RenPar'9,
Lausanne. Suisse, 20–23 mai 1997.

MAÎS : un système de pagination en mémoire distante dans un environnement réseau à haut débit

Philippe Cadinot[a], Neilze Dorta[a b], et Bertil Folliot[a]
E-mail : Philippe.Cadinot@masi.ibp.fr, Neilze.Dorta@inria.fr, Bertil.Folliot@masi.ibp.fr

[a]LIP6-MASI, Université Paris 6

[b]INRIA Rocquencourt, projet SOR

Mots Clé : pagination répartie, réseau à haut débit, `remote write`.

1. Introduction

Les réseaux à haut débit (Ethernet 100 Base T, ATM, FDDI) permettent d'accéder à des données plus rapidement lorsqu'elles sont situées dans la mémoire centrale d'une machine distante inutilisé que lorsqu'elles sont situées sur un disque local. La modification de la traditionnelle pyramide des temps d'accès incite à reconcevoir certains mécanismes de systèmes d'exploitation de manière à substituer les accès à un disque local vers une mémoire distante.

Notre objectif consiste à développer un support d'exécution efficace pour des applications nécessitant de grandes quantités de mémoire. Le principe consiste à diminuer le coût du mécanisme de pagination en tirant partie de la vitesse de transfert de mémoire à mémoire tout en augmentant l'espace mémoire disponible mais sans modifier la configuration physique existante. Les applications considérées sont des outils parallèles dédiés de CAO-VLSI. Ces outils accèdent à de très nombreuses structures de données (par exemple, la vérification à plat de règles de CAO). La mise à disposition de l'ensemble des mémoires des stations d'un réseau, permet non seulement d'améliorer les performances du mécanisme de pagination mais également d'exécuter de nouvelles gammes d'applications ayant des grands besoins en mémoire.

Les travaux existants de la pagination distante sur disque et en mémoire centrale, ainsi que la conception de MAÎS se trouvent dans [10]. La section 2 décrit la réalisation et la section 3 présente les modifications apportées au système d'exploitation. Nous concluons sur les premières mesures de performances obtenues et donnons les perspectives de ces travaux en section 4.

2. Réalisation de MAÎS

La mise en œuvre de MAÎS repose sur le modèle client/serveur. Les serveurs de pages sont identifiés comme étant des machines inutilisées du réseau. Les processus clients, souhaitant éviter de décharger des pages sur leur disque local utilisent la mémoire centrale des machines serveurs de pages. Cette désignation est actuellement faite de façon statique par l'administrateur du système : les machines sont désignées clientes ou serveurs et ne changent pas de rôle au cours d'une exécution. Nous prévoyons `n` serveurs et `k` clients, où les machines peuvent devenir cliente ou serveur en fonction de l'évolution des activités utilisateurs. Nous ne considérons pas pour l'instant le cas du partage de pages entre plusieurs processus, car il suppose un modèle de mémoire virtuelle partagée répartie [7], en dehors de la problématique de la pagination répartie.

Fondamentalement, MAÎS cherche à tirer profit de la vitesse du réseau et des mémoires inutilisées des autres stations de travail, sans se soucier de minimiser le coût total des références mémoires [5]. C'est pourquoi, contrairement au travail de Levy [5] qui a une politique de gestion de mémoire global pour toutes les stations de travail, MAÎS gère le système de pagination dans chaque machine séparément.

2.1. Difficultés

La réalisation d'un mécanisme de pagination répartie est complexe puisqu'elle nécessite de résoudre les points suivants:

- modification du paginateur et des structures de données du noyau pour lui permettre d'identifier les pages pouvant être situées à distance, ainsi que localiser les machines détentrices de pages;
- création d'une interface de communication de noyau/noyau permettant la création d'espace de mémoire utilisable à distance ainsi que l'envoi et la récupération des pages [11];
- identification des machines inutilisées qui pourront jouer le rôle de serveur de page [9, 1];
- analyse du grain de la taille d'une page de manière à optimiser les transferts.

Par la suite nous décrivons la solution que nous apportons aux deux premiers problèmes.

2.2. Support de Réalisation

Notre plate-forme de test est composée de 5 Pentium 150 Mhz interconnectés par un concentrateur Ethernet 100 Base T (100 Mbit/s). Le système retenu est une version publique, modifiable et documentée du système UNIX : le système FreeBSD 2.2 (version mars 96). Ce système dispose d'un mécanisme de pagination en mode utilisateur, ce qui permet de limiter les modifications à apporter au noyau. Cette plate-forme constitue un modèle de ce que l'on peut trouver dans de nombreux réseaux d'entreprises. Nous y avons ajouté un émulateur du protocole d'écriture distante (nommé DDSBP : `Direct Deposit Based Protocol`) [4].

