

REPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE

UNIVERSITE DE DSCHANG

ECOLE DOCTORALE



REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND

UNIVERSITY OF DSCHANG

POST GRADUATE SCHOOL

DSCHANG SCHOOL OF SCIENCES AND TECHNOLOGY
Unité de Recherche en Informatique Fondamentale, Ingénierie et Application (URIFIA)

Distributed shared memory model

Présenté par :
TCHIO AMOUGOU Styves daudet

*Matricule : CM-UDS-14SCI0251
Licencié en Informatique Fondamentale*

Sous la direction de
Dr BOMGNI ALAIN Bertrand
(Chargé de Cours, Université de Dschang)



Sommaire

1 *Case 1 :*

2 *Case 2 :*



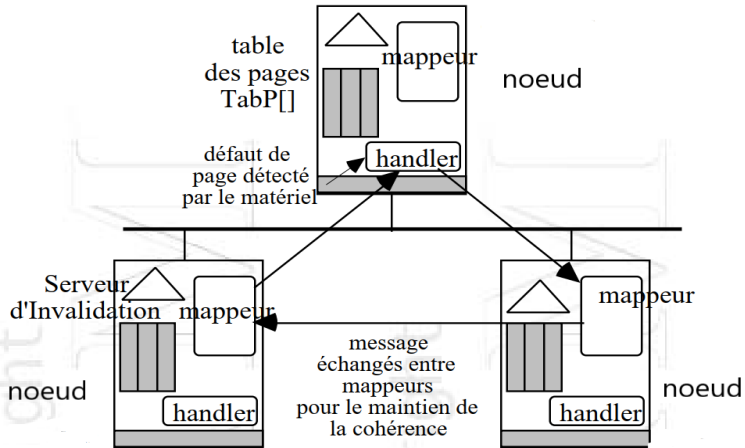


FIGURE – 1



Modèle de distribution de mémoire

- La mémoire fonctionne sur le principe d'une mémoire paginée.
- Lors d'un défaut de page, le processeur s'adresse à un **handler** qui recherche la page demandée auprès des autres processeurs du réseau.
- Le **mappeur** est la partie du système attachée à un processeur qui répond aux demandes de pages provenant des autres processeurs du réseau.
- Les **mappeurs** communiquent entre eux pour maintenir la cohérence entre les différentes copies des pages.
- Les communications sont fiables : pas de perte de messages, pas de duplication de messages, pas de corruption de messages, pas de déséquencelement des messages, ou alors, le protocole de transport est capable de le détecter et de le corriger.



Les informations que maintiennent les mappeurs sur les pages sont contenues dans une table `TabP[]` , table des pages qui contient les champs suivants :

- **droits d'accès** : représente le mode d'accès à la page : read, write, nil,
- **copyset** : sites avec une copie contient l'ensemble des noeuds qui ont une copie (en lecture) de la page,
- **lock** : verrou : joue le rôle de variable sémaphore pour l'accès en exclusion mutuelle à la page pendant un défaut, deux primitives sont utilisées :
 - ▶ `LOCK()` pour tester le sémaphore et éventuellement bloquer le demandeur,
 - ▶ `UNLOCK()` pour déverrouiller l'accès,
- **probOwner** indique un propriétaire estimé (probable) de la page.
- **Numéro Pages** correspond au numéro de page Exactement comme dans les mémoires paginées



Table des Pages

N° page	verrou	{sites avec une copie}	propr probable	droits d'accès

FIGURE – Table de pages



- Dans la table de pages d'un noeud, se trouve tout les pages de la mémoire virtuel de son voisinage.
- les noeuds qui non pas de problème de mémoire dans le réseau participent aussi a la gestion de la mémoire il seront juste équipés d'un handler et d'un mappeur. leurs tables de pages seront vide.
- comme la mémoire virtuel est classe par ordre de grandeur décroissante de mémoire restante quand un noeud a déjà INi il cherche juste dans sa table de pages la première pages donc le copyset correspond au Noeud INi et met a jour le probOwner des pages qu'il veut occuper.



Exemple

Pour le Noeud 3 est en manque de mémoire. $L_3 = 9; 10$
 $M_{m3} = 1T, M_{r9} = 3T, M_{r10} = 6T$.

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

FIGURE – Table de pages



Tables de pages |

Numero de pages	Lock	Copyset	Prop Owner	access
1				
2				
3				
4				
5				
6				
7				
7				
8				
9				

FIGURE – Table de pages initiale



Tables de pages |

Numero de pages	Lock	Copyset	Prop Owner	access
1			3 <u> </u>	
2				
3				
4				
5				
6				
7				
7				
8				
9				

FIGURE – Table de pages après calcul



Exemple

Le Noeud 4 est en manque de mémoire. $L_4 = 1, 2, 5, 6$
 $M_{m4} = 1T, M_{r5} = 7T, M_{r1} = M_{r2} = 1T$.

