Réalisation d'une Mémoire Partagée Répartie

Nombre d'étudiants par projet : 2-3
Langage : C sous Linux
Contact : Pierre Sens
Email : Pierre.Sens@lip6.fr

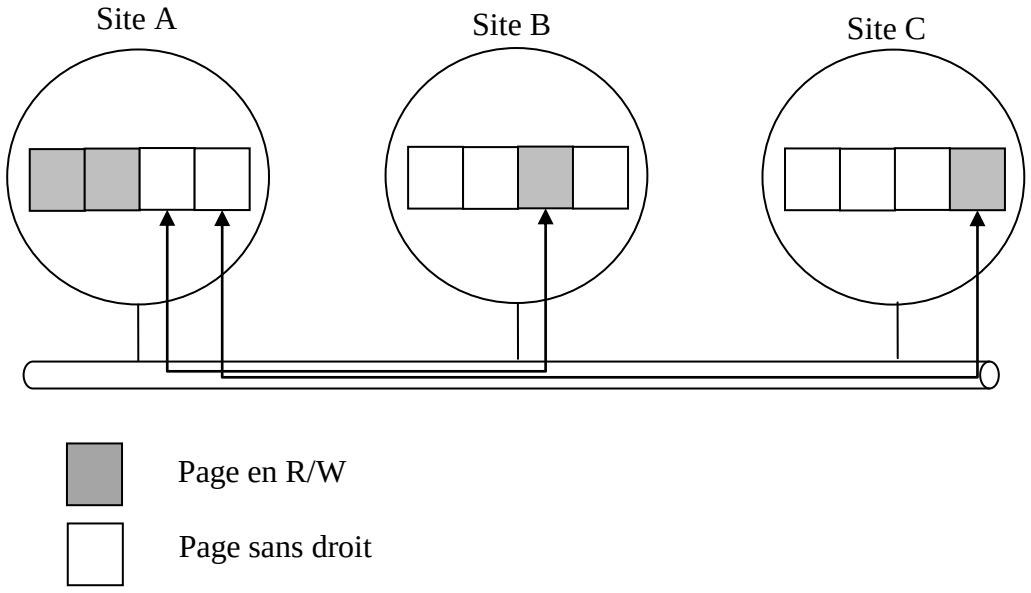
Sujet :

Les mémoires partagée réparties permettent de simuler une zone de mémoire partagée pour des processus s'exécutant sur des *machines différentes* (qui ne partagent donc pas physiquement de mémoire).

Les processus pourront lire et modifier la mémoire partagée. Les modifications devront être répercutées à tous les sites utilisant cette zone de mémoire. Ce type de mécanisme permet de développer plus facilement des applications réparties.

De algorithmes de maintien de la cohérence devront assurer que chaque donnée lue est à jour.

Le segment est découpé en pages. Initialement, un seul processus a le droit de lire ou modifier le segment. Si un autre processus veut accéder à une donnée du segment, la page correspondante devra lui être envoyé via le réseau. Ainsi une page pourra être répliqué sur plusieurs site. Dès qu'un processus modifie une page, il faut *invalider ou mettre à jour* toutes les copies de la page sur les autres sites.



Mise en oeuvre :

La mise de ce projet reposera sur les sockets ainsi que sur l'utilisation des primitives de gestion mémoire *mmap* et *mprotect* permettant de gérer les droits d'accès aux différentes pages.

Un serveur centralisé (le maître) regroupera l'ensemble des informations sur les pages (en autre qui possède la dernière version d'une page, la liste des sites possédant une copie ...). La site maître possède initialement toutes les données partagées avec les droits en lecture/écriture. Les autres sites (les esclaves), n'ont pas les droits d'accès à la mémoire partagée. A leur premier accès, les esclaves recupérent du maître une copie la page accédée.

Le projet devra implemter sous forme de bibliothèque l'interface suivante :

```
void *InitMaster();
```

Initialisation du maître, retourne l'adresse de la mémoire partagée répartie

```
void *InitSlave(char *HostMaster);
```

Initialisation d'un esclave, retourne l'adresse de la mémoire partagée répartie

```
void lock-read(void *adr);
```

Demande un verrou pour l'accès en lecture à l'adresse adr. Si la page correspondante en verrouillé en écriture par un autre site cette primitive est bloquante

```
void lock-write(void *adr);
```

Demande un verrou pour l'accès en écriture à l'adresse adr. Si la page correspondante en verrouillé en lecture ou en écriture par un autre site cette primitive est bloquante