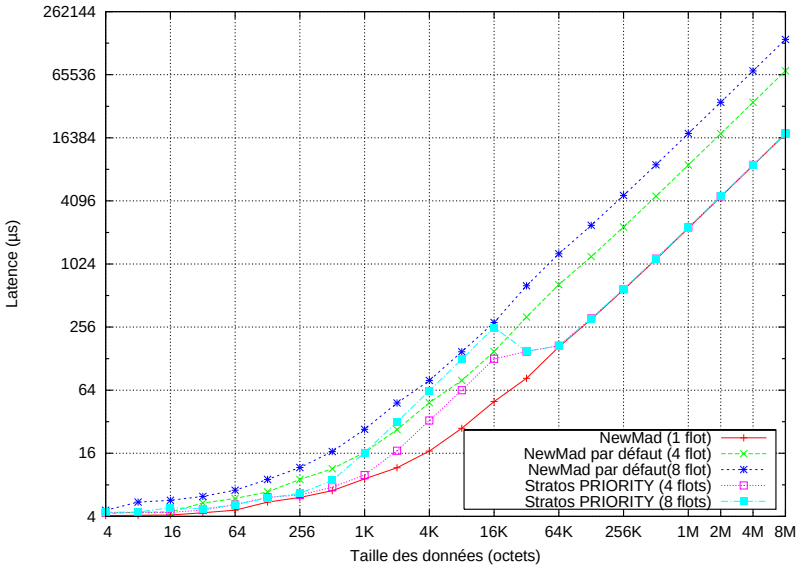


Fig. 5.3 – Transmission d'un flux prioritaire - NewMadeleine par défaut vs Stratos PRIORITY.

Les valeurs observées dans la figure 5.3 comparent le cas où aucune priorité ne différencie les flux (utilisation de la politique principale de NewMadeleine) et celui où le flux dont on calcule les performances, ceci pour différents nombres de flux sous-prioritaires. Le cas particulier où un seul flux (le `0`) transmet sur le réseau nous fournit les valeurs « étalons ». Il s'agit du cas idéal pour lequel aucun autre flux de données de vient perturber le flux `0`. Nous constatons que la gestion des priorités faites dans Stratos permet de conserver les performances pour le flux `0`. En effet quel que soit le nombre de flux intervenant, la stratégie privilégie le flux prioritaire et lui garantit les performances optimales.

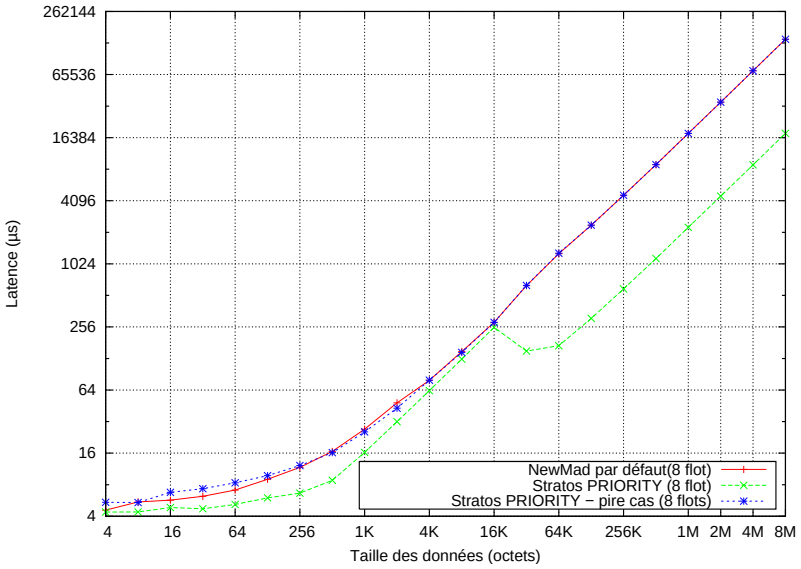


FIG. 5.4 – Comparaison du pire cas possible pour un flux avec NewMadeleine par défaut et Stratos PRIORITY.

Les courbes du schéma 5.4 permet d'observer le comportement des transmissions pour le flux prioritaire. Nous comparons ainsi les valeurs pour le meilleur cas pour lequel le flux `0` est prioritaire par rapport aux autres, avec les valeurs dans le cas où aucune priorité n'est gérée (le comportement initial de NEWMADELEINE), et celles dans le pire des cas où ce sont tous les autres flux qui sont prioritaires par rapport au flux `0`. Nous constatons que la stratégie principale de NEWMADELEINE offre des performances similaires au pire des cas.