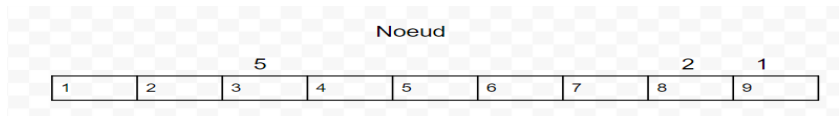


FIGURE – Segmentation de la mémoire virtuel en page pour le noeud 4

Le Noeud 4 est en conflit avec le noeud 6 et les deux noeuds n'ont aucun noeud voisin en commun.
 le Noeud 4 va occuper 2T sur le noeud 5.



Tables de pages |

Numero de pages	Lock	Copyset	Prop Owner	access
1				
2				
3				
4				
5				
6				
7				
7				
8				
9				

FIGURE – Table de pages initiale



Tables de pages

5	Numero de pages	Lock	Copyset	Prop Owner	access
	1			4	
	2			4	
	3				
	4				
2	5				
	6				
	7				
	7				
	8				
1	9				

FIGURE – Table de pages après calcul



Exemple

Pour le noeud 6, $L_6 = 4, 7, 8$, $L_7 = 5, 6, 8, 11$

Le noeud 6 et 7 sont en conflit et ont un voisin en commun qui est le noeud 8.

$Ind_{s7} = 11$ et $Ind_{s6} = 8$ donc le noeud 6 occupe 2T sur le noeud 8.

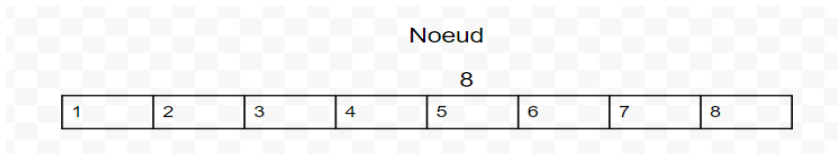


FIGURE – Segmentation de la mémoire virtuel en page pour le noeud 6



Tables de pages

Numero de pages	Lock	Copysset	Prop Owner	access
1				
2				
3				
4				
5				
6				
7				
7				
8				

FIGURE – Table de pages initiale



Tables de pages

Numero de pages	Lock	Copyset	Prop Owner	access
1			6	
2			6	
3				
4				
5				
6				
7				
7				
8				

FIGURE – Table de pages après calcul



Exemple

Pour le noeud 7, le noeud 7 occupe 2T sur le noeud 11.



FIGURE – Segmentation de la mémoire virtuel en page pour le noeud 7



- Dans la table de pages d'un noeud, se trouve tout les pages de la mémoire virtuel de son voisinage.
- les noeuds participent au calcul.
- chaque noeud indique a ses voisins ou il doivent stocker leur liste de voisin
- comme la mémoire virtuel est classe par ordre de grandeur décroissante de mémoire restante quand un noeud a déjà INi il cherche juste dans sa table de pages la première pages donc le copyset correspond au Noeud INi et met a jour le probOwner des pages qu'il veut occuper.



REPUBLIQUE DU CAMEROUN
PAIX-TRAVAIL-PATRIE

UNIVERSITE DE DSCHANG

ECOLE DOCTORALE



REPUBLIC OF CAMEROON
PEACE-WORK-FATHERLAND

UNIVERSITY OF DSCHANG

POST GRADUATE SCHOOL




DSCHANG SCHOOL OF SCIENCES AND TECHNOLOGY
Unité de Recherche en Informatique Fondamentale, Ingénierie et Application (URIFIA)

Présenté par :
TCHIO AMOUGOU Styves daudet

Matricule : CM-UDS-14SCI0251
Licencié en Informatique Fondamentale

Sous la direction de
Dr BOMGNI ALAIN Bertrand
(Chargé de Cours, Université de Dschang)



-  [1].Diksha Verma, Anjali Tyagi, Deepak Sharma. *page based distributed shared memory*. Reading, 2014.
-  [2]. Changhun Lee. *Distributed Shared Memory*. Proceedings on the 15th Cisl Winter Workshop Kushu, Japan, February 2002.
-  [3].Kai Li, Paul Hudak. *Memory Coherence in Shared Virtual Memory Systems*. ACM TOCS, V7, N4, November 1989.

