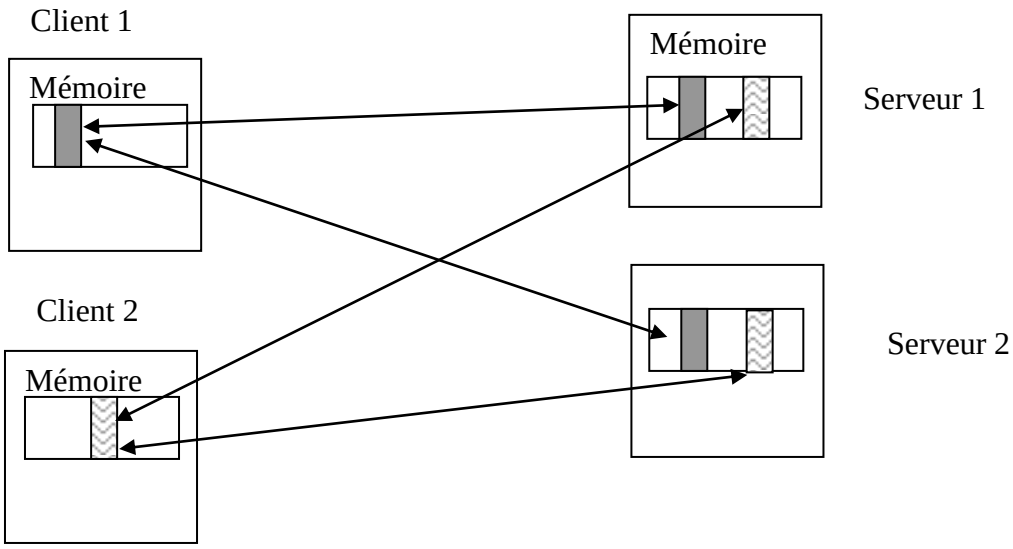
Entrepôt réparti en mémoire

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

Sujet :

L'objectif est de concevoir un système de stockage de données persitantes. Les données sont stockées dans un segment mémoire dupliqué sur N serveurs. Chaque client s'attache au segment de mémoire persistante dans laquelle il peut créer une donnée, lire et modifier des données déjà existantes. Les données présentes dans cette zone mémoire sont transmises aux serveurs répliqués.

Le schéma suivant présente l'architecture :



Le segment est découpé en pages, il est répliqué sur tous les serveurs. Initialement, seuls les serveurs ont le droit de lire et d'écrire sur le segment.
Le client doit tout d'abord « mappé » le segment dans sa mémoire (utilisation de mmap). Initialement, il n'a aucun droit en lecture et écriture sur cette zone. Ensuite, le client peut faire les opérations suivantes :

- Création d'une nouvelle donnée (p_create) : les serveurs réservent une zone libre dans le segment persistant pour pouvoir stocker une nouvelle données. La donnée est associée à un nom.
- Accès en lecture une donnée existante (p_access_read) : le client demande une copie locale d'une donnée déjà créée. La données ne doit être manipulable qu'en lecture. Si un autre client est en cours d'utilisation sur cette donnée en mode « read-write », l'appel est bloquant jusqu'à ce que la donnée soit libérée (p_release_readwrite).

- Accès en lecture/ecritrue une donnée existante (p_access_readwrite) : le client demande une copie locale d'une donnée déjà créée. La données est manipulable en lecture et écriture. Si un autre client est en cours d'utilisation sur cette donnée en mode read ou readwrite l'appel est bloquant jusqu'à ce que la donnée soit libérée (p_release_read ou p_release_readwrite).
- Manipulation de la donnée : une fois créée ou récupérée la donnée peut être lue et modifiée selon son mode directement en mémoire.
- Libération de la donnée (p_release_read): la donnée est libérée (l'appelant n'a plus le droit d'utiliser la donnée). Le serveur débloque alors éventuellement un client en attente de la donnée.
- Libération et mise à jour de la donnée (p_release_readwrite): la donnée est libérée (l'appelant n'a plus le droit d'utiliser la donnée) et sa mise à jour est transmise au serveur. Le serveur débloque alors éventuellement un client en attente de la donnée.

Les serveurs de stockage sont répliqués afin de tolérer les fautes. En cas de défaillance d'un serveur, les clients doivent continuer de fonctionner tant qu'au moins un serveur est en état de marche.

Le code suivant illustre un exemple d'utilisation sur un client :

```
...
int *cpt ;
double *a;

init_persistant() ;  /* Initialisation de la mémoire persitante */

p_create(sizeof(int), "compteur"); /* Création de la donnée "compteur"
*/

cpt = p_access_readwrite("compteur") ; /* Demander l'acces à la donnée
*/
*cpt = 23*45;  /* Manipulation de la donnée */
p_release_readwrite(cpt);  /* Libere et mise à jour de la donnée */
```

Mise en oeuvre :

La mise de ce projet reposera sur les sockets pour les communications entre clients et serveurs ainsi que sur l'utilisation des primitives de gestion mémoire mmap et mmprotect permettant de gérer les droits d'accès aux différentes pages. Lors des expérimentations, il faudra simuler des fautes de serveurs.