5.4 Conclusions

Les valeurs observées dans ce chapitre nous ont permis d'une part de vérifier que l'ajout de la gestion de la qualité de service dans la bibliothèque de communication NEWMADELEINE n'affecte pas ses performances usuelles. D'autre part, nous avons pu souligner l'intérêt de pouvoir définir des contraintes sur les flots de données pour en prendre compte et ainsi adapter au mieux leur multiplexage.

Chapitre 6

Conclusion et perspectives

6.1 Bilan

Dans de nombreux domaines, tels que la recherche ou l'industrie, les exigences en puissance de calcul sont considérables et pourvoir à cet appétit grandissant n'est pas une mince affaire. Pour pallier le problème du coût des solutions dédiées offertes par les grands constructeurs, l'utilisation des grappes de calcul et des réseaux rapides s'est peu à peu répandue. Les applications se basent alors sur des interfaces de communication pour bénéficier des performances offertes par ces solutions. Cependant, les difficultés sont multiples pour ces interfaces. D'une part, les technologies des réseaux sont nombreuses et très différentes les unes des autres. D'autre part, les architectures des machines évoluent régulièrement et deviennent nettement parallèles (multiprocesseurs, multicores, etc.), ce qui entraîne une augmentation du nombre de flots de données à gérer. De plus, les applications ont elles aussi tendance à se complexifier et à générer de plus en plus de flux de communication différents.

Les interfaces de communication classiques composent avec ces difficultés en réalisant un compromis entre performances et portabilité. La plupart d'entre elles font des choix qui les cloisonnent dans un schéma de communication strict (recouvrement ou non, privilégier la latence, etc.) La bibliothèque de communication NewMadeleine quant à elle opte pour un fonctionnement plus dynamique, en optimisant les requêtes fournies par l'application selon plusieurs paramètres (état de la machine, des cartes réseau, etc.)

Dans ce mémoire, nous avons présenté une solution nommée Stratos prenant en considération les besoins propres à chacun des flux applicatifs et implanté une notion inédite dans le domaine des réseaux rapides : la qualité de service. À cette fin, nous avons étudié, dans l'optique d'une adaptation au contexte des réseaux rapides, plusieurs des mécanismes issus des travaux sur la qualité de service dans les réseaux classiques et mis en œuvre une nouvelle stratégie d'ordonnancement au cœur du moteur d'optimisation de la bibliothèque NewMadeleine. Celle-ci manipule les différents flux en provenance de la couche applicative pour les diriger vers des politiques d'ordonnancement adaptées à leurs besoins (latence, réactivité, etc.)

L'évaluation des multiples algorithmes d'ordonnancement implémentés a permis de valider l'efficacité de notre solution, notamment lorsque des flux aux priorités différentes partagent un même lien de communication.

6.2 Alternatives

Une autre voie à adopter en ce qui concerne l'implémentation de notre solution aurait pu être l'écriture complète d'un ordonnanceur au sein de NewMadeleine, ce qui est parfaitement réalisable car cette bibliothèque de communication permet de définir et d'utiliser des ordronnanceurs interchangeables. Il ne s'agit plus de l'écriture d'une stratégie supplémentaire appartenant à l'actuel ordonnanceur de la bibliothèque. La gestion de la qualité de service aurait été intrinsèquement liée à la mécanique de NewMadeleine. Cependant, le positionnement des stratégies dans l'architecture de NewMadeleine nous autorise tous les réordonnancements et optimisations nécessaires aux algorithmes de garantie de qualité de service. De plus les résultats auraient été moins puissants, l'ordonnanceur actuel de NewMadeleine intègre diverses optimisations dans d'autres parties extérieures au code des stratégies.

Par ailleurs, nous avons présentés quelques algorithmes de qualité de service dans la section 2.6.2. D'autres algorithmes extrêmement performants auraient pu venir se rajouter au panel actuel, tels que PGPS (Packets Generalized Processor Sharing) qui dessert les paquets selon l'ordre croissant de leur date virtuelle de fin d'émission. Cet algorithme permet un partage réellement équitable de la bande passante mais nécessite que le processus qui l'exécute ait un contrôle total de la bande passante disponible, ce qui n'est pas notre cas. En outre, les temps d'exécution à disposition sont totalement différents (quelques μs pour les réseaux rapides contre plusieurs dizaines de ms pour les réesaux classiques.

6.3 Perspectives

Ces travaux sur Stratos ouvrent de nombreuses perspectives à explorer. À court terme, il s'agit de terminer la mise en œuvre de la gestion de la réactivité pour effectuer les tests décrits en section 4.3 et ainsi valider une politique supplémentaire à offrir aux flux applicatifs.

