RenPar'17 / SympA'2006 / CFSE'5 / JC'2006
Canet en Roussillon, 4 au 6 octobre 2006

NEWMADELEINE : ordonnancement et optimisation de schémas de communication haute performance

Elisabeth Brunet

Projet INRIA RUNTIME, LaBRI
Université Bordeaux I - 351, cours de la Libération - 33405 Talence Cedex

---

Résumé
Malgré les progrès spectaculaires accomplis par les interfaces de communication pour réseaux rapides ces quinze dernières années, de nombreuses optimisations potentielles échappent encore aux bibliothèques de communication. La faute en revient principalement à une conception focalisée sur la réduction à l'extrême du chemin critique afin de minimiser la latence. Dans cet article, nous présentons une nouvelle architecture de bibliothèque de communication bâtie autour d'un puissant moteur d'optimisation des transferts dont l'activité s'accorde avec celle des cartes réseau. Le code des stratégies d'optimisations est générique et portable, et il est paramétré à l'exécution par les capacités des pilotes réseau sous-jacents. La base de données des stratégies d'optimisation prédéfinies est facilement extensible.

Mots-clés : Communications Hautes Performances, Ordonnancement, Optimisation.

---

1. Introduction

Dans le paysage du calcul parallèle, la percée des machines que sont les grappes de PC repose évidemment beaucoup sur leur avantageux rapport performance/coût, mais aussi sur les progrès réalisés en matière de réseaux d'interconnexion *rapides*. Et dans ce domaine, les évolutions du matériel (latences très faibles associées à des débits élevés) ont été accompagnées par des progrès logiciels remarquables : les recherches menées depuis une quinzaine d'années en matière de bibliothèques de communication ont permis, en associant un vaste éventail de techniques telles que les communications en mode utilisateur, les échanges zéro-copie ou le déport de certaines opérations sur les cartes réseaux, de réduire au minimum le chemin critique des traitements réalisés lors d'une communication. Depuis quelques années, le surcoût logiciel des bibliothèques spécialisées (Elan/Quadrics [5] ou encore MX/Myrinet [6]) se réduit à quelques centaines de cycles-processeur par transaction. Mieux encore, les implémentations récentes du standard MPI ([2], [4]), qui sont constituées pour une grande part d'appels directs aux routines de l'interface sous-jacente, affichent quasiment les mêmes performances.

Aujourd'hui, s'il n'est plus vraiment pertinent de chercher à diminuer le surcoût logiciel occasionné par ces bibliothèques lors des transferts unitaires de données, il reste néanmoins de nombreux défis à relever en terme d'optimisation des transferts dès lors que les schémas de communications deviennent complexes (messages multi-segments avec dépendances) ou que différents flots de communications indépendants partagent concurremment les mêmes cartes réseau. Dans cet article, nous montrons pourquoi les interfaces de communication actuelles sont limitées sur ce plan et nous proposons une nouvelle architecture de bibliothèque de communication capable d'appliquer des optimisations agressives sur la globalité des flux de communication utilisés les applications.

2. De nouvelles opportunités d'optimisation sur réseaux rapides

Les besoins des applications de calcul hautes performances ne se réduisent pas toujours à minimiser la latence des échanges de données pris de manière individuelle. En effet, privilégier la bande passante ou encore le recouvrement des temps de communication peut s'avérer judicieux pour des applications utilisant un système de stockage de données à distance ou encore des applications ayant une forte charge de calcul à exécuter. Nous avons montré par le passé ([1]) que les applications et environnements de programmation mettant en œuvre des protocoles ou des mécanismes de communication de haut niveau (mémoire virtuellement partagée, appels de procédure à distance, etc.) peuvent tirer un grand bénéfice d'une interface de communication plus puissante que MPI. Le manque d'expressivité de cette dernière interface, même si elle se comporte de manière optimale sur les incontournables tests de type *ping-pong*, ne permet pas d'exprimer des indications subtiles telles que les dépendances entre les différents fragments (service, arguments, objet concerné) d'une invocation de méthode distante. C'est pourtant en permettant l'expression et en exploitant l'information de ces contraintes qu'un support de communication peut exploiter les capacités du réseau de manière optimale : il peut alors choisir d'accumuler des paquets afin d'utiliser des fonctionnalités de collection/dissémination (*gather*/*scatter*) ou *agréger* plusieurs petites requêtes en une seule plus grosse si les contraintes sont suffisamment relachées pour le permettre, *réordonner des paquets*, ou encore favoriser l'acheminement prioritaire de fragments importants (comme un numéro de service RPC, nécessaire pour préparer de façon anticipée les zones de mémoire destinées à recevoir les arguments du service). Pour ces classes d'applications caractérisées par des schémas de communications complexes et irréguliers, il ne s'agit donc pas de faire une simple traduction des opérations de communication en appels à des routines du pilote réseau, mais bien d'*interpréter*, *réorganiser* et *optimiser* le flux d'opérations de communication de façon dynamique.

