Rapport

Projet - Hidoop

Ignacio Oros Campo

Matthieu Hamel

Thibaud Merieux

Duc-Do Trinh

Adrien Gallego

Table des matières

1 Implantation de HDFS

La classe Hdfs a été implantée le plus simplement possible. La convention de nommage est que tous les fichiers portent le même nom sur chacun de leur serveur, qui est aussi le nom du fichier d'origine. Les serveurs sont ordonnés (les adresses se trouvent dans un tableau tabserveurs) et tabserveurs[i] recueillera le ième fragment.

1.1 HdfsRead

HdfsRead permet de récuperer des fragments d'un fichier de type KV dans les serveurs du système de stockage de données et recomposer dans l'ordre le fichier originel en ordonnant les fragments correctement. Ici, on ordonne les fragments d'un même fichier selon les ports croissants. C'est à dire, le serveur associé au socket dont le port est le plus petit contiendra le premier fragment et le serveur associé au port le plus important le dernier et ainsi de suite. On a implanté la convention de nommer le fichier recomposé dans le client comme chaque fragment sur chaque serveur parce que ici les fragment sont de taille variable, ce qui permet de stocker un seul fragment par serveur.

1.2 HdfsWrite

HdfsWrite permet d'envoyer aux serveurs les différents fragments d'un fichier. L'envoi du i-ième fragment se fait toujours au i-ième serveur.
Afin que le projet soit adaptable à d'autres situations plus souples, les fichiers (LINE y compris) sont stockés sur les serveurs au format KV. Puisque HdfsRead demande les fragments dans l'ordre à chaque serveur, cela n'est pas utile à l'heure actuelle, mais pourrait le devenir dans une situation où l'on aurait besoin de savoir quel fragment se situe où.

1.3 HdfsDelete

Pas de grosse difficulté sur cette partie.

2 Implantation de Hidoop

Pour l'implantation l'essentiel de Hidoop, l'essentiel du travail est dans Job qui récupère et donne du travail aux Workers pour ensuite récupérer les résultats et effectuer le reduce final.

2.1 Worker, WorkerImpl

Worker pourra lancer des maps et manière non bloquante, on utilisera le lancement de threads. Ces derniers lanceront les maps et enverront un signal quand elles finissent via le Callback.

2.2 Job

Job aura comme rôle de lancer tous les maps sur toutes les machines distantes. On a par construction qu'un seul fragment par machine déjà installé au préalable par HDFSWrite. Ainsi Job lance un unique map par machine en utilisant un while et attendra grâce à un sémaphore la terminaison des maps de tous les Workers. Ensuite il reste à réassembler tous les fichiers résultats sur toutes les machines à l'aide de HDFSRead pour y appliquer le reduce final. Une fois le résultat final obtenu on supprime les fichiers intermédiaires inutiles avec HDFSDelete.

3 Tests

3.1 HDFS

Les tests ont été faits manuellement : à la fois en local et avec les machines de l'ENSEEIHT.

3.2 Hidoop

Les tests réalisés sur Hidoop ont été réalisés manuellement. Nous avons effectué le comptage des mots avec l'application Count fournie et avec notre implantation du Map Reduce. Ensuite nous avons comparé les résultats en utilisant la fonction diff. On a ainsi vérifié que les fichiers résultats sont exactement les mêmes et donc que notre implantation de Map Reduce fonctionnait comme attendu.

3.3 Performance

Les tests ont été réalisés en local et non pas avec des machines différentes physiquement. Ainsi les tests de performance ne sont pas cohérents. En effet avec l'optimisation de la JVM, Count effectue le comptage plus rapidement que le Map Reduce en local.