Projet Hidoop

Projet de données réparties - Réponses HDFS

LAPLAGNE Chloé
RAZAFIMANANTSOA Nathan

Ce document présente les corrections et réponses à l'évaluation sur notre première version de HDFS, ainsi que les améliorations apportées.

1) Réponses aux remarques de l'évaluation intermédiaire

- **Correction de Bugs** :

  - Contexte : *en cas de coupure de connexion avec un des serveurs, la suppression mettait bien les métadonnées à jour, laissant les fragments du fichier sur le serveur*.

    Ce bug a été corrigé, les métadonnées ne sont modifiées que si la connexion avec tous les serveurs requis a réussi (les pannes de serveurs ne sont pas gérées dans cette version).

- **Pistes d'amélioration** :

  - Proposition : *dans HdfsWrite, remplacer le paramètre 'taille d'un chunk' par 'nombre de chunks'*.

    Ici, lorsqu'un utilisateur écrit un fichier il connaît la taille de ce fichier. En revanche, il ne sait pas forcément combien de serveurs sont en ligne à un moment donné et ce nombre peut être variable.
    Indiquer la taille des chunks permet de mieux se représenter ce qui sera traité par une opération Map et d'adapter par exemple une taille de chunk par défaut pour un format donné.
    Nous avons donc conservé ce choix mais nous avons essayé de le rendre plus clair dans le message d'aide.
    De plus, nous avons enlevé pour simplifier le mode 'distributed' qui permettait de répartir les chunks sur tous les serveurs. Ce mode était en

effet surtout utile au début du développement et n'est plus très pertinent sur un fonctionnement classique.

2) Améliorations

- **Implantation de la fiabilité** dans les échanges client / serveur :
    - Mise en place de codes d'erreur et de retour codés sur un entier de type long. Ces codes sont échangés entre serveur et client et leur permettent de communiquer :
        - -1 **FILE_NOT_FOUND** : fichier introuvable ou vide sur le serveur
        - -2 **FILE_EMPTY** : fichier vide envoyé sur le serveur (ignoré)
        - -3 **CHUNK_TOO_LARGE** : taille du chunk trop importante par rapport à la moyenne annoncée (comme on coupe par ligne, la limite d'un chunk faisant 2 * chunkSize semble raisonnable: on a peu de chances d'avoir une ligne qui fait la taille d'un chunk...)
        - -4 **INCONSISTENT_FILE_SIZE** : fichier reçu de taille différente de celle annoncée par le serveur
        - -5 **IO_ERROR** : erreur de communication ou de système de fichier
        - -6 **FILE_TOO_LARGE** : place disponible sur le serveur insuffisante pour le chunk
    - Affichage des codes d'erreur sur les communications ayant échoué entre client et serveur.

- **Restructuration de HdfsClient** :
  Les 3 fonctions principales de HdfsClient (Write, Read et Delete) ayant un schéma de fonctionnement similaire (récupération des chunks, lancement d'une tâche pour chaque chunk puis récupération des résultats), nous avons choisi de rassembler les parties communes en une classe abstraite `ClientServerTask`. Chaque tâche (`Writer`, `Reader` et `Deleter`) hérite ensuite de cette classe pour implémenter son fonctionnement.

- **Implémentation du NameNode** :
  En prévision de la prochaine version, nous avons choisi d'implémenter un daemon en RMI tournant en local et gérant les métadonnées et la récupération de la liste de serveurs disponibles. Cela permet de ne pas avoir à lire le fichier de configuration à chaque commande de l'utilisateur et permettra notamment la gestion des pannes de serveurs dans la version suivante.

3) Tests réalisés

- Les tests unitaires dans HdfsTest.java concernent des fonctions auxiliaires de la classe Constantes.java.
- Le script *testHdfs.sh* permet de lancer Hdfswrite puis HdfsRead en local et de vérifier que le fichier lu est bien le même que celui de base.
  Il permet aussi de voir les temps d'exécution.
  Commande : `$ ./testHdfs.sh <fichier> [options_hdfswrite]`
- Tests fonctionnels avec medium.txt (4.1MB) puis GO (2.9GB) sur 4 machines serveurs situées à l'N7 et une machine client connectée par le VPN (débit autour de 5.5Mo/s) :

```
hidoop> hdfs -w medium.txt --chunks-size=1MB
Splitting file in 5 chunks...
# 100 %
medium.txt successfully saved.
-- time (ms) : 1796
hidoop> hdfs -r medium.txt
Reading file...
# 100 %
medium.txt successfully read to r_medium.txt.
-- time (ms) : 1264
hidoop> hdfs -d medium.txt
Deleting file...
# 100 %
medium.txt successfully deleted.
-- time (ms) : 103
hidoop> hdfs -w GO --chunks-size=64MB
Splitting file in 46 chunks...
# 100 %
GO successfully saved.
-- time (ms) : 534723
```

- Test d'écriture sur le réseau de l'N7 : pour un fichier de 6.5 Go, le temps d'écriture du fichier dans HDFS est resté relativement stable [68.2s ; 72.4s] pour un nombre de chunks allant de 4 à 384.