Par ailleurs, l'évolution des techniques de programmation vers des applications modulaires et composites, résultant de l'intégration de divers environnements de programmation (cf *PadicoTM* [3]) et souvent multiprogrammées pour exploiter les architectures multiprocesseurs modernes (multicores), engendrent des besoins applicatifs nouveaux : cette évolution amène une multiplication des flux de communication logiques entre les différents modules de deux processus communiquant. Les capacités de multiplexage physique des cartes réseau — très réduites en raison de la consommation de ressources occasionnée dans la mémoire embarquée sur les cartes — ne sont pas prévues pour répondre à cette demande (seulement 3 unités de multiplexage par défaut pour une carte de communication Myrinet avec MX, par exemple). Il est donc nécessaire de « mixer » les multiples flux logiques sur les ressources de multiplexage physiques. En utilisant conjointement un multiplexage logiciel des flux et les techniques d'optimisation évoquées précedemment (agrégation, réordonnancement) de façon globale à l'ensemble des flux, il serait possible d'appliquer de façon circonstancielle des optimisations purement opportunistes, à condition que la bibliothèque de communication soit capable de gérer des flux de données non-déterministes du côté du récepteur. C'est ce que nous détaillons dans la section suivante.

3. Le moteur de communication NEWMADELEINE

Afin de permettre l'implantation de telles optimisations de manière performante, nous proposons une nouvelle architecture de communication, nommée NEWMADELEINE[1], dont les caractéristiques principales sont détaillées ci-après.

3.1. Constitution d'une fenetre de travail

Les optimisations dynamiques multi-paquets et multi-flux mentionnées précédemment nécessitent par définition une *fenêtre* de travail de plusieurs paquets pour être réalisables. Or, le fonctionnement tradi-

[1] Le nom NEWMADELEINE traduit le fait que l'interface de programmation est héritée du logiciel MADELEINE.

tionnellement *synchrone* des supports de communication — où les opérations de communications sont corrélées à l'activité de l'application — se prête mal à ce mode opératoire : soit un paquet est immédiatement transmis sur le réseau lorsqu'il est soumis par l'application et il n'y a pas d'accumulation (c'est le fonctionnement habituel), soit le support de communication devrait garder arbitrairement le paquet jusqu'à ce que l'application envoie un ou plusieurs autres paquets afin de constituer la fenêtre, ce qui n'est pas réaliste du point de vue de la latence induite et qui n'est de toute façon pas acceptable d'un point de vue algorithmique en raison des risques de deadlock. Afin de permettre la constitution effective de la fenêtre recherchée sans subir de tels désagréments, il est nécessaire d'abandonner la corrélation entre l'application et le traitement des opérations de communication. Dans ce cas, les opérations doivent être traités suivant l'activité de la carte (ou des cartes) de communication. Tant que celle-ci est occupée, on accumule une fenêtre de paquets, et quand elle devient inoccupée, on analyse la fenêtre de paquets collectés entre-temps, et on en extrait une requête que l'on soumet au réseau pour redonner du travail à la carte. La constitution de la fenêtre est donc naturelle, et ne présente pas les inconvénients algorithmiques de l'approche synchrone.

