Revue du travail réalisé par le binôme Wahl-Echcherqaoui sur la partie Hdfs Projet données réparties-hidoop

Chaimaa Louahabi - Ahmed Ghanim

December 23, 2020

Contents

1 Partie Technique

1.1 Présentation de l'architecture

Le code pour la partie HDFS se décompose en trois classes : HdfsClient, HdfsServer et HdfsFile. Le client représente l'utilisateur qui envoie son fichier sur le serveur et le récupère. Le serveur est le point d'entrée du cluster qui s'occupe de décomposer et de recomposer les fichiers envoyés par le client. Chaque ordinateur ou noeuds du cluster lance un daemon qui va récupérer des fragments de fichier. Il y a trois fonctionnalités réalisables pour un utilisateur : hdfsWrite qui envoie un fichier sur le serveur, hdfsRead qui récupère un fichier depuis le serveur et hdfsDelete qui supprime un fichier du serveur.

1.2 Tests et Résultats

- HdfsServer ne doit pas être serializable. Par contre, il doit pouvoir recevoir les commandes de HdfsClient en utlisant des sockets, et puis il peut qu'il les exécuter soi-même en créant des ServerSlave(qui extends Thread) qui l'exécute pour lui, ce qui augmentera le degré du paralélisme.
- La communication en mode TCP doit être entre HdfsClient et HdfsServer et pas entnre HdfsNameNode et HdfsServer, vue que HdfsNameNode ne s'occupe que de la gestion des ports des serveurs et des chuks des fichiers de HDFS.

2 Synthèse

2.1 Critères d'évaluation

1. Correction: Le produit n'est pas encore fini d'où l'impossibilté de faire des tests.
2. Complétude: même s'il y a des points qui ne sont pas encore implémentés mais ils sont présents dans la squelette du code réalisé (des fonctions avec un corps vide) ce qui fait que tous les points de la spécification ont été abordés .
3. Cohérence:
   - il semble que HdfsServer et hdfsNode ont le même rôle. On peut se contenter de HdfsServer et lui ajouter les deux fonctions de HdfsNode.
   - La structure est claire.

2.2 Pistes d'amélioration

Vue que HdfsClient n'est pas encore implanté, voici les choix de conception que nous avons envisagés :

- HdfsNode serait un remoteObject, pour que le hdfsClient puisse lui appeler à distance et lui demander d'écrire ou lire un chunk.

- hdfsClient obtiendra les informations concernant les ports des dataNode en se connectant à hdfsNameNode.
  Remarque : Pour lire un fichier à partir de HDFS, HdfsClient aura besoin d'obtenir depuis NameNode les ports et l'ordre des chuncks. Pour écrire dans HDFS, il aura besoin des numéros des ports seulement, donc, ça sera bien de distinguer entre une requête d'écriture et une requête de lecture.

- Nous trouvons que c'est plus simple de fragmenter un fichier selon le nombre de data nodes disponible. S'il y a N dataNode et le fichier contient M enregistrements, alors un bloc contiendra M/N enregistrement (si M n'est pas divisible par N, on cherche X le plus grand diviseur de M qui est inférieur à N, et donc le fichier sera fragmenté sur X dataNode.