



Manuel D'utilisation

Projet De Données Répartis - HagiDooP

16.01.2024

BELAHRACH Safae - CHEKRAOUI Louiza
ENSEEIH - M1 - L12 - 2ème année Systèmes Logiciels

Vue d'ensemble

- Configuration de HDFS SERVER
- Configuration de HDFS CLIENT
- Lancement de l'application MyMapReduce

Objectifs

Comme l'objectif de ce projet est d'implanter le service **HDFS (Hagidoop Distributed File System)** avec les fonctionnalités suivantes :

- Découpage des fichiers en fragments.
- Stockage des fragments sur les nœuds du cluster.
- Copie d'un fichier du système de fichiers local dans le FS HDFS.
- Copie d'un fichier du FS HDFS dans le FS local.
- Suppression des fragments d'un fichier stocké dans HDFS,

de mettre en place un service **Hagidoop** permettant l'exécution répartie et parallèle des traitements **Map**, de mettre en place la communication entre le démon (**Worker**) responsable de l'exécution des tâches Map et ses clients à l'aide de **RMI**, d'implémenter la classe **JobLauncher** pour lancer un calcul parallèle en utilisant le modèle **Map-Reduce**.

Grandes étapes

Configuration de HDFS SERVER

- >> Lancement du Serveur HDFS :

- Ouvrez un terminal.
- Exécutez la commande :

java HdfsServer <socketPortNumber> <nodeName> <rmiPort>.

Exemple :

Pour lancer les Serveurs HDFS on utilise la commande

```
java hdfs.HdfsServer 120 node1 4000
```

Cette commande permet d'un côté d'ouvrir un **ServerSocket** qui reste à l'écoute d'un potentiel client qui lui enverra les commandes à exécuter à savoir **write**, **read** ou **delete** cela est fait au biais d'un **KV** spéciale envoyée au début.

La commande permet aussi de démarrer un registre **RMI** et d'enregistrer le un **Worker (WorkerImpl)** autant que service afin qu'il puisse être utilisé pour lancer la fonction map par la suite.

Cette commande configure un serveur HDFS sur le nœud "node1" avec le port de socket 1099 et le port RMI 4000.

● Résultats attendus :

```
safae@DESKTOP-R8LVKKU: /m... × | safae@DESKTOP-R8LVKKU: /m... | safae@DESKTOP-R8LVKKU: /m... | safae@DESKTOP-R8LVKKU: /m... |
safae@DESKTOP-R8LVKKU: /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java hdfs.HdfsServer 1099 node1 4000
***** Listening To Commands Sockets HDFS *****
***** Setting Up RMI *****
***** RMI: rmi://localhost:4000/node1*****
```

Configuration de HDFS CLIENT

1 - >> Écriture d'un Fichier dans HDFS :

- Ouvrez un terminal.
- Exécutez la commande : `java HdfsClient write <txt|kv> <file>`.

Exemple :

```
java hdfs.HdfsClient write txt filesample
```

La commande write de **HdfsClient** permet d'écrire un fichier qui sera lu sur le système de fichiers local, découpé en fragments (autant que le nombre de machines spécifiées dans la classe **Project**) et les fragments sont envoyés pour stockage sur les différentes machines.

Cette commande divise le fichier initial en fragments et génère des fichiers `filesample-i.txt` dans le répertoire spécifié (`Projet.PATH/data`), en fonction du nombre de nœuds configurés.

● Résultats attendus :

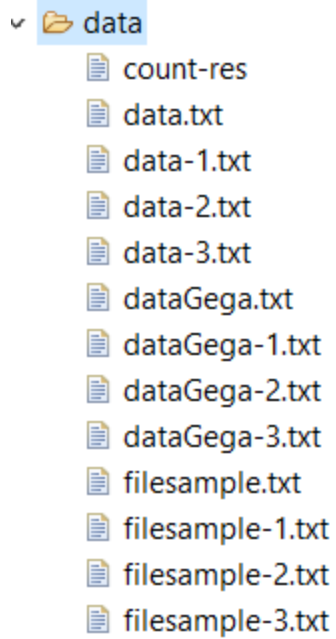
- Au niveau du terminal où on a lancé la commande, on aura :

```
safae@DESKTOP-R8LVKKU: /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java hdfs.HdfsClient write txt filesample
Connexion établie avec le serveur.
Connexion établie avec le serveur.
Connexion établie avec le serveur.
```