3.2. Stratégies, tactiques et sélection d'optimisation

La fenêtre de travail étant constituée, se pose maintenant la question du travail d'optimisation proprement dit. Considérons l'état des unités de multiplexages physiques à un instant donné. Si au moins une unité de multiplexage est disponible à cet instant, il faut lui donner du travail à réaliser en appliquant une *fonction d'optimisation* chargée de sélectionner (ou de générer, par exemple, par une agrégation) le prochain paquet à soumettre à chacune des unités inactives en considérant les paquets de la fenêtre de travail. En entrée de la fonction d'optimisation, nous avons une grande variété d'arguments potentiels, dont voici une liste non exhaustive : le nombre de paquets dans la fenêtre, leurs caractéristiques respectives (destination, flux, longueur, numéro de séquence, attributs de dépendance), les caractéristiques nominales et fonctionnelles du réseau, éventuellement des indications de l'application sur la politique d'ordonnancement des paquets, ainsi que le nombre d'unités de multiplexages physiques disponibles et leur état d'activité à l'instant considéré.

Devant un tel nombre de paramètres, plusieurs tactiques d'optimisations sont possibles, mais la combinaison optimale de ces tactiques reste un problème difficile. Aussi, plutôt que de proposer une heuristique d'optimisation figée, nous proposons au contraire que la fonction d'optimisation soit sélectionnable (à terme dynamiquement) parmi un ensemble de stratégies extensible et programmable : chaque *stratégie* met en œuvre l'application d'un certain nombre de *tactiques*, c'est-à-dire d'opérations élémentaires d'optimisation du panel d'opérations usuelles, dans la réalisation d'un objectif d'optimisation particulier.

Si aucune unité de multiplexage n'est disponible, le cas est simple : on remet le travail d'optimisation à une date ultérieure. Une autre possibilité envisageable serait de préparer un unique paquet à émettre pour anticiper sur la prochaine libération d'une des unités, et pouvoir la réalimenter immédiatement (donc en travaillant avec un coup d'avance). Une troisième alternative serait d'appliquer malgré tout la fonction d'optimisation si la quantité de paquets accumulés atteint un certain seuil.

3.3. Architecture

L'architecture de NEWMADELEINE est organisée en trois couches : une couche de collecte des données auprès des applications, une couche d'ordonnancement et d'optimisation et une couche de transfert pilotant les cartes de communication (voir la figure 1).

La couche de transfert mime les cartes réseaux à la façon d'un ordonnanceur de processus qui, lorsqu'il est invoqué par un processeur, déroule un algorithme permettant de choisir un nouveau processus prêt. Son rôle est de surveiller l'état d'avancement des cartes, et de dialoguer avec la partie supérieure pour obtenir de nouveaux paquets optimisés dès que cette dernière achève sa tâche. Pour cela, une partie générique interroge successivement les pilotes spécifiques à chaque réseau, uniques détenteurs des pri-

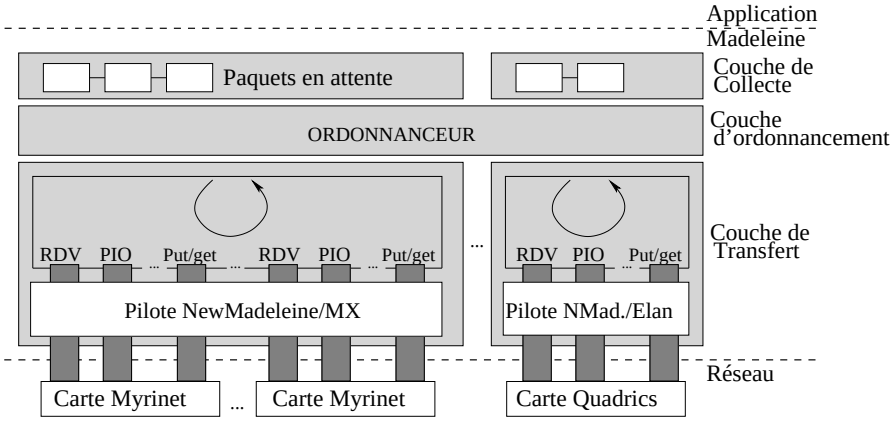


FIG. 1 – Architecture de NewMadeleine

mitives permettant l'interrogation de la carte elle-même. C'est une couche très fine sans intelligence : les primitives des pilotes se résumant à une primitive d'envoi, de réception et de scrutation.

