Projet Hagidoop

AKKAR Khadija - ABOUMEJD Wissal (L1)

Travail réalisé :

FileReaderWriter

Nous avons débuté en implémentant l'interface FileReaderWriter de deux façons : une pour le type de fichier texte (txt) et une autre pour les fichiers au format clé-valeur (kv).

Le service HDFS

Nous avons établi la communication entre le client Hdfs et le serveur Hdfs en utilisant des sockets en mode TCP. Une fois la connexion testée et établie entre le client et plusieurs serveurs dans le local avec différents ports, nous avons implémenté les méthodes suivantes:

- `HdfsWrite` : découpe le fichier local en fragments et les envoie sur plusieurs machines pour le stockage.
- `HdfsRead` : lit les fragments depuis différentes machines, les concatènes et les stocke localement.
- `HdfsDelete` : supprime les fragments d'un fichier stocké dans HDFS.

Nous avons testé ces trois méthodes en récupérant des commandes sur l'inputStream dans la classe du serveur, que ce soit pour lire ("read"), écrire ("write"), ou supprimer ("delete"), en utilisant une méthode `main` dans la classe HdfsClient avec le fichier `filesample.txt`.

Service Hagidoop

Worker

Dans notre conception, la communication entre les workers et JobLauncher est réalisée via RMI (Remote Method Invocation), choisi pour sa capacité à faciliter la communication distante entre les différentes parties de notre système distribué.

La classe `WorkerImpl` est une implémentation concrète de l'interface `Worker`. Elle représente un nœud de traitement au sein du système distribué. Nous avons implémenté le corps de la méthode `runMap`, qui démarre les opérations de map pour chaque worker. De plus, nous avons ajouté une fonction qui renvoie le chemin absolu du répertoire de travail du Worker, facilitant la gestion des chemins dans les opérations de traitement.

La classe principale de cette classe est le point d'entrée qui initialise un serveur de noms RMI sur un port spécifié, crée une instance de `WorkerImpl`, la lie à l'URL spécifiée, permettant ainsi aux clients de l'invoquer à distance.

JobLauncher

Cette classe configure plusieurs instances de Worker en utilisant des URLs spécifiques et les lie aux Workers existants. Elle charge des données à partir de fichiers, les distribue aux Workers pour effectuer des opérations de type Map. Après la phase Map, la classe effectue une opération "reduce" en utilisant `MyMapReduce`. Les résultats des Workers sont consolidés pour produire un résultat final.

Au cours de l'avancement de notre projet, nous avons pu tester le service de Hagidoop en local sur nos machines (localhost), et finalement, nous sommes parvenus à le tester entre différentes machines, y compris les machines de l'N7.

Manuel d'utilisation :

Pour tester notre travail en mode distribué, il faut changer les noms des deux premiers éléments de la ligne 22 de la classe HdfsClient dans le package 'hdfs' et dans la ligne 12 dans la classe JobLauncher dans le package ‘daemon' par les noms des machines à utiliser. Sinon, si c'est en local, il faut les définir tous en 'localhost'.

1. Compilation dans src à l’aide de la commande javac */.java
2. Lancement des serveurs en attribuant à chacun un port via la ligne de commande.
    - Dans le terminal de la première machine, effectuer la commande :

      `java -cp src hdfs.HdfsServer`

      Écrire le numéro du port 3158

    - Dans le terminal de la deuxième machine, effectuer la commande :

      `java -cp src hdfs.HdfsServer`

      Écrire le numéro du port 3282

3. Lancement des workers en attribuant à chacun un port via la ligne de commande.
    - Dans le terminal de la première machine, effectuer la commande :

      `java -cp src daemon.WorkerImpl`

      Écrire le numéro du port 8000

    - Dans le terminal de la deuxième machine, effectuer la commande :

      `java -cp src daemon.WorkerImpl`

      Écrire le numéro du port 8001

4. Lancement de notre `JobLauncher` avec le nom de notre fichier de test en tant qu'argument.
    - Dans le terminal de la 3ème machine, effectuer la commande :

      `java -cp src application.MyMapReduce PATH/filesample.txt`

      `(pour PATH c’est le path Absolu vers le filesample.txt existant dans le dossier data)`

    - Dans le dossier data dans la troisième machine, vous trouverez un fichier appelé `resultat.txt` contenant le résultat du traitement.

Performances réalisées :

Avec notre système Hagidoop, le temps d’exécution est de 1735 ms, tandis qu'avec Count, le temps d’exécution est de 64 ms.

Or, on n'est pas arrivée à tester nos programmes avec des fichiers de grands taille.