Noyau de communication sécurisé pour la machine parallèle MPC

Alexandre Fenyö[a], Pierre David[b] et Alain Greiner[a]

[a]Laboratoire LIP6 / Département ASIM / UPMC
4 place Jussieu, F-75252 Paris Cedex 05, France

[b]Laboratoire PRiSM, UVSQ
45 avenue des Etats-Unis, F-78035 Versailles Cedex, France

Résumé
La machine parallèle MPC est composée, au niveau matériel, de nœuds processeurs de type PC/Pentium, interconnectés par un réseau rapide gigabit développé au sein de l'équipe ASIM du laboratoire LIP6. Pour utiliser au mieux le matériel, qui fournit une fonction d'écriture distante très efficace, de nouveaux protocoles de communication sécurisés ont été développés et mis en œuvre sous Unix FreeBSD.

Mots-clés : machine parallèle, noyau de communication, protocole sécurisé, faute matérielle

---

1. Introduction

La machine MPC est une machine parallèle conçue au laboratoire LIP6 de l'Université Pierre et Marie Curie (UPMC), en collaboration avec le laboratoire PRiSM de l'Université de Versailles/St Quentin en Yvelines (UVSQ) et l'Ecole Nationale Supérieure des Télécommunications (`http://mpc.lip6.fr`). Les composantes matérielles de la machine MPC sont décrites dans [1]. Son système d'exploitation est basé sur FreeBSD (Unix dérivé de BSD-4.4), qui a été étendu avec des modules noyau à chargement dynamique, mettant en œuvre les protocoles de communication, et des démons externes se chargeant de la configuration et du contrôle du système global. Pour permettre aux applications de tirer bénéfice de la faible latence matérielle du réseau ($< 5\ \mu s$ de mémoire à mémoire), trois niveaux de protocoles de type «zéro-copie» ont été mis en œuvre dans le noyau Unix et sont décrits ci-après (voir figure 1). Les fonctions permettant de les activer sont non bloquantes et disponibles au sein du noyau. Elles permettent, pour la plupart, à l'application appelante de passer en argument une fonction de signalisation à exécuter en fin de traitement. Cela permet d'obtenir simplement des appels systèmes bloquants et non bloquants, suivant l'utilisation que l'on fait de la signalisation.

2. Écriture distante

Le réseau d'interconnexion HSL (High Speed Link) de la machine MPC fournit une primitive de communication optimisée : l'écriture distante. Le contrôleur de réseau se comporte comme un contrôleur DMA où la lecture des données est effectuée par le contrôleur de réseau du nœud local, et où l'écriture des données est effectuée par le contrôleur de réseau du nœud distant. Les données sont découpées en paquets par le matériel et celui-ci effectue un comptage en réception pour signaler la fin de la transmission par une interruption, si tous les paquets ont été correctement reçus. Du point de vue logiciel, l'écriture distante est fournie par la couche PUT, qui est l'API de plus bas niveau. L'application utilisatrice doit s'inscrire auprès de cette couche. L'application fournit pour cela une fonction de signalisation asynchrone, et se fait attribuer un ensemble d'*identificateurs de messages* (entiers sur 24 bits) associés à cette fonction. Dès lors, l'application peut poster une requête DMA non bloquante en fournissant un identificateur de message, une zone de mémoire physique contiguë ainsi qu'une zone de mémoire sur le nœud distant, spécifiées par des adresses physiques, et PUT se charge de recopier la zone locale sur le nœud destinataire. Lorsque les données sont entièrement rentrées dans le réseau, PUT l'indique par un appel à la fonction de signalisation asynchrone sur le nœud local. De même, une fois les données correctement déposées, PUT appelle la fonction de signalisation sur le nœud distant. Ces fonctions reçoivent en paramètre l'identificateur de message, ce qui permet à l'application utilisatrice de déterminer les données concernées. Selon le même principe, la couche PUT permet de transporter des messages courts (8 octets), sans préciser de zone de mémoire locale ou distante : les données sont directement fournies en émission comme paramètre de la fonction PUT, et en réception en paramètre de la fonction de signalisation.

