Rapport final
Projet Données Réparties

Adrien Chevallereau
Dino Gurnari
Ghislain Réveiller

Département Sciences du Numérique
Deuxième année - Architecture, Systèmes et Réseaux
2021-2022

1 Architecture

Voici l'architecture de notre projet, les méthodes n'ont pas été représenté pour rendre cela plus clair :

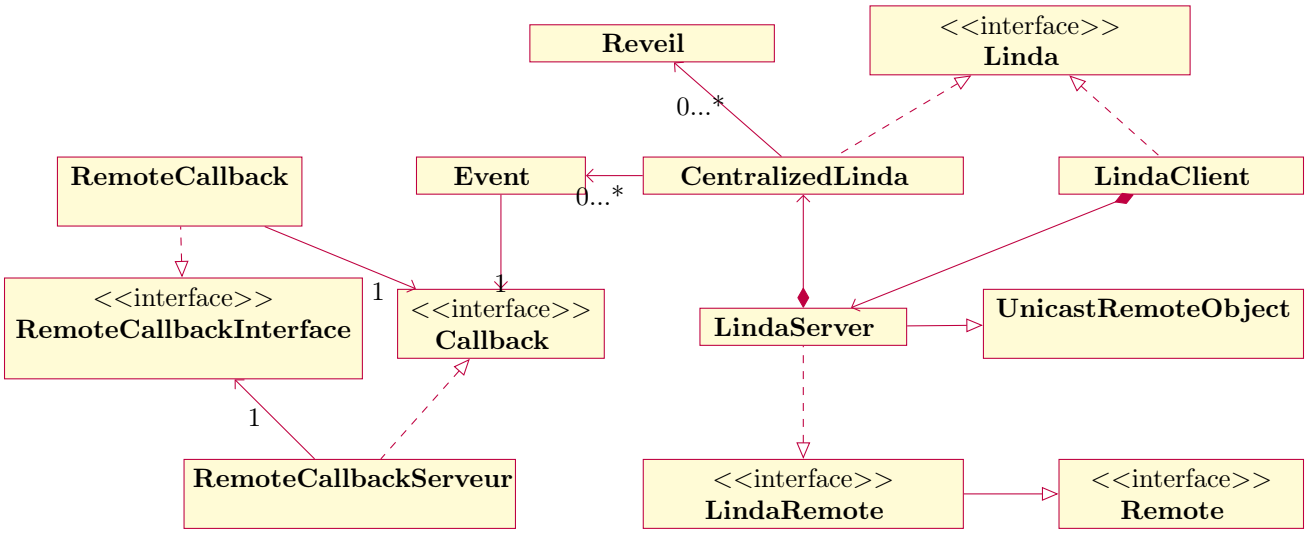

FIGURE 1 – Diagramme UML principal

2 Algorithmes principaux

2.1 Linda Centralisé

Write

On ajoute le Tuple demandé puis on regarde si des `Read` sont en attente puis s'il y a des `Take`. S'il y en a on les reveil.

Read

On regarde si un `Tuple` correspondant au *template* est dans la liste de `Tuple`, si oui on renvoie une copie de celui-ci, sinon on instancie un `Reveil` que l'on ajoute à la liste des *read* en attentes.

Take

Comme pour `Read` mais on supprime le `Tuple` de la liste.

ReadAll

On parcours la liste des `Tuples` et on récupères tout ceux qui corresponde au *template*.

TakeAll

Comme pour le `ReadAll` mais nous sommes obligés de passer par `tryTake` pour ne pas modifier directement la liste sur laquelle on itère.

EventRegister

Dans un premmier temps, si le timing est `IMMIDIATE` on regarde si le tuple est présent dans le serveur et on renvoi le callback si oui. Sinon on enregistre l'evenement dans la bonne liste (`READ` ou `TAKE`). Nous avons 2 listes différentes car nous avons décidé de faire tout les `READ` avant les `TAKE`. Ainsi à chaque write, on test si un tuple correspond à un des event enregistré dans la liste et on appelle le callback si positif puis on supprime l'event de la liste.

2.2 Crible d'Ératosthène

Nous avons réalisé deux versions du crible d'Ératosthène, une première, séquentielle, reprenant la logique de l'algorithme original, c'est-à-dire que nous générons tout les entiers de 2 à n puis en partant de 2 nous retirons tout les multiples de celui-ci puis nous passons à l'entier suivant qui n'ayant pas été retiré est donc premier.