La couche d'ordonnancement-optimisation est la partie intermédiaire. Les paquets déposés par l'application sont tirés par les cartes réseau en temps voulu par son intermédiaire. Elle est alors chargée de manipuler la liste pour former un paquet et fournir un assemblage à la fois conforme aux contraintes applicatives et avantageux du point de vue des performances. Ce paquet est alors directement transmis à la carte demandeuse pour émission immédiate. Pour ne pas faire patienter la carte, on pourrait préparer quelques paquets en avance. Cependant, il est préférable de se limiter à un nombre restreint de tels paquets car cela laisse à l'ordonnanceur une marge de manœuvre plus importante qui ne peut être que bénéfique. Finalement, la carte ne pouvant consommer qu'un paquet à la fois, un paquet en cours et un paquet d'avance est une mesure adaptée.

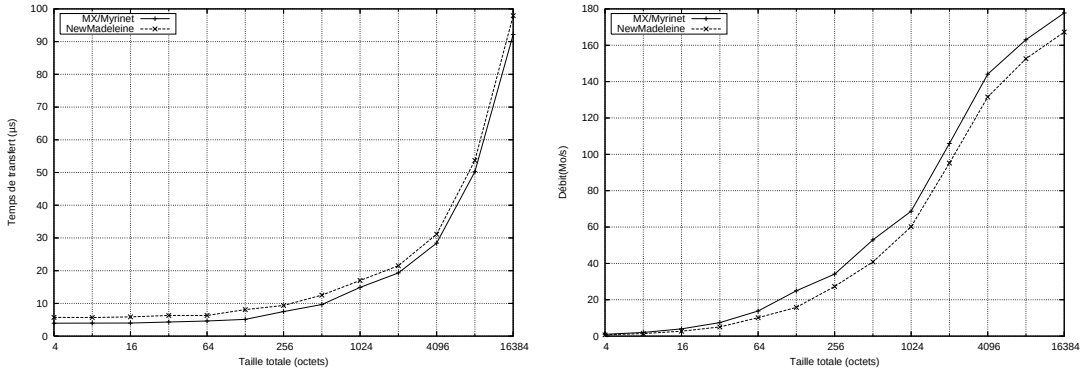
Cette couche met également en œuvre les protocoles réseaux basiques nécessaires au bon déroulement de communication (tels que le contrôle de flux, les prises de rendez-vous, etc.) mais assure aussi la remontée du succès des requêtes à l'application concernée.

La couche de collecte est chargée de recenser les données déposées par les différents flux de communication ainsi que les meta-données nécessaires à leur identification par le récepteur (i.e. numéro de canal, identifiant de l'expéditeur, numéro de séquence). Une fois encapsulées et afin de permettre un équilibrage de charge entre les différentes cartes présentes, les données collectées sont alors insérées sur une liste dédiée à la technologie réseau désignée par l'application ou, à défaut, sur une liste générale pour un équilibrage de charge sur l'ensemble des cartes de technologie même différente en présence.

4. Implantation et évaluation

Un prototype de **NewMadeleine** muni d'un ordonnanceur statique est d'ores et déjà opérationnel sur GM/MYRINET, MX/MYRINET, ELAN/QUADRICS, SISCI/SCI et TCP/ETHERNET et le sera sous peu sur INFINIBAND. Pour le moment, une unique stratégie est prise en compte qui consiste à agréger les paquets tant que la taille cumulée ne requiert pas un envoi via un rendez-vous préalable. Cependant, le développement de stratégies d'optimisation étant à la fois délicat et commun à de nombreuses technologies réseaux, nous avons développé une bibliothèque qui permettra par la suite d'écrire des stratégies d'ordonnancement indépendamment du protocole réseau sous-jacent, afin d'enrichir l'ensemble des stratégies en faisant bénéficier automatiquement toutes les technologies réseau et d'évaluer et comparer en temps réel les stratégies entre elles pour déterminer la meilleure à un instant donné.