La couche PUT constitue la brique de base fondamentale pour construire la pile de protocoles de la machine MPC. Néanmoins, elle nécessite des zones contiguës et verrouillées en mémoire physique pour contenir les données de l'application. Pour obtenir de telles zones, nous avons conçu un module noyau (CMEM), à l'aide du gestionnaire de mémoire virtuelle de FreeBSD, initialement développé pour le système Mach [2]. Ce module, chargé à l'initialisation du système, réserve une zone physique contiguë de grande taille, et en réattribue des parties aux utilisateurs de PUT.

3. Passage de messages en adresses physiques

La couche SCP/P (Secure Channelized Protocol / Physical addresses), bâtie sur PUT, fournit deux primitives non bloquantes, *Send* et *Receive*. Plutôt que de préciser des adresses distantes, l'émetteur (fonction *Send*) doit simplement indiquer un canal sur lequel les données sont transportées en séquence. Les données doivent être verrouillées mais il n'est pas nécessaire qu'elles soient contiguës. Dans chaque nœud, des boîtes aux lettres fixes, attribuées par CMEM, sont mises à disposition des autres nœuds pour accueillir la localisation physique des tampons de réception. Un identificateur de message est associé à chaque boîte aux lettre. L'émetteur utilise ce même identificateur pour marquer les données émises. Lors de la conclusion de l'échange, une fonction de signalisation, fournie en paramètre des fonctions *Send* et *Receive*, est appelée. Deux démons (`mpcclient/mpcserver`) sur chaque nœud, dialoguant via RPC à travers le réseau de contrôle[1], se chargent, à l'initialisation du système, d'informer les autres nœuds de la localisation des boîtes aux lettres qui leur sont attribuées.

4. Tolérance aux fautes du réseau

Le taux de pannes du réseau HSL est très faible (comparable à celui d'un bus partagé). Néanmoins, nous avons souhaité sécuriser le protocole SCP/P, tout en gardant sa caractéristique «zéro-copie», ce qui pose des problèmes peu communs :

- Les données erronées, à la différence des protocoles classiques, peuvent être déposées dans la zone de réception, car l'automate de DMA matériel [3] calcule les codes CRC au fur et à mesure du dépôt des données, et c'est seulement lorsque le transfert est terminé qu'il indique éventuellement une erreur. Pour s'adapter à cette situation, le protocole MICP (Message Identifier Cleaner Protocol) vient s'intercaler, en cas de faute, entre les différentes émissions/réceptions et permet ainsi de gérer la validité des données locales.

- L'opération *Send* n'émet les données sur le réseau que si l'opération *Receive* a été effectuée sur le nœud distant pour indiquer la localisation des tampons de réception. Le récepteur ne peut donc pas distinguer l'absence de *Send*, de la perte d'un message *Receive*. Pour résoudre ce problème, la sous-couche SFCP (Sender Flow Control Protocol), sous l'impulsion de l'émetteur, transmet régulièrement au récepteur le numéro de séquence courant sur le canal. Le procédé est renouvelé jusqu'à ce qu'un acquittement soit reçu.

- Dans un protocole sécurisé, on ne peut pas se contenter de prévenir l'utilisateur de *Send* lorsque les données sont rentrées dans le réseau. Le protocole RFCP (Receiver Flow Control Protocol), activé cette fois par le récepteur, permet la libération des données chez l'émetteur. Pour cela le récepteur lui indique le nombre de *Send* correctement reçus.

- Lorsque des données sont perdues ou considérées comme telles après l'expiration d'un délai de garde, le récepteur doit envoyer à nouveau des informations à destination des boîtes aux lettres de l'émetteur. Il lui faut, pour ce faire, associer à cette écriture distante l'identificateur de la boîte aux lettre choisie. Mais si l'écriture précédente vers cette même boîte aux lettre a subi une *perte partielle*, c'est-à-dire que des paquets ont été corrompus ou perdus mais que d'autres ont été correctement acheminés, le comptage des paquets associés à l'identificateur de message en question

[1] Le réseau de contrôle est un réseau Ethernet bas débit.

va se trouver erroné, ce qui risque de provoquer prématurément l'interruption de fin de transfert. La couche MICP permet de contourner ce problème en réinitialisant les compteurs de paquets du côté émetteur comme du côté récepteur.