Dans un avenir moins proche, il serait intéressant d'étendre la gestion globale du mixage des flux au moment où une carte réseau devient disponible. Pour le moment, la stratégie est de fournir un paquet provenant d'une politique d'ordonnancement à chaque fois que la carte en fait la demande. L'alternance se réalise par un tourniquet sur les politiques actives à ce moment-là. Une idée serait de tenter d'agréger les flux en provenance des différentes politiques si cela est possible, c'est-à-dire en accord avec les contraintes propre à chacune de ces politiques. L'utilisation du lien de communication en serait maximisé.

Enfin, à plus long terme, le fonctionnement des politiques d'ordonnancement pourrait être modifié pour dresser un ordre de priorité englobant la totalité des files des différentes politiques. En effet actuellement les flux de différentes politiques possèdent toutes la même priorité de service : par exemple, un flux qui nécessite une latence élevée aura autant de priorité qu'un flux de données critiques, ce qui n'est pas forcément le comportement que l'on désire obtenir au niveau applicatif.

Bibliographie

[1] Projet lego. http ://graal.ens-lyon.fr/LEGO/.

[2] Infiniband Trade Association. Infiniband technology overview, 2006. http ://www.infinibandta.org/.

[3] Olivier Aumage. *Madeleine : une interface de communication performante et portable pour exploiter les interconnexions hétérogènes de grappes.* Thèse de doctorat, spécialité informatique, École normale supérieure de Lyon, 46, allée d'Italie, 69364 Lyon cedex 07, France, September 2002. 154 pages.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service, 1998.

[5] Elisabeth Brunet. Newmadeleine : ordonnancement et optimisation de schémas de communication haute performance (version étendue de perpi'06). *Technique et Science Informatiques*, 2007. Submitted.

[6] Octavio Napoleón Medina Carvajal. *Étude des algorithmes d'attribution de priorités dans un Internet à différenciation de services.* Thèse de doctorat, Université de Rennes I, March 2001.

[7] Aurélien Esnard. *Analyse, conception et réalisation d'un environnement pour le pilotage et la visualisation en ligne de simulations numériques parallèles.* Informatique, Université de Bordeaux 1, décembre 2005.

[8] Thierry Dauxois et Michel Peyrard. La dernière expérience d'enrico fermi. *La Recherche*, May 2005. N°386.

[9] IEEE. Standard for scalable coherent interface (sci), August 1993. Standard IEEE numéro 1596.

[10] N. Karonis, B. Toonen, and I. Foster. Mpich-g2 : A grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing (JPDC)*, May 2003. Vol. 63, No. 5, pp. 551-563.

[11] John L.Hennessy and David A.Patterson. *Computer Architecture, A Quantitative Approach.* Morgan Kaufmann editions, 1990. 4ème édition.

[12] Emmanuel Lochin. *Qualité de Service dans l'Internet. Garantie de débit TCP dans la classe AF.* Thèse de doctorat, Université de Paris VI, December 2004.

[13] Guillaume Mercier. *Communications à hautes performances portables en environnements hiérarchiques, hétérogènes et dynamiques.* Thèse de doctorat, Université de Bordeaux 1, Labri, Bordeaux, France, December 2004.

[14] Myricom. Myricom website. http ://www.myri.com/.

[15] Inc. Myricom. Myrinet express (mx) : A high performance, low-level, message-passing-interface for myrinet, 2006.

[16] Raymond Namyst. *Contribution à la conception de supports exécutifs multithreads performants*. Habilitation à diriger des recherches, Université Claude Bernard de Lyon, pour des travaux effectués à l'école normale supérieure de Lyon, December 2001.

[17] Scott Pakin, Vijay Karamcheti, and Andrew A. Chien. Fast Messages : Efficient, portable communication for workstation clusters and MPPs. *IEEE Concurrency*, 5(2) :60–73, / 1997.

[18] Quadrics Ltd. *Elan Programming Manual*, apr 2005. http ://www.quadrics.com.

[19] K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C. Lee, and H. Casanova. GridRPC : A Remote Procedure Call API for Grid Computing, june 2002.

[20] Top500 SuperComputer Sites. 28th top500 list released, November 2006. http ://www.top500.org/.

[21] Leila Toumi. *Algorithmes et mécanismes pour la qualité de service dans des réseaux hétérogènes*. Thèse de doctorat, Institut National Polytechnique de Grenoble, December 2002.

[22] François Trahay. Gestion de la réactivité des communications réseau. Mémoire de dea, Université Bordeaux 1, June 2006.

[23] Élisabeth Brunet. Support d'ordonnancement et d'optimisation automatisés des communications pour les réseaux hautes performances. Research Report 5641, INRIA, July 2005.