De plus, afin de pouvoir intégrer facilement de nouvelles technologies réseaux, nous avons défini une in-



(a) Latence. (b) Débit.

FIG. 2 – Ping-pong - NEWMADELEINE vs MX.

terface commune aux différents pilotes : fonctions d'initialisation, de fermeture, d'envoi, de réception et de scrutation. Ces fonctions sont de très bas niveau puisque leur corps ne se restreint pratiquement qu'à l'appel direct de la primitive de l'interface de communication bas niveau associée. Afin que l'ordonnanceur exploite au mieux une technologie réseau donnée, les spécificités de celle-ci sont répertoriées : capacités de collection/dissémination (*gather*/*scatter*), seuils critiques des différentes méthodes de transfert, capacités d'accès à la mémoire distante (*RDMA*), etc.

4.1. Evaluation

Nous avons réalisé trois types d'expérience : un ping-pong "classique" afin de déterminer le surcoût logiciel induit par NEWMADELEINE, l'utilisation d'une *pseudo*-mémoire virtuellement partagée (MVP) afin de mesurer l'apport du réordonnancement des paquets d'un même flux de communication et l'utilisation de ping-pongs parallèles pour mesurer l'intérêt du multiplexage de flux concurrents. L'ensemble des tests ont été réalisés au dessus de la bibliothèque de communication MX/MYRINET à l'aide d'une grappe de bi-Xeon 2,6 GHz sous LINUX 2.6, chaque noeud étant équipé d'une carte MYRINET 2000.

Surcoût brut par message élémentaire. Nous commençons par mesurer à l'aide d'un ping-pong les performances de NEWMADELEINE en les comparant à celles de la bibliothèque de communication MX (figure 2(a) et 2(b)). Nous observons un surcoût logiciel constant de moins de 2 μs de la part de notre bibliothèque. Deux facteurs sont en cause. Tout d'abord, les données sont systématiquement accompagnées d'entêtes afin de permettre les inversions et le multiplexage. Ainsi, à taille de données utiles égales, la masse (entêtes + données utiles) des données effectivement transmise est supérieure avec NEWMADELEINE. Ensuite, l'application d'optimisations sur une liste d'un élément unique ne peut être que superflue par rapport à un envoi direct. Nous pensons que ce surcoût pourra être atténué dans la prochaine implémentation qui pourra notamment simplifier les mécanismes invoqués dans de telles situations.

Demandes de pages auprès d'une MVP. Un algorithme schématique de demandes de pages auprès d'une MVP est décrit à la figure 3. Le schéma de communication produit par une interface de communication ne faisant pas de réordonnancement est présenté à la figure 4(a). Chaque demande d'envoi (traduite par un appel à la fonction *pack* (voir la figure 3)) se solde par l'envoi effectif demandé et symétriquement en réception (via des appels à la fonction *unpack*). A la sortie de l'appel, la requête est terminée du point de vue du programmeur. Les paquets n'ont pourtant pas de dépendances bloquantes entre eux à l'exception de la réception notée (7) du côté client qui dépend de la réception précédente (6) dans la figure 3 et cela au sein d'une seule et même demande. Dans notre proposition, nous donnons à l'application la possibilité d'exprimer ces dépendances afin d'autoriser des inversions et de mieux ordonnancer les paquets en exploitant le réseau sous-jacent. Ainsi, si on opte pour une tactique agrégeant

Du côté du client,

```
int nb_pages_a_demander = random();
int numero_page;
bool trouve;

for(int i = 0; i < nb_pages_a_demander; i++){
  numero_page = random();

(1) pack(destination, mon_id, sizeof(int));
(2) pack(destination, numero_page, sizeof(int));
(3) pack(destination, envoi_diff, sizeof(bool));
(4) pack(destination, type_acces, sizeof(int));

(5) unpack(&numero_page, sizeof(int));
(6) unpack(&trouve, sizeof(bool));
  if(trouve){
(7)   unpack(&page, taille_page);
  } else {
    // recherche de page sur un autre noeud
  }
}
```

