Projet Données Réparties

Rapport Final Linda ++

Kamal Hammi
Mohamed Ikich
Salahdine Ouhmmiali
Groupe G10

Département Sciences du Numérique - Deuxième année
2021-2022

1 Implementation de la Cache

La cache est implementé en utilisant une liste des tuples propre au client et dedans on ajoute les tuples déja lus depuis le serveur, ce dernier est interrogé aprés ne pas avoir trouvé le tuple voulue dans la cache. On a utilisé un CallBack de correction (qui a pour but dans le call de supprimer le tuple en question dans la cache) pour invalider les tuples présents dans les caches mais pris dans le serveur, en s'abonnant avec le tuple lu depuis le serveur, qui stocke les callback dans une liste spécifique, et qui à chaque supprission d'un tuple dans l'espace des tuples, le serveur appelle la fonction corriger pour éxecuter le call des callbacks de correction chez les clients.

2 Tests de la Cache

1. Le fichier TestCache.java contient un test de la cache dont on crée un thread qui écrit des tuples puis un autre qui lit un motif trois fois, et donc il nous a apprus aprés l'éxecution que bien le thread lit le tuple la 1er fois depuis le serveur, et la 2eme et 3eme en cache.
2. Le fichier TestCache2.java est le mème que le premier test sauf qu'il y a un autre thread qui prend le tuple voulue aprés la 1er lecture depuis le serveur et la 2eme depuis la cache, et donc la 3eme lecture se blocke puisque vraiment le tuple n'est pas présent ni dans la cache ni dans le serveur (la correction est bien faite).
3. Le fichier TestCahce3.java est le mème que le 2eme sauf que le thread qui écrit ajoute un autre tuple de mème motif voulu, et donc le thread lisant ne se blocke pas dans la 3eme lecture et il lit depuis le serveur le tuple qu'il trouve.

3 Developpement d'une application de Test des performances

On a developper une application de test des performances en temps (fichier ApplicationTestCacheMultiThread.java), qui prend en paramètres un nombre de lecture L, un nombre de Threads qui lisent R, un nombre de threads qui écrient W, et un nombre de threads qui take T. Donc l'application lance ces threads chacun a son client et les threads qui lisant lit L fois. Après l'exectution de cette application avec la linda sans et avec la cache on a calculé le temps mis en fonction de L et on a obtient le résultat dans la figure 1.

On remarque que la version de cache est plus rapide que celle de sans cache, ce qui est prévue.

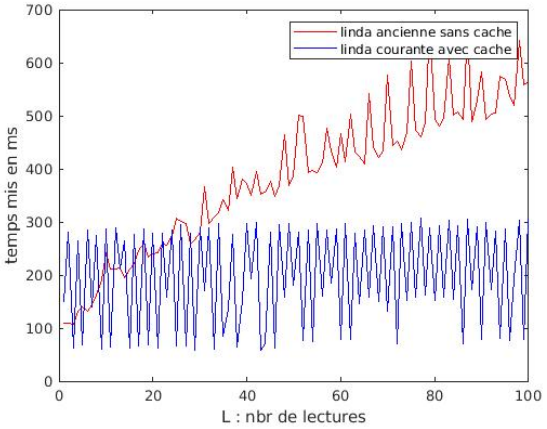

FIGURE 1 – Le temps mis par l'application dans le cas du linda sans cache et avec cache en fonction du nombre de lecture L