Projet Données Réparties
Un service de partage d'objets répartis et dupliqués en Java

SNIDI Nabil
EL HABTI Ouassel
KARMAOUI Oussama

Département Sciences du Numérique - Deuxième année
2022-2023

Contents

1 Introduction

Dans cette partie, nous allons nous intéresser à la gestion active des copies dans le cadre du protocole de duplication d'objets partagés (PODP). Le but de cette étape est de compléter le PODP pour répondre aux situations où les copies doivent être mises à jour de manière coordonnée et réactive, c'est-à-dire lorsque les sites ont besoin d'une vue synchrone de l'objet partagé (cas d'abonnement).

2 Implémentation

L'implémentation de l'extension proposée se divise en deux parties. D'abord, on envoit d'une manière synchrone la dernière mise à jour de l'objet partagé au abonnés de cet dernier. Ensuite, il faut modifier la méthode `unlock` du SharedObject de telle manière à lui permettre qu'au moment où l'écriture aura lieu (`WLT`) on fera en sorte de modifier l'état de l'écrivain et envoyer la dernière mise à jour vers tout les autres clients abonnés .

2.1 Abonnement et désabonnement

On choisit de garder les memes noms de méthodes que la version précédente `(subscribe et unsubscribe)` qui jouent le role de `(track et leave_track)` pour notre cas. Pour permettre aux clients de s'abonner et de se désabonner à une vue synchrone de l'objet partagé, nous avons étendu plusieurs méthodes du `ServerObject, Client, SharedObject, Server` en ajoutant de nouveaux argument :

- **Dans l'interface du SharedObject :** `if (Client.isSubscribed(this.id)) this.lock = LockState.RLC;` : cette assertion permet de vérifier si le client est abonné au SharedObject, si oui ça permet de modifier son lock vers RLC, après on publie la dernière mise à jour de l'objet partagé vers les autres clients abonnés à l'aide de la modification de la méthode qu'on avait avant dans `Client.server.pub(Client.client,this.id,this.obj)`.
- **Dans l'interface du Client :** `public void notifyCallback(Object obj,int id)` : cette méthode permet à un client de recevoir la dernière mise à jour s'il est abonné, on a effectué à changement par rapport à la première étape pour que pour que cette méthode pour rajouter la fonctionnalité de mise à jour sans contact du Serveur.
- **Dans l'interface du ServerObject :** `public synchronized void notifySubscribers(Client_itf client2, Object obj2)` : cette méthode permet d'appeler la méthode `notifyCallback du Client` pour tout les clients abonnés sauf le client qui détient le lock pour l'écriture.

2.2 Notification des changements d'états locaux

Pour permettre aux clients de notifier le serveur de leurs changements d'états locaux, nous avons ajouté une nouvelle méthode à l'interface du client :

- `notifyCallBack()` : cette méthode permet au client de notifier le serveur d'un changement d'état de l'objet partagé. Le nouvel état de l'objet partagé est fourni sous forme de message transferé au autres clients.

Lorsqu'un client appelle la méthode `pub`, le serveur met à jour l'état de l'objet partagé correspondant et itère sur la liste d'abonnés pour cet objet partagé afin d'appeler le rappel fourni par chaque client abonné.(On voulais utiliser un Callback et on est en trains de formaliser ce cas d'utilisation)

3 IRC

Nous avons laissé comme l'étape précédente deux boutons (Subscribe et Unsubscribe) à l'interface de l'IRC ainsi qu'un panel pour les notifications et un `LightUpPanel` qui s'agit qu'une zone rouge qui devient vert lors de la réception d'une notification. Puis, on a ajouté une nouvelle bouton qui met on veut la gestion des copies actives **read_light**, en effet cette boutton aura comme ActionListener un Listener qui lit directement la dernière valeur enregistrée dans l'objet partagé sans avoir à contacter le serveur.