Les opérations *Send* et *Receive* sont regroupées dans la sous-couche BSCP (Basic SCP). Les sous-couches MICP, RFCP et SFCP utilisent des messages courts, qui sont transmis en un seul paquet sur le réseau. Cela permet de s'affranchir des problèmes de perte partielle de données, de la gestion des identificateurs de messages (un seul identificateur utilisé par chacune de ces sous-couches), et de la gestion de boîtes aux lettres (les données sont transportées hors-bande, il n'y a pas d'adresse locale ni distante à fournir).

Le coût de la sécurisation du protocole SCP, en l'absence d'erreur, est très faible : le protocole MICP n'est alors pas mis en œuvre, et la bande passante occupée par les messages courts de SFCP et RFCP est négligeable. L'unique impact sur les performances se traduit par un délai supplémentaire avant la signalisation de fin de transaction chez l'émetteur comme chez le récepteur. Ce délai correspond au temps d'aller-retour d'un message court sur le réseau.

Notons que le problème de la tolérance aux fautes du réseau est résolu grâce à l'existence des messages courts, transportés en un seul paquet.

5. Passage de messages en adresses virtuelles

La couche SCP/V (Secure Channelized Protocol / Virtual addresses) s'appuie sur SCP/P afin de fournir un service d'échange sur des canaux, les zones de mémoire étant cette fois-ci désignées par des adresses virtuelles dans l'espace du noyau ou d'un processus. Aux deux primitives *Send* et *Receive* de cette couche, on fait correspondre les deux primitives apparentées de la couche inférieure, en ayant au préalable effectué un découpage en zones physiques contiguës à l'aide de la sous-couche V2P (Virtual To Physical). Dans le cas d'une réception dans une zone de mémoire d'un processus, on vérifie aussi les droits d'accès en écriture de la zone. Dans un souci d'efficacité, les opérations de déréférenciation virtuelle/physique et de vérification des droits d'accès sont mises en œuvre directement par consultation des tables de la MMU du processeur, plutôt que par un parcours des tables des pages du système de mémoire virtuelle Mach.

6. Sécurité et intégrité du système

Avant l'appel à la primitive *Send* ou *Receive*, les données doivent être verrouillées. Si les données font partie de l'espace en mémoire d'un processus, celui-ci peut les déverrouiller par erreur (appel système `munlock()`) en cours de transfert. Du côté émetteur, on risque ainsi d'envoyer le contenu de pages physiques d'autres processus chez le récepteur, ce qui pose le problème de la confidentialité des données. Du côté récepteur, on risque de voir des données venir se déposer dans des zones allouées entre temps à d'autres processus, voire même au noyau, et on se trouve donc également confronté au problème de l'intégrité du système. On pourrait imaginer interdire l'appel système `munlock()` pendant les transferts, mais la fin prématurée du processus repose les mêmes problèmes. La solution, mise en œuvre au sein de la sous-couche VMR (Virtual Memory Referencer), consiste à créer et gérer de façon adéquate une carte de la mémoire au sein du système de mémoire virtuelle de Mach, du même type que celles qui décrivent les espaces en mémoire des processus et du noyau. Cette carte n'est associée à aucun processus. Dès qu'une opération *Send* ou *Receive* intervient dans un quelconque processus, on recherche un espace libre dans la carte de la mémoire pour y insérer des entrées référençant entièrement les objets de la mémoire Mach correspondants (le plus souvent il s'agit d'une zone du tas ou de la pile). On simule alors des fautes de page sur ces entrées de carte de la mémoire puis on les verrouille. On a alors la garantie que même si le processus vient à les déverrouiller, les références de notre carte de la mémoire empêcheront les données de réellement quitter la mémoire physique, par exemple pour retourner dans la zone de pagination sur disque.

Plutôt que de libérer la carte de la mémoire à chaque fin de transaction, une opération périodique de *ramasse-miettes* se charge de la mettre à jour. Le coût de la couche VMR est le suivant : il faut 6 μs sur un Pentium 200 pour vérifier qu'une zone en mémoire de 4 Ko est déjà référencée dans la carte de VMR, alors qu'il faut plus de 200 μs pour l'y insérer.

La figure 1 présente l'empilement des couches de protocoles, depuis le pilote de périphérique, jusqu'à la couche de plus haut niveau adaptée aux opérations en adresses virtuelles. La figure 3 présente les messages en jeu dans un échange *Send/Receive* sans faute (MICP n'intervient donc pas).