Par la suite, nous nous placerons dans le cadre du projet MPC [3, 4] du laboratoire MASI, qui vise à la conception d'une machine multi-ordinateurs. Elle est composée d'un grand nombre de processus standards (Pentium) reliés par un réseau à haut débit. Ce réseau, le `High Speed Link` (HSL) [6], utilise le routeur `Rcube` développé au laboratoire MASI. Il a pour caractéristique un très grand débit (640Mo/s) et un temps de latence faible (<10 micro-secondes). Il est associé à un protocole matériel d'écriture directe en mémoire distante (`DDSBP`). Ce réseau assure des transferts de messages en quelques dizaines de micro-secondes, à comparer aux 10 milli-secondes d'un réseau Ethernet 10 Base T (10Mbit/s) standard.

3. Modifications apportées au système d'exploitation

FreeBSD 2.2 offre une interface de pagination constituée de trois types de paginateurs : `vnode pager, device pager, swap pager`. Le dernier traite des objets qui "mappent" de la mémoire anonyme (non persistante) et c'est à ce niveau que la pagination à distance se situe. La réalisation de MAÎS consiste à réécrire un paginateur du type `swap` qui remplace le traitement effectué lors d'un défaut de page pour des objets non persistants, de manière sélective. Un processus candidat à la pagination répartie situé sur une machine cliente se fait connaître par un appel au driver MAÎS. Cela permet d'assurer l'intégrité du système, de sorte à ne pas paginer à distance les processus qui ne le souhaitent pas (processus systèmes).

Les structures de données utilisées pour gérer la Mémoire Virtuelle dans FreeBSD 2.2 sont : `vmspace, vmmap, vm_map_entry, object, vm_page`. Un `object` est une structure indépendante qui décrit la source (fichier, périphérique, mémoire anonyme) de la donnée pour un intervalle d'adresses. Puisqu'il n'y a pas de lien entre un objet et un processus [8], un paginateur est référencé pour chaque objet pour gérer la pagination dans un périphérique physique. Cela est réalisé en deux étapes :

- tout d'abord en modifiant l'indicateur d'état de l'objet (appel à la fonction `mais_client`) ,
- ensuite, lors du premier `pageout`, le paginateur MAÎS est alloué.

Les paginateurs sont désignés par des entrées dans la table des `pagers` : `pagerops[]`. L'ajout d'un nouveau paginateur a donc exigé la modification de la structure de gestion des paginateurs du système FreeBSD en ajoutant des entrées supplémentaires à la table `pagerops[]` (nous avons prévue la possibilité d'avoir plusieurs paginateurs, adaptable à plusieurs types de processus). Chaque entrée de la table `pagerops[]` pointe vers une structure `pagerops` qui contient des pointeurs vers des fonctions propres à chaque paginateur. Malheureusement, certaines fonctions du `swap pager` (allouer le `pager`, désallouer le `pager`...) n'appartiennent pas à la structure `pagerops`, alors que ce fonctions sont nécessaires à des paginateurs de type MAÎS. Nous avons donc généralisées l'accès à ces fonctions en les incluants dans la structure `pagerops`. Les structures utilisées pour désigner l'espace de `swap` ont été modifiées de manière à identifier la machine sur laquelle se trouve la page. Normalement une page référence un objet qui référence un certain nombre de numéros de blocs disque : page $\rightarrow$ objet $\rightarrow$ pager $\rightarrow$ n^o. de bloc.
La modification consiste alors à associer au numéro de bloc le numéro du nœud sur lequel le bloc se trouve. Ce numéro de nœud correspond à l'adresse de la machine sur le réseau.

3.1. Interface de communication noyau/noyau du serveur MAÎS

MAÎS est composé d'un driver et d'un processus démon. Le driver contient le code et les données du démon, du serveur et du client.

Le processus démon gère l'insertion et le retrait des nœuds ainsi que le passage du mode client au mode serveur. En mode serveur le processus démon fournit des quota de pages aux nœuds clients et en mode client il maintient les quota de pages et effectue la pagination sur les serveurs de pages.

Le serveur de page a pour fonction d'allouer un espace mémoire, de diviser cet espace en pages et de servir les demandes des clients MAÎS en provenance des processus démons. Chaque serveur de page conserve également l'identité des propriétaires de pages.

Le client gère son quota de pages par des fonctions `get_swap_space` et `free_swap_space`. Il effectue les lectures et les écritures à distance dans les serveurs par des fonctions `get_pages` et `put_pages`.

3.2. Réservation d'un quota de pages

De manière à réagir rapidement aux demandes de déchargement de pages, les clients conservent un quota de pages (un `pool` contenant une liste d'adresses physiques) fourni par les machines identifiées comme serveurs. Après le transfert d'un ensemble de pages sur le serveur, celui-ci peut renouveler le `pool` du client en lui fournissant un nouveau quota. Lors d'un défaut de page, FreeBSD demande l'anticipation de `x` pages au paginateur. Le paginateur MAÎS utilise une politique basée sur l'historique des adresses déjà demandées pour décider le sens (croissante, décroissante ou une moyenne) de l'anticipation.