Du côté du serveur,

```
int source, numero_page;
bool envoi_diff;
int type_acces;

while(1){

(1) unpack(&source, sizeof(int));
(2) unpack(&numero_page, sizeof(int));
(3) unpack(&envoi_diff, sizeof(bool));
(4) unpack(&type_acces, sizeof(int));

 page = recherche_page(numero_page, type_acces);

(5) pack(source, numero_page, sizeof(int));
  if(page){
(6)   pack(source, touve, sizeof(bool));
(7)   pack(source, page, taille_page);
  } else
(8)   pack(source, pas_trouve, sizeof(bool));
}
```

FIG. 3 – Code d'une série de demande de pages auprès d'une MVP.

les paquets jusqu'à un certain seuil, on pourra observer de nouveaux schémas de communications tels que ceux décrits dans les figures 4 (b) et 4 (c). Les capacités même du réseau sous-jacent à, entre autres, gérer ou non les techniques du collection/dissémination sont un point déterminant dans la façon dont l'ordonnancement va se dérouler.

Nous présentons dans le tableau 5 une moyenne des résultats obtenus pour la demande et la réception de 3 pages de 65 Ko via MX/MYRINET et en utilisant la fonctionnalité de *gather/scatter*. La taille des pages a été choisie de manière à ce que leur transfert ne soit pas fait de la même manière que la requête elle-même (on force le passage par un rendez-vous). Le test ne porte que sur un nombre réduit d'échanges et l'écart entre les résultats est déjà important. Ceci est particulièrement intéressant pour toutes les applications basées sur les schémas de communication évoluant différemment suivant des contextes donnés, typiquement des schémas basés sur le paradigme de communication des appels de procédure à distance. En effet, ces schémas non prédictibles ne pouvant être pré-cablés par un programmeur averti au moment même du développement de l'application sont dépendants d'un ordonnancement s'accommodant de la situation courante.

Ping-pongs parallèles. A présent que nous sommes en mesure d'ordonnancer les paquets d'un même flux, nous considérons l'ensemble des paquets des flux de communication concurrents afin de créer des opportunités d'optimisation plus nombreuses. Pour cela, nous exécutons en parallèle deux ping-pongs dont le nombre de paquets par *ping* est variable pour accroître le nombre de paquets à agréger et ainsi marquer plus distinctement les effets du multiplexage. Les courbes de la figure 6(a) présentent le temps d'envoi moyen pour un paquet parmi des flux de communication à 10 éléments chacun. Nous avons ensuite jouer sur le nombre de paquets par ping (voir figure 6(b)). Comme on peut le constater, la stratégie d'agglomération agressive des paquets amène un gain sensible qui s'accroît lorsque le nombre de paquets agglomérés augmente. Cette optimisation est particulièrement adaptée aux applications qui s'échangent très fréquemment des messages courts comme des messages de contrôle ou de synchronisation autour de plusieurs flux de communication.

5. Travaux apparentés

Nombre de travaux de recherche ont été menés dans le but d'exploiter au mieux les réseaux rapides.

MPICH-NEMESIS[2] est à notre connaissance la meilleure implémentation actuelle de MPI en terme de

