Projet Données Réparties : implantation du modèle Map-Reduce en Java

Rouveure Mathis
Blot Olivier

Département Sciences du Numérique Département Logiciel - Deuxième Année
Année Universitaire 2023-2024

I. Mise en place du service HDFS

Dans un premier temps, nous avons mis en place le service HDFS, représenté sur la figure ci-dessous :

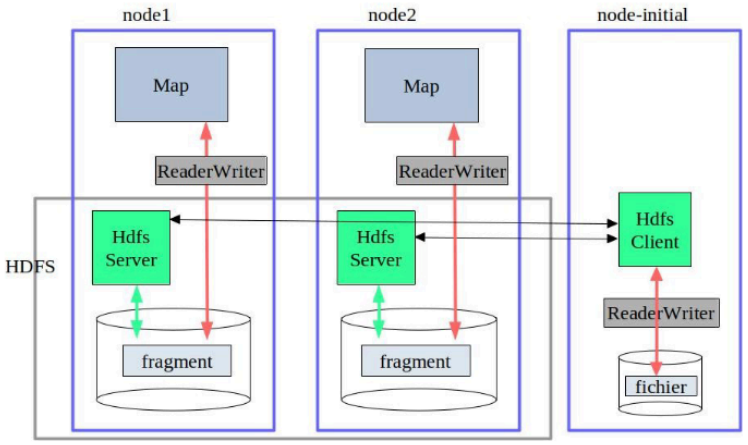

Figure 1 : schéma représentatif du service HDFS

Pour ce faire nous avons réalisé deux classes différentes : une classe HdfsClient, ayant pour objectif d'effectuer des requêtes aux serveurs disponibles ; et une classe HdfsServer, ayant pour but de réaliser les requêtes du client. Ces requêtes sont les suivantes :

- HdfsWrite : à partir d'un fichier texte local (ou un fichier KV), on découpe ce texte en parts égales, dont le nombre de parts correspond au nombre de serveurs disponibles. Chaque serveur reçoit un "morceau" de ce texte, qu'il vient stocker de son côté.
- HdfsRead : fait une requête de lecture aux serveurs en ligne, qui viennent envoyer le texte qu'ils ont stocké, et le client vient concaténer ces fragments de texte afin de retrouver le texte original.
- HdfsDelete : fait la demande aux serveurs de supprimer les fichiers de texte qui contiennent les fragments de texte de leur côté.

II. Mise en place de MapReduce

Dans un second temps, nous avons essayé de mettre en place le service MapReduce, dont l'objectif est de l'appliquer à un fichier texte, et qu'il nous renvoie un fichier KV, dont la clé correspond au nombre d'itération d'un mot, et la valeur le mot en question.

Le fonctionnement de ce dernier réside dans plusieurs étapes consécutives : le lancement du service Hdfs dans un premier temps afin de venir fragmenter le texte et le stocker dans les différents serveurs disponibles. Dans un second temps on vient lancer un worker par serveur, et on exécute JobLauncher. JobLauncher vient lancer simultanément la méthode Map de MyMapReduce sur chacun des workers à l'aide d'une connexion RMI. Les workers sont lancés parallèlement. Enfin, lorsque le mapping est terminé sur tous les workers,

on vient appliquer la méthode reduce (de MyMapReduce également) sur ces workers, qui vient concaténer les fichiers de chaque workers (sous forme de KV), et rassembler les mots semblables en ajoutant leur nombre d'itérations. On obtient ainsi
Pour récupérer le résultat des mappings de chaque worker, on met en place une connexion réseau entre eux et le JobLauncher par l'intermédiaire d'un Adaptateur. Les Workers envoient via write les KV obtenus par le mapping et Adaptateur les reçoit et les enregistre dans une file. Adaptateur implémente reader et dispose d'une méthode read() consistant à lire les éléments de la file et à les retirer uns à uns. Les connexions serveurs sont aussi établies de façon parallèle.

III. Manuel utilisateur

Pour HDFS

Pour lancer le service HDFS, on vient lancer des serveurs soit en local, avec l'adresse localhost, soit sur une machine de l'école avec le nom de la machine en adresse. Côté client, on vient remplir le fichier src/config/adresses.txt de la façon suivante :

adress port server_id

Pour lancer les serveurs :

java -cp bin hdfs.HdfsServer <adress> <port>

Pour lancer le client :

java -cp bin hdfs.HdfsClient read <file>
java -cp bin hdfs.HdfsClient write <file> <txt : 0|kv : 1>
java -cp bin hdfs.HdfsClient delete <file>

Pour MyMapReduce

Pour lancer MyMapReduce, il faut dans un premier temps lancer les serveurs, et utiliser le write de client. On lance ensuite un worker associé à chaque nœud, et on peut alors exécuter MyMapReduce.
Pour ce faire, on exécute la commande suivante :

java -cp bin application.MyMapReduce <file>

On peut ainsi récupérer le fichier réduit dans le dossier data/reduced.