EVALUATION DU LOT A

BERNOUSSI Aymen

MESKINE Hatim

Partie technique :

Résultats des tests : Calcul du nombre d'occurrences des mots dans des fichiers de tailles différentes (Lancement en local) :

Les tests sont réalisés avec des données relativement petites (pas très volumineuses). Du coup, on ne peut pas observer le boost de performance qu'on peut obtenir avec ce type de service.

Nombre de lignes	468425	1536885	3090990	9272971
Version itérative : Count	503	1255	2057	6185
MyMapReduce	1970	3097	3772	9654

Correction (validation des fonctionnalités) :

Validation du contenu du répertoire hdfs qui contient les fichiers concernant la partie HDFS.

- **HdfsServer.java :** Le serveur traite correctement les différentes commandes qu'il reçoit. Il arrive à effacer les fichiers qu'il a créés et écrire de nouveau fichier. Cependant en essayant de lire un fichier, on crée un fichier de type ***KVFormat*** ce qui ne correspond pas nécessairement au type du fichier. La méthode ***HdfsRead*** fonctionne pour les fichiers contenant des ***KV*** mais pas pour d'autres types.

- **HdfsClient.java :** les méthodes implémentées sont : *HdfsWrite* , *HdfsDelete* et *HdsfRead*. Les deux premières semblent fonctionner correctement mais la dernière a le même problème que dans ***HdfsServer***. La lecture suppose que le fichier contient des données de type ***KV*** ce qui cause des problèmes lors de la lecture de fichiers d'autres types.

- **Informations.java :** Classe contenant les données à envoyer lors de l'écriture d'un fichier dans un serveur.

Performances :

- On remarque que les commandes **“READ”** et **“DELETE”** fonctionnent correctement pour tous les fichiers que nous avons testés. Cependant la commande “WRITE” nous donne une erreur ***Exception in thread “main” java.lang.OutOfMemoryError: Java heap space*** pour des fichiers de grande taille (A partir de 10.000.000 de lignes à peu près).

- Il serait préférable de lancer les commandes du serveur en parallèle en utilisant des threads. Dans la version actuelle, ce serveur se retrouve bloqué et ne peut pas recevoir de nouvelles commandes.

Qualité du code :

- Code difficile à comprendre avec des fonctions lourdes qui peuvent être simplifiées.
- Possibilité d’améliorer le code en ajoutant des classes d’aide qui contiennent le code à faire pour chaque commande.
- Ajout de commentaire pour mieux comprendre le rôle de chaque partie du code

---

Synthèse :

Correction :

- Comme vu précédemment, la commande ***“READ”*** chez le serveur et le client doit encore être améliorée pour pouvoir prendre en compte tout type de fichier.
- Le traitement d’exception doit être amélioré. En effet, actuellement, une erreur entraîne la fin du processus ***“HdfsServer”*** et il doit donc être relancé manuellement.

Complétude :

- Il était demandé d’implémenter un système de fichier local pour plusieurs serveurs. Le service Hdfs implémenté permet de faire ceci en simulant le fonctionnement sur plusieurs machines par un fonctionnement sur une seule machine mais dans différents sockets.
- On arrive à faire des lectures/écritures cohérentes.
- Les formats de données pour la lecture ne sont pas encore terminées.

Pertinence :

- Le travail présenté répond à ce qui est demandé :
    - Chaque machine (ici socket) lance un serveur et attend des commandes provenant d’un client.
    - Trois commandes sont proposées : “READ”, “DELETE” et “WRITE”
    - Les trois commandes arrivent à faire correctement ce qui est attendu (sauf gérer le type pour la lecture et le problème de mémoire pour l’écriture)

Cohérence :

- Le travail réalisé respecte l’architecture déjà définie et réalise le travail demandé.

Améliorations :

- Lancer des threads dans le serveur lors d'une réception d'une commande
- Gérer le type des fichiers pour pouvoir lire (ceci pourrait se faire soit en ajoutant un serveur qui contient des informations sur les fichiers de chaque serveur ou en créant un objet dans le serveur qui arrive à garder ces informations).
- Gérer plus efficacement l'écriture des fichiers pour essayer d'éviter les problèmes de mémoire.