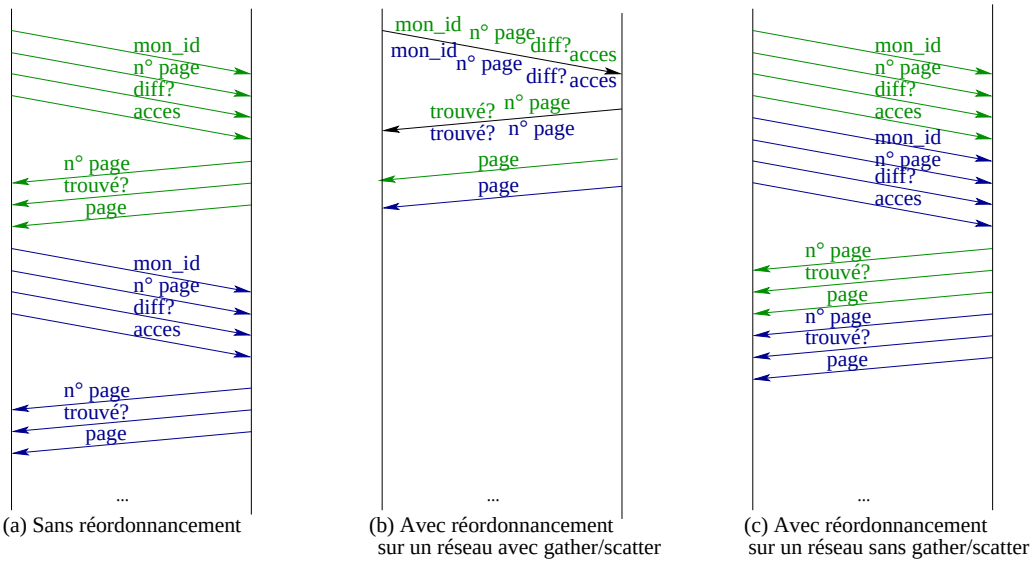
latence et de débit. Cependant, ordonnancer les communications ni même les multiplexer n'est en accord avec leur orientation latence. MPICH n'a pas les mêmes classes d'application cibles.
PM VERSION 2[8] est une bibliothèque de communication de bas niveau. Elle permet l'exécution d'une même application sur différents réseaux. Cependant, PM2 est concue pour être la base de YAMPII, une implémentation de MPI produite par la même équipe, également orientée latence.
VMI 2[7] est une bibliothèque de communication bas-niveau mais orientée tolérance aux pannes. Ainsi, elle est capable de faire de l'équilibrage de charge sur des réseaux homogènes et hétérogènes. Elle ne multiplexe pas les flux de communication mais fait de l'ordonnancement entre les cartes réseau, ce qui une politique d'ordonnancement orienté débit.
MADELEINE 3 [1] est la version antérieure de de la bibliothèque NEWMADELEINE. Elle proposait déjà à travers son interface enrichie des moyens d'agglomérer des messages n'ayant pas d'inter-dépendances. Cependant, le réordonnancement impliquant des inversions de paquets n'est pas envisageable car les données transitent sous leur forme brute. De plus, le multiplexage n'y est pas géré nativement.

6. Conclusion

Si l'exploitation brute des réseaux rapides est à présent parfaitement maîtrisée par des bibliothèques de communication dotées d'implémentations hyper-optimisées, de nombreux gains de performance sont encore possibles si l'on considère les flux de communications entre paires de processus dans leur globalité.
Dans cet article, nous présentons une nouvelle architecture de communication pour réseaux rapides nommée NEWMADELEINE. Sa principale originalité réside dans son moteur d'optimisation des transferts de données qui, tout en exploitation les informations fournies par l'interface (dépendances entre segments, contraintes temporelles), permet d'appliquer dynamiquement des stratégies d'optimisation génériques. La bibliothèque est multithreadée, ce qui lui permet de déclencher les optimisations *just-in-time*, en fonction de l'activité des cartes réseau. Entre chaque paire de processus, les optimisations ont une portée globale aux flux de données, quel que soit le nombre de canaux de communications utilisés, ce qui permet des optimisations opportunistes et aggressives.
Les évaluations de l'implémentation préliminaire sur MX/Myrinet montrent que les bénéfices de l'approche sont sensibles sur des exemples rencontrés dans des applications réelles. Dans un futur proche, des évaluations plus vastes seront menées au sein de l'environnement PadicoTM [3], qui promet d'être un terrain d'optimisations potentielles très vaste en matière de multiplexage des communications. Par ailleurs, nous comptons enrichir la bibliothèque de manière à augmenter les stratégies d'optimisation disponibles et automatiser le choix de la meilleure stratégie à appliquer.