Le choix d'un serveur de page est résolu par une politique circulaire, agissant ainsi de façon préventive quant à la répartition de la charge [2].

4. Conclusion

Nous avons décrit la réalisation de MAÎS, un mécanisme de pagination répartie en mémoire, dédié à des applications ayant des besoins en mémoire dépassant les capacités d'une seule machine. MAÎS a été conçus avec pour objectif de minimiser les modifications du noyau. Un autre aspect intéressant de MAÎS est un réel degré d'indépendance au système. Dans ce but, seuls le driver et le paginateur MAÎS sont dépendants du système FreeBSD.

Ce travail s'inscrit dans le cadre du projet de la machine parallèle MPC [3, 4]. Le tableau suivant indique: le temps de pagination standard (en écriture et en lecture), le temps du serveur MAÎS (réseau HSL + latence système), les speed up obtenu ainsi que leur valeur maximale.

Nb.	`Page In`					`Page Out`				
de	Std.	MAÎS(ms)		Speed Up		Std.	MAÎS(ms)		Speed Up	
page	(ms)	HSL	Système	MAÎS	Max	(ms)	HSL	Système	MAÎS	Max
1	12,2	0,113	0,35	26,35	108	-	0,12	0,46	-	-
10X1	24,3	1,13	3,5	5,25	21,5	13,4	1,2	4,6	2,3	11,2
10X16	117,5	18	57	1,57	6,5	135,4	19,2	39,5	2,3	7,05

Les résultats obtenus dans cette toute première implémentation sont encourageants (gains de 1,6 à 26). Ils représentent de 24 à 32 % des valeurs maximales possibles. Les cas les plus courants sont l'évincement d'un bloc de page (gain de 2,3), et le chargement d'une série de pages individuelles (gains de 5,35 et 26). Le but est de continuer le développement vers une version totalement dynamique et tolérante aux fautes. L'idée consiste à intégrer la tolérance aux fautes directement dans le gestionnaire de swap réparti au moyen d'un mécanisme de réplication des pages, ou par calcul de parité. Cette solution est la plus efficace mais est la plus complexe à mettre en œuvre, c'est à terme celle que nous comptons implanter.

Bibliographie

1. B.Bretelle, T.Terracol, B.Folliot, and P.Sens. Problématique du placement dynamique dans les systèmes répartis haut-débit. (Bretelle,Terracol,Folliot,Sens)@masi.ibp.fr, 1996.
2. B.Folliot. *Méthodes et Outils de partage de charge pour la conception et la mise en oeuvre d'applications dans les systèmes répartis hétérogènes.* PhD thesis, Université Pierre et Marie Curie, Paris VI, 1992.
3. C.A.Thekkat. System support for efficient network communication. Tech. rep., Departement of Computer Science and Engineering, University of Washington, Seattle, July 1994.
4. F.Potter, P.David, and A.Fenyö. Interface logicielle pour la machine mpc. Tech. rep., Laboratoire MASI - PRiSM , Université Pierre et Marie Curie - Université de Versailles, 1996.
5. H.M.Levy, M.J.Feeley, Morgan, F.Pighin, A.Karlin, and C.A.Thekkath. Implementing global memory management in a workstation cluster. Tech. Rep. TR-95-08-07, Dept. of Computer Science - University of Washington, Aug. 1995.
6. J.Lecler, F.Potter, J.L.Desbarbieux, F.Wajsbrt, and A.Greiner. Pci_ddc / spécifications (rapport). Tech. rep., Laboratoire MASI, Institut Blaise Pascal. Université Pierre et Marie Curie, Apr. 1996.
7. K.Li, and P.Hudak. Memory coherence in shared virtual memory systems. *ACM Transactions on Computer Systems 7*, 4 (1989), 321–359.
8. M.K.McKusic, K.Bostic, M.J.Karels, and J.S.Quarterman. *The Design and Implementation of the 4.4BSD Operating System.* Addison-Wesley, 1996.
9. M.M.Theimer, and K.A.Lantz. Finding idle machines in a workstation-based distributed system. In *IEEE Trans. on Software Engineering* (Nov. 1989), no. 15 in (11), IEEE.
10. P.Cadinot, N.Dorta, B.Folliot, and P.Sens. Mécanisme de va-et-vient réparti en mémoire dans un environnement réseau haut débit. *MPR6 - Journées de Recherche sur la Mémoire répartie Partagée* (May 1996), 143–150.
11. V.Trecordi. Inter-process communication performance in high-speed networks. In *International Conference on Communications* (1994), vol. 1, IEEE, pp. 57–63.