Rapport Hidoop, deuxième partie, point d'avancement 1

February 2021

Contents

List of Figures

1 L'architecture de Hdfs

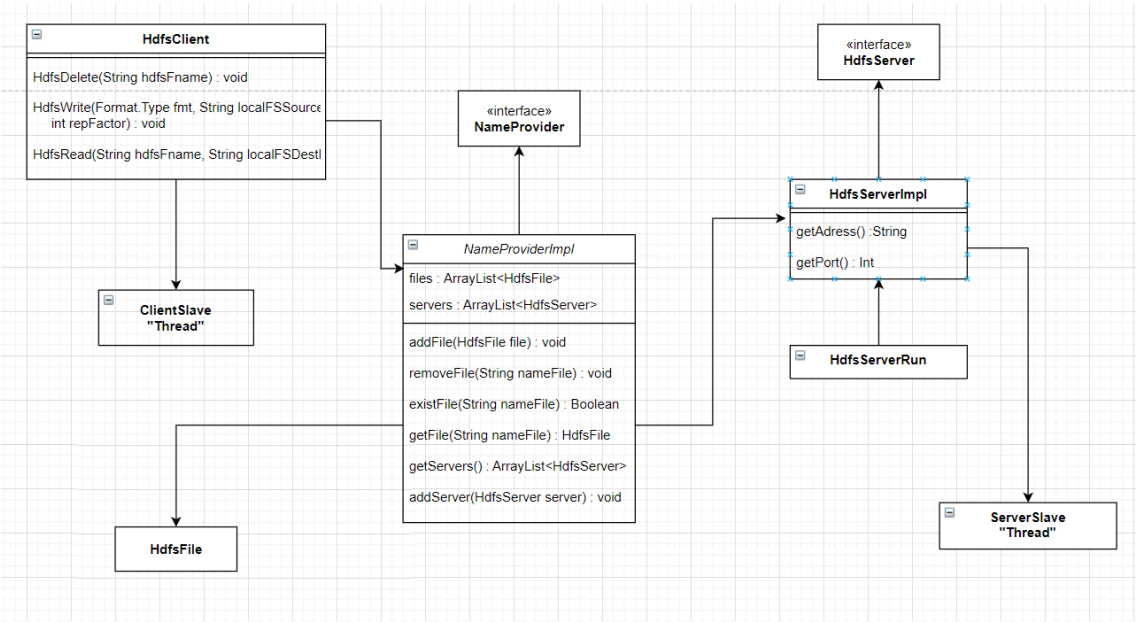

Figure 1: L'architecture de la partie hdfs

- HdfsClient : Permet d'écrire, lire ou effacer des noeuds sur les serveurs
- ClientSlave : Gère les émissons par TCP
- HdfsServeur : Gère un noeud de stockage
- HdfsServeurSlave : Gère les opérations sur les fichiers du noeud
- HdfsIO : Fonctions d'écriture et de lecture sur le réseau
- HdfsFile : Représente un fichier et sa localisation
- NameProvider : Localise les serveurs et les fichiers sur les serveurs

2 L'architecture de Hidoop

- Job : il lance l'application en distribuant les tâches sur les workers et communiquant avec CallBac surt la terlinaison des maps , et finalement, récuperant le résultat de reduce.
- Worker : il lance autant de TaskExecuters que de demandes simultannées de maps

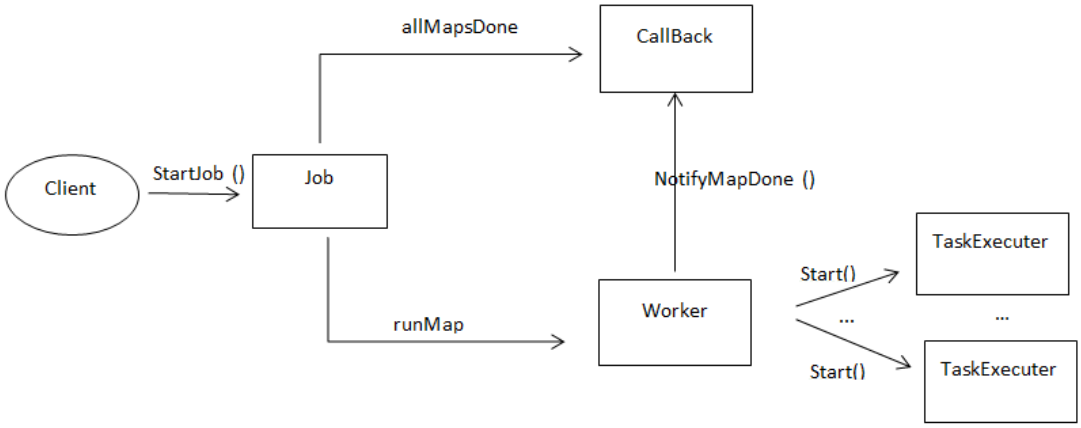

Figure 2: L'architecture de la partie hidoop

- CallBack : il est initialisé par JOb avec le nombre de maps lancés, chaque Worker le notifie à la fin d'un map.
- TaskExecutor : c'est un Thread qui est lancé par un worker, il exécute un map.

Remarque : Cette partie fait appel également à HdfsRead(..) de la partie Hdfs pour récupérer le résultat du reduce localement.

3 La synthèse des corrections/modifications apportées à la version originale de Hidoop

Dans le but de produire une premi‘ere version r´epartie et op´erationnelle et qui donne des r´esultats coh´erents avec ce qui est attendu en terme de performance, on a utiliser un buffer pour que les opérations read et write de la partie Hdfs soient plus rapide. On aussi affecter à chaque serveur un identifiant unique qui va nous aider après à detecter les serveurs tombés en panne et rectifier l'erreur. Et dans le but de vérifier les fichiers générés on a ajouté une fonctionnalité qui verifie que les fichiers sont exactement les mêmes.

4 Les tests de vérification effectués

Nous avons testé notre programme avec l'application WordCount pré-définie et nous avons comparé les résultats avec l'application Count qui s'exécute en séquentiel. Les résultats des tests ainsi que les remarques faites sont présentés dans la partie d'étude de scalabilité sur l'application WordCount.

5 Les outils développés pour faciliter le déploiement

- Configuration:
  Les noms des machines servers set les ports des workers ont déclarés dans `src/config/conf.txt` .
  Le nom de machine et port sous lequel le nameProvider est lancé, est reseigné dans le fichier `src/config/hdfs.json`

- Lancement:
  Pour lancer une application sur la platforme hidoop, il faut lancer le script `lanceur.sh`. Vous pouvez visualiser l'output dans le fichier généré `nohup.out`, il contient ce qui a été affiché chez le nameProvider, serveurs, et workers.
  Après la récupération du résultat de l'application il faut nécessairement lancer le script `arret.sh` pour arrêter tous les processus qui tournent toujours (nameProvider et tous les serveurs et workers).

6 L'étude de scalabilité sur l'application WordCount

Pour un même fichier de taille 2,7 Go, nous avons lancé