Bibliographie

1. O. AUMAGE, L. BOUGÉ, A. DENIS, L. EYRAUD, J.-F. MÉHAUT, G. MERCIER, R. NAMYST et L. PRYLLI. « A Portable and Efficient Communication Library for High-Performance Cluster Computing ». *Cluster Computing*, 5(1) :43–54, 2002.
2. D. BUNTINAS, G. MERCIER et W. GROPP. « Data Transfers Between Processes in a SMP System : Performance Study and Application to MPI ». Dans *International Conference on Parallel Processing*, 2006.
3. A. DENIS, C. PÉREZ et T. PRIOL. « PadicoTM : An Open Integration Framework for Communication Middleware and Runtimes ». *Future Generation Computer Systems*, 19(4) :575–585, 2003.
4. Richard L. GRAHAM, Timothy S. WOODALL et Jeffrey M. SQUYRES. « Open MPI : A Flexible High Performance MPI ». Dans *Proceedings, 6th Annual International Conference on Parallel Processing and Applied Mathematics*, 2005.
5. Quadrics LTD.. « Elan Programming Manual », 2003. http ://www.quadrics.com/.
6. Inc. MYRICOM. « Myrinet EXpress (MX) : A High Performance, Low-level, Message-Passing Interface for Myrinet ». 2003.
7. Scott PAKIN et Avneesh PANT. « VMI 2.0 : A Dynamically Reconfigurable Messaging Layer for Availability, Usability, and Management ». Dans *8th International Symposium on High Performance Computer Architecture (HPCA-8)*, 2002.
8. T. TAKAHASHI, S. SUMIMOTO, A. HORI, H. HARADA et Y. ISHIKAWA. « PM2 : High Performance Communication Middleware for Heterogeneous Network Environments ». Dans *SC'00*, pages 52–53, 2000.



(a) Sans réordonnancement

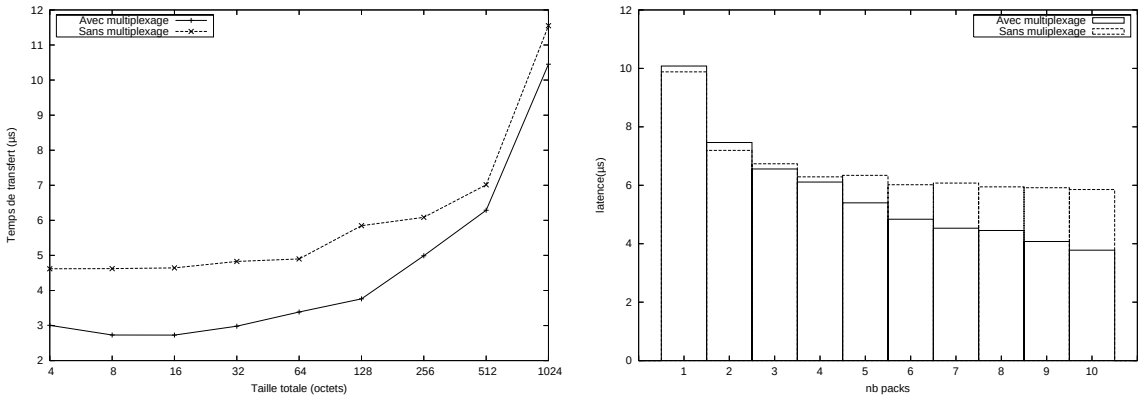
(b) Avec réordonnancement sur un réseau avec gather/scatter

(c) Avec réordonnancement sur un réseau sans gather/scatter

FIG. 4 – Schémas de communication de la demande de 2 pages auprès d'une MVP

	Sans ordonnancement	Avec ordonnancement
Latence	88 μs	61 μs

FIG. 5 – Résultats obtenus pour la demande et la réception de 3 pages de 65 Ko via MX/MYRINET



(a) Temps moyen de transfert d'un paquet au sein de 2 flux de communication de 10 éléments chacun

(b) Temps moyen d'envoi d'un paquet de 128 octets alors que le nombre d'éléments par flux varie

FIG. 6 – Multiplexage de 2 flux de n packs chacun. La masse effective transférée est de 2*n*taille considérée.