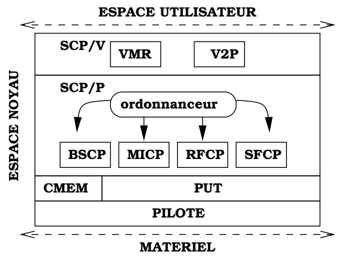

Figure 1 : empilement des couches SCP

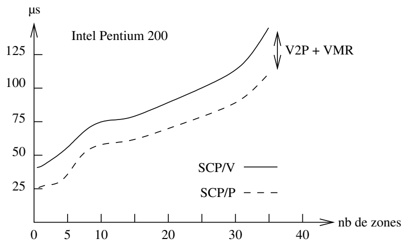

Figure 2 : délai de traversée des couches

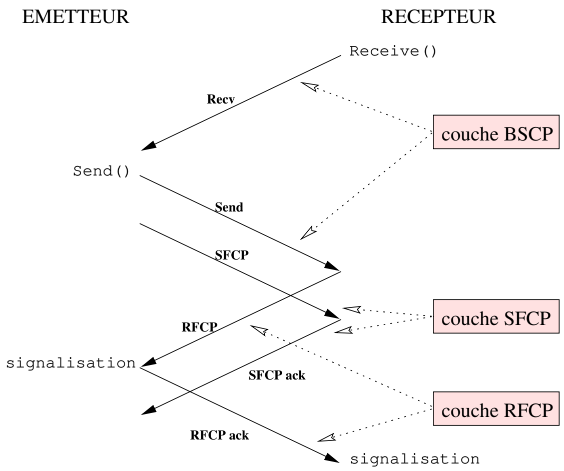

Figure 3 : synopsis d'un échange sans faute

7. Mesures de performances

La figure 2 indique la latence de traversée des couches logicielles pour effectuer un appel aux opérations *Send* de la couche physique et de la couche virtuelle. Il s'agit du temps mesuré entre l'appel et le moment où l'automate de DMA local a reçu les ordres d'émission. On remarquera que le coût logiciel ne dépend pas de la taille des données mais du nombre de zones physiques non contiguës à transférer.

8. Conclusion

Le noyau de communication de la machine MPC a permis d'expérimenter, sur l'émulateur de la machine, un certain nombre d'applications, dont le simulateur logique PI-SIM sur PUT, le système de pagination distribué MAÎS [4] sur SCP/P et l'implémentation optimisée de PVM (F. Silva, I. Demeure) sur SCP/V. Une machine MPC à 4 nœuds est en cours d'installation au LIP6. Elle permettra d'analyser les performances que SCP saura tirer du réseau gigabit pour des applications réelles et de les comparer aux performances obtenues avec d'autres bibliothèques bas-niveau comme les Messages Actifs [5].

Bibliographie

1. P. David, J.L. Desbarbieux, A. Fenyö, A. Greiner, J.J. Lecler, F. Potter, V. Reibaldi, F. Wajsbürt et B. Zerrouk – La machine MPC – Réseaux à haut débit de stations pour le support d'applications parallèles et réparties. Calculateurs Parallèles, Réseaux et Systèmes Répartis, 1998, (accepté).
2. R. Rashid, A. Tevanian, Jr., M. Young, D. Golub, R. Baron, D. Black, W. Bolosky, et J. Chew – Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architectures – Proceedings of the 2nd Symposium on Architectural Support for Programming Languages and Operating Systems, ACM, October, 1987.
3. F. Wajsbürt, J.L. Desbarbieux, C. Spasevski, S. Penain et A. Greiner – An Integrated PCI component for IEEE 1355 Networks – Advanced in information technologies : The business challenge. pages 844-850, Florence, Italie, IOS Press, 1998. Edited by J.-Y Roger et al, Novembre 1997.
4. P. Cadinot, N. Dorta et B. Folliot – MAÎS: un système de pagination en mémoire distante dans un environnement réseau à haut débit – RenPar'97 - 9ièmes Rencontres Francophones du Parallélisme.
5. S. Lumetta, A. Mainwaring et D. Culler – Multi-Protocol Active Messages on a Cluster of SMP's – Proceedings of SC97.