17/12

Evaluation HDFS

Corentin Dominguez, Guillaume Claverie

Partie technique

Présentation et résultats des tests

Premier test :

- Le premier test a été de tenter de démarrer un HDFS. Ensuite, nous avons utilisé un client pour tenter d'uploader un fichier, sur un node. On remarque que cela entraîne la création d'un fichier fragment dans le dossier node-1.
- Étant donné que nous n'avons utilisé qu'un seul node, ce fragment est en fait identique au fichier uploadé, ce qui est bien le résultat attendu.
- Dernière étape : utiliser un HDFS client pour récupérer le fichier présent sur le HDFS : Fonctionne, on récupère bien le fichier qui a été envoyé d'origine.

Test avec plusieurs nodes :

- Tentative de lancer 2 serveurs : le deuxième n'est pas pris en compte. On remarque que l'adresse et le port sont hardcodés dans le client, donc il ne peut pas prendre en compte plus d'un serveur.

Tests cas invalides :

- Lorsqu'on démarre le client sans serveur à la destination attendue, on a une erreur (Connexion refusée + affichage du stack)
- Lorsqu'on tente d'écrire un fichier vers le HDFS qui n'existe pas sur la machine du client, on a une erreur (File not found + affichage du stack)
- Lorsqu'on tente de read avec un HDFSClient un fichier qui n'existe pas sur le serveur, on a une erreur (EOFException + affichage du stack).

Correction : validation (ou non) des fonctionnalités

D'après les tests, les fonctions Read et Write fonctionnent, mais avec un seul node uniquement (conformément à ce qui était indiqué dans leur rapport intermédiaire).
La spécification de la fonction delete est présente, mais la fonctionnalité n'est pas encore implémentée.

Performances

Pas de remarque pertinente pour l'instant.

Bugs détectés et propositions de correction

Pas de bug détecté en écriture ou lecture dans les cas attendus avec un seul node à l'écoute sur le "bon port" (celui sur lequel le client essaye de se connecter).
En revanche pour les cas invalides, par exemple lorsque le serveur n'est pas lancé ou que le port client/serveur n'est pas le même, on repère une exception de connexion refusée.
De plus, toujours pour les cas de test cas invalides, écrire ou lire un fichier qui n'existe pas lève une exception non traitée.

Commentaires et propositions sur la qualité du code

Code bien indenté, et commenté. Nommage adapté. RAS.
Il n'y a pas d'élément qui empêche la compréhension du code fourni.
Attention peut être au fait d'écrire les adresses et ports directement dans le code, sans doute car il est encore en cours de réalisation.

Synthèse

Correction

Le code fonctionne correctement pour le cas nominal avec un nœud dans le HDFS. Ce cas correspond à la situation où le client choisit de read ou de write un fichier qui existe, c'est-à-dire qui a un reconnu par le HDFS. Pour ce cas, on peut dire que le résultat retourné est juste. En effet, si l'on cherche à lire un fichier présent dans le HDFS, on obtient bien le fichier dans sa globalité.

De plus, si l'on écrit sur le fichier puis on lit le résultat de cette écriture, on remarque que l'écriture a bien été effectuée.

En revanche, pour les cas invalides présentés dans la partie technique, le code fourni ne fonctionne pas car ces cas provoquent l'arrêt du programme.

Complétude

Les trois essentiels de la spécification correspondent aux méthodes read, write et delete qui permettent respectivement de lire un fichier, écrire un fichier et supprimer un fichier.

Les deux premiers points ont été abordés. Effectivement, on peut lire un fichier et écrire sur un fichier présent dans le HDFS. En revanche, ils n'ont été pour l'instant que partiellement traités car comme nous l'avons déjà relevé, on ne fait cela qu'avec un nœud présent dans le HDFS.

Concernant la fonction qui permet de supprimer un fichier, elle n'a pas encore été abordée.

Pertinence

Pour le transfert de données, on remarque l'utilisation de socket et de ObjectStream, ce qui nous semble pertinent dans cette situation

Comme nous avons pu le constater lors de nos tests de notre partie, qui utilise le client HDFS, les fonctions associées à la lecture et à l'écriture respectent bien leur signature et la spécification. (Fonction delete pas encore disponible)

Cohérence

Le code est clair et respecte bien la squelette fourni, et il n'y a pas de redondance, ni de code inutile.

De plus, les éléments sont bien nommés, et accompagnés des commentaires réguliers permettent une compréhension et une lecture facile du code

Propositions de pistes d’amélioration

Points d’améliorations	Motivation
Meilleure gestion des exceptions dues à une mauvaise utilisation de l’utilisateur On pourrait par exemple afficher l’usage attendu.	Certains cas d’utilisation arrêtent le programme avec peu d’indications pour comprendre d’où vient le soucis
Ne pas mettre les adresses et ports à l’intérieur du code	Utiliser des variables permettrait de faciliter la modification (par exemple, si on voudrait modifier l'adresse du serveur, on devrait chercher partout où son adresse apparaît actuellement… avec une variable, il n’y aurait qu’un modification à faire)