Projet Données Réparties 2

MEGEMONT Clément SEGHROUCHNI Anas

Département Sciences du Numérique - Deuxième année
2022-2023

1 Implémentation

Nous avons modifié plusieurs fonctions pour faire en sorte que lorsqu'un écrivain unlock un *SharedObject*, tous les *SharedObject* des abonnés sont mis à jour avec la nouvelle valeur. Premièrement, nous avons modifié *maj* dans la classe *SharedObject*, qui prend maintenant en argument l'objet modifié par l'écrivain, et qui change l'ancienne valeur de clui-ci par la nouvelle. Ceci permet à chaque abonné de récupérer la nouvelle version de l'objet modifié. Le verrou est aussi mis en RLC pour les abonnés pour qu'ils puissent accéder à nouveau à l'objet sans passer par le serveur, puisqu'ils auront toujours dans leur cache la version mise à jour de l'objet.

2 Choix de conception

Nous avons choisi une version où les abonnés ont toujours (si possible) une version à jour de l'objet. C'est-à-dire que lorsqu'un client s'abonne,si il n'y a pas d'écrivain, son objet est directement mis à jour. Sinon, il sera mis à jour en même temps que les autres abonnés lorsque l'écrivain va lacher le verrou. Nous avons choisi cette conception afin de limiter les appels au serveur fait par les abonnés. On profite de l'appel au serveur fait par la fonction *follow* pour mettre à jour l'objet.

3 Comparaison qualitative du mode paresseux et du mode actif

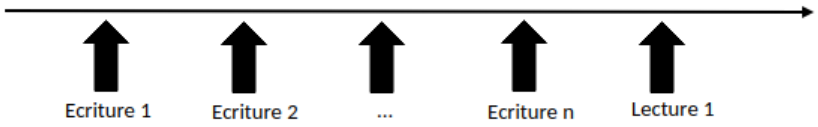

FIGURE 1 – Une lecture après n écritures

Sur le schéma précédent, dans le mode actif l'objet est modifié n fois (après chaque écriture) multiplié par le nombre d'abonnés. Ce qui rend le volume de données échangées énorme. De plus les (n-1) premieres modifications sont inutiles pour cet abonné puisqu'il demande une unique lecture. A contrario, le cas paresseux ne demande lui qu'un appel au serveur lors de l'écriture. Ainsi le volume de données échangées est diminué.

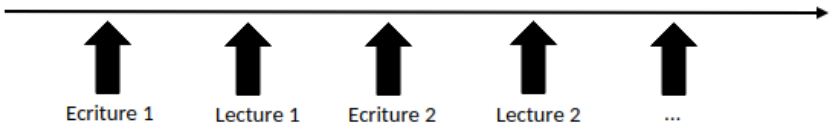

FIGURE 2 – Enchainement lecture/écriture

Dans le cas d'objets volumineux, une lecture par appel au serveur prend plus de temps qu'un appel dans le cache. Ainsi sur le schéma ci-dessus, dans le cas actif, la lecture 1 pourra s'effectuer directement puisque l'objet aura été mis dans le cache entre l'écriture et la lecture. Contrairement au cas paresseux où l'appel au serveur se fait qu'au moment de la demande de lecture. En conclusion, le mode actif est plus efficace pour des objets volumieux avec lectures dispersées.

4 Démonstration

Voir video ci-jointe.