Rapport final HDFS

BARIOZ Clément et VINCENT Léo

1. Introduction

L'objectif de cette partie est d'implémenter le service HDFS de la plateforme et en particulier la classe HdfsClient qui permet de manipuler les fichiers dans HDFS.

Opérations

Pour cela nous avons implémenté les 3 méthodes principales de la classe selon leur spécification :

- **HdfsWrite(Format.Type fmt, String localFSSourceFname, int repFactor)** permet d'écrire un fichier dans HDFS. Le fichier localFSSourceFname est lu sur le système de fichiers local, découpé en fragments de taille fixe, et les fragments sont envoyés pour stockage sur les différentes machines. fmt est le format du fichier (Format.Type.LINE ou Format.Type.KV). repFactor est le facteur de duplication des fragments ; pour cette version il sera considéré comme valant 1 (pas de duplication).
- **HdfsRead(String hdfsFname, String localFSDestFname)** permet de lire un fichier de nom hdfsFname à partir de HDFS. Les fragments du fichier sont lus à partir des différentes machines, concaténés et stockés localement dans un fichier de nom localFSDestFname.
- **HdfsDelete(String hdfsFname)** permet de supprimer les fragments d'un fichier stocké dans HDFS.

2. Implémentation

HdfsWrite

Pour cette méthode, la première étape est de découper le fichier initial en sous fichiers (chunks). Pour cela, on détermine le nombre de chunk nécessaire en fonction de la taille du fichier à découper. Ensuite, on créer les sous fichiers correspondant et on écrit dans chacun d'eux une partie du fichier. Une fois les chunks créés, on les range dans une hashmap qui prend pour clé leur nom et pour valeur les chunks eux même.

La deuxième étape consiste à les répartir sur les différentes machines. Cette première version ne prend pas en compte la valeur de repFactor étant ici considéré comme valant 1. La solution proposée ici consiste à envoyer chaque chunk sur une machine choisit au hasard en fonction des ports disponibles. On écrit ensuite tout simplement le contenu du chunk en question sur le port en créant les sockets appropriés.

Une fois cela fait en garde en mémoire la localisation de chaque chunk sur les machines en mettant un jour la variable globale localisation qui est une hashmap qui a pour clé le nom des chunk et en valeur une liste d'entier (ici la liste ne contient qu'un seul élément) qui est la liste des ports sur lesquels se trouve le chunk.

La méthode semble fonctionner correctement après plusieurs tests concluants.

HdfsDelete
Pour chaque DataNode du système de fichiers HDFS, on crée un socket entre ce DataNode et le NameNode, on supprime le fragment et on ferme le socket.
La méthode semble fonctionner correctement après plusieurs tests concluants.

HdfsRead
On crée tout d'abord un fichier local nommé localFSDestFname qu'on ouvre en écriture. On retrouve ensuite toutes les machines contenant un fragment du fichier qu'on veut lire. Pour chaque DataNode contenant un fragment, on établit une connexion (via un Socket) et on écrit les lignes du fragment dans le fichier local. On ferme ensuite le socket et on passe au fragment suivant jusqu'à ce qu'il ne reste plus aucun fragment. On peut ensuite fermer le fichier local en écriture.
Cependant, la méthode renvoie un fichier contenant des caractères illisibles même si elle termine. On pense que cela vient du fait qu'on ouvre plusieurs InputObjectStream / OutputObjectStream et qu'il faudrait les fermer au fur et à mesure mais on arrive pas à trouver lequel....

3. Prise en compte des évaluations de la partie Hdfs

Nous avons ajouté le démon HdfsServeur, qu'on lance sur chaque machine afin de bien répondre au sujet. Par conséquent, nous avons utilisé les sockets TCP afin de faire communiquer les clients avec le serveur.
Dorénavant, le client envoie bien un message au serveur du type (nom de commande, nom du fichier) comme par exemple (CMD_READ, fichierA) pour lire le fichierA.
Nous avons également revu les méthodes Write, Read et Delete afin de les adapter au format Client/Serveur.

4. Idées d'amélioration

- Factoriser le code pour éliminer les redondances (notamment lors de la lecture et l'écriture de fichier avec des InputStream/OutputStream)
- Ajouter une interface via le terminal afin de saisir les adresses IP des machines et les numéros de port
- Ajouter le facteur de réplication
- Eviter de couper une ligne en deux lors de la création de fragment avec la fonction Write.
- Revoir la fonction Read afin qu'elle donne le bon résultat et non un fichier illisible...