3.1 Vue sur l'application IRC

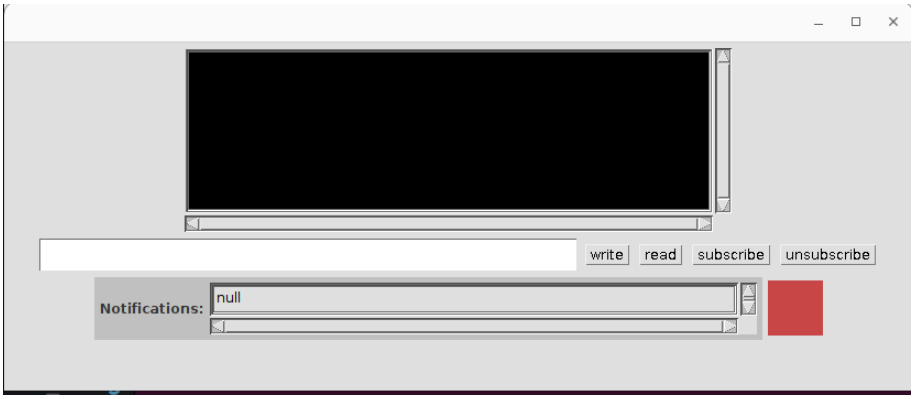

Figure 1: Vue globale sur IRC

3.2 Read_light

Après un enchaînement d'opérations subscribe/unsubscribe, vous pouvez voir que lorsqu'on fait subsribe et on fait un read_light ça récupère directement la version locale de l'objet partagé, lorsqu'on fait un write dans un client biensur.
Ceci s'affiche sur le panel tout les clients abonné sue vous pouvez le voir sur le terminal (Automatically read from cach)

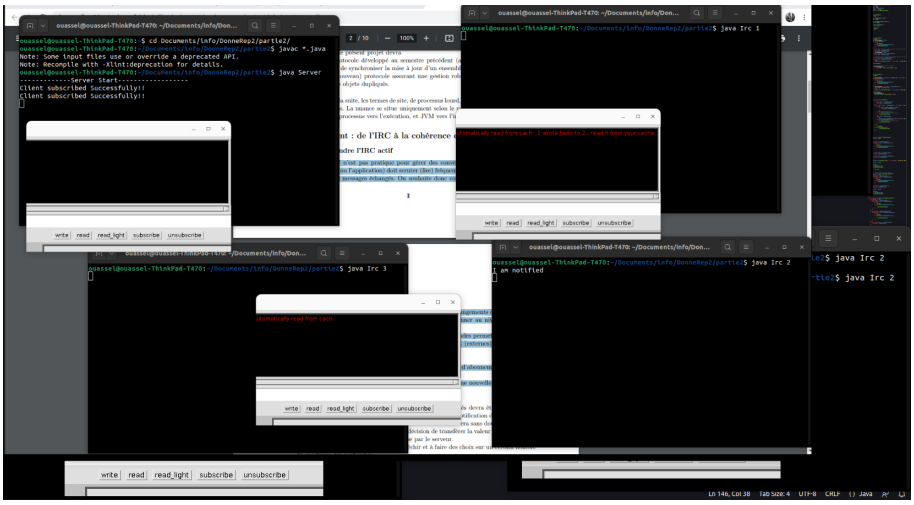

Figure 2:

3.3 Conclusion

En conclusion, au cours de l'étape 2, nous avons proposé une extension du protocole PODP pour répondre aux situations où les copies doivent être mises à jour de manière coordonnée et réactive (cohérence continue). Pour cela, nous avons introduit des méthodes permettant aux sites de s'abonner et se désabonner afin de benificier d'une vue synchrone de l'objet partagé (envoie automatiques aux sites et possibilité d'acceder au cache sans passage par le serveur à l'aide de "read_light"), et de notifier le serveur de leurs changements d'états locaux. Vu qu'il n ya pas encore de read() et write(), afin que les sites qui ont besoin d'une bue synchrone de l'object partagé récupèrent la version finale en faisant un aller-retour de "requetes" entre SharedObject et ServerObject en passant par Client puis Server pour les sites. Cette extension permet donc une gestion plus efficace des écritures et des lectures sur des objets volumineux, où la mise à jour des copies est asynchrone et à l'initiative des clients. En somme, l'étape 2 a permis d'améliorer le protocole PODP en introduisant une cohérence continue pour les objets partagés.

Note: Quelques éléments Swing de l'interface graphique n'ont pas encore été utilisés, mais nous les affinerons prochainement. Nous ajouterons également d'autres fonctionnalités intéressantes qui sont associées à ces éléments.