- Au niveau du terminal des HdfsServer, on obtient :

```
safae@DESKTOP-R8LVKKU: /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java hdfs.HdfsServer 1099 node1 4000
***** Listening To Commands Sockets HDFS *****
***** Setting Up RMI *****
***** RMI: rmi://localhost:4000/node1*****
Fragment écrit avec succès : /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/data/filesample-1.txt
```

- Au niveau du répertoire `Projet.PATH/data`, on obtient :



2 - >> Lecture d'un Fichier depuis HDFS :

- Ouvrez un terminal.
- Exécutez la commande : **java HdfsClient read <file>**.

Exemple :

```
java hdfs.HdfsClient read filesample
```

La commande de **HdfsClient** permet pour un fichier donné de récupérer ses fragments des nœuds et de reconstituer le fichier.

Cette commande permet de lire le fichier à partir des fragments stockés et de le reconstituer localement.

- **Résultats attendus :**

En supprimant par exemple `filesample.txt`, on peut le récupérer à travers ses fragments `filesample-1.txt`.

```
safae@DESKTOP-R8LVKKU:/mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java hdfs.HdfsClient read filesample
Connexion établie avec le serveur.
Connexion établie avec le serveur.
Connexion établie avec le serveur.

safae@DESKTOP-R8LVKKU:/mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java hdfs.HdfsServer 1099 node1 4000
***** Listening To Commands Sockets HDFS *****
***** Setting Up RMI *****
***** RMI: rmi://localhost:4000/node1*****
Fragment écrit avec succès : /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/data/filesample-1.txt
Fragment lu avec succès : /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/data/filesample-1.txt
Fragment écrit avec succès : /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/data/filesample-1.txt
```

3 - >> Suppression des Fragments d'un Fichier dans HDFS :

- Ouvrez un terminal.
- Exécutez la commande : **java HdfsClient delete <file>**.

Exemple :

```
java hdfs.HdfsClient delete filesample
```

La commande `delete` de **HdfsClient** permet pour un fichier donné de repérer ses fragments dans les nœuds et de les supprimer.

Cette commande supprime les fragments du fichier spécifié dans HDFS.

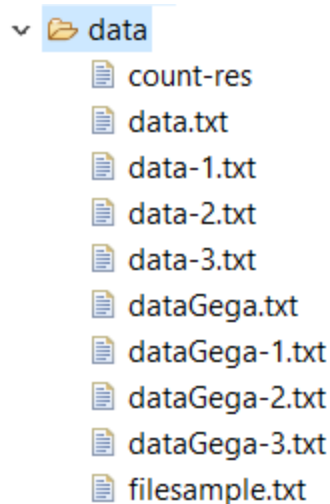
- **Résultats attendus :**

```
safae@DESKTOP-R8LVKKU:/mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java hdfs.HdfsClient delete filesample
Connexion établie avec le serveur.
Connexion établie avec le serveur.
Connexion établie avec le serveur.
```

```

safae@DESKTOP-R8LVKKU:/mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java hdfs.HdfsServer 1099 node1 4000
***** Listening To Commands Sockets HDFS *****
***** Setting Up RMI *****
***** RMI: rmi://localhost:4000/node1*****
Fragment écrit avec succès : /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/data/filesample-1.txt
Fragment lu avec succès : /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/data/filesample-1.txt
Fragment écrit avec succès : /mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/data/filesample-1.txt
Fragment supprimé avec succès : filesample-1.txt

```



Lancement de l'Application de Map-Reduce

- >> Lancement du Job Map-Reduce :


1. Ouvrez un terminal.
2. Exécutez la commande : **java application.MyMapReduce <nameFile>**

Exemple :

```

safae@DESKTOP-R8LVKKU:/mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java application.MyMapReduce filesample
***** Client connected *****
*****
***** Client connected *****
*****
***** Client connected *****
*****
time in ms =261
safae@DESKTOP-R8LVKKU:/mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/bin$ java application.Count filesample.txt
/mnt/c/Users/Safae/Documents/aPDR/version16/hagidoop/data/filesample.txt
time in ms =35

```



Pour ce cas, l'exécution du Map-Reduce sur un fichier de 2 Ko démontre que le temps requis par l'approche Count est plus performant que celui du Map-Reduce pour les fichiers de petite taille.