Pour la version concurrente nous utilisons un pool de `Threads` qui prennent les entiers restant dans l'ordre et retire leurs multiples. Nous avons créé la class `ThreadErathosthene` pour le cette implantation.

Pour les deux versions nous nous appuyons sur l'interface `Eratosthene` qui fournis la génération des n premiers entiers dans un `Linda` et qui fournis également le retrait des multiples de `i` dans un `Linda`.

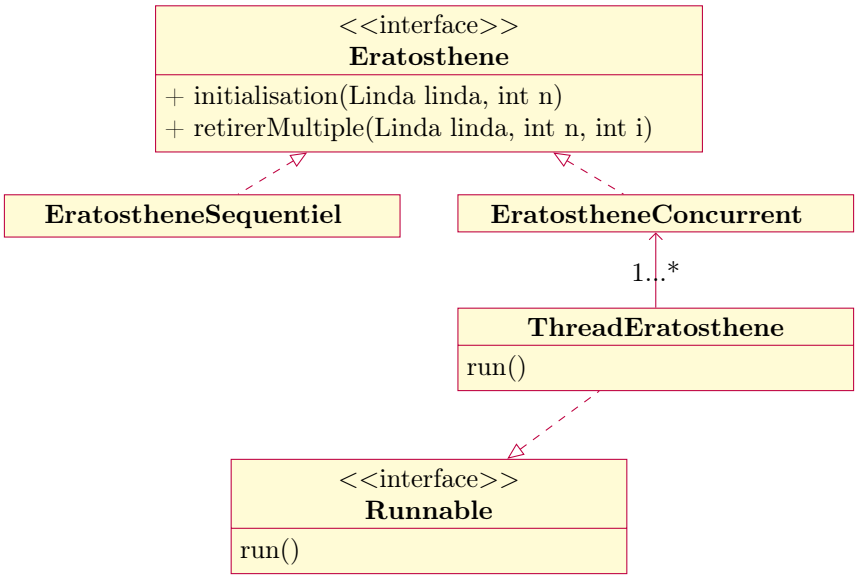

FIGURE 2 – Diagramme UML du crible

3 Difficultés rencontrées et leur résolution

Dans le Linda Centralisé :

— Gestion des `Read` et `Take` en attentes : Nous avons créé une classe `Reveil` qui a en attributs le `Tuple` recherché et le `Thread` qui sera réveillé lorsque l'on ajoutera le `Tuple` avec un *write*. Ce `Reveil` est ajouté dans une liste qui sert de file d'attente, en effet l'ordre de réveil est le suivant :
1) *Tout les* `Read` *correspondants*
2) *Le premier* `Take` *correspondant*

Dans le Linda Serveur :

— Implantation de la primitive d'abonnement eventRegister : Pour faire cela nous avons du faire l'inverse de l'implantation du `LindaServer`, c'est-à-dire que via RMI le `LindaServer` instancie une référence *remote* au `Callback` du `LinadClient`. Ainsi il peut faire un appel à sa méthode `call`.

4 Exemples originaux

4.1 Crible d'Ératosthène

Nous avons créé un test de comparaison (`ComparaisonTempsCrible.java`) de temps d'exécution pour les deux versions du crible. Ce test détermine le temps d'exécution pour différentes valeurs de n pour les deux cribles. Les valeurs que nous avons utilisées sont : `{10, 50, 100, 500, 1000, 5000, 10000}`.

Voici les résultats obtenus pour un pool de 10 `Threads` :

Valeurs de n	10	50	100	500	1000	5000	10000	50000
Séquentiel	39 ms	6 ms	12 ms	47 ms	45 ms	224 ms	594 ms	43425 ms
Concurrent	6 ms	16 ms	13 ms	21 ms	29 ms	293 ms	876 ms	28328 ms

Nous observons que notre implantation du crible d'Ératosthène concurrent à l'aide d'un pool de `Threads` est parfois moins efficace que la version séquentielle, cela peut être dû au temps de création des `Threads` qui peut être long et qui selon le nombre d'entiers peut ralentir l'algorithme.

4.2 Recherche approximative améliorée

Nous avons pu implanter la prise en compte de plusieurs chercheur dans le programme de recherche approximative. Faute de temps nous n'avons pas pu ajouter plus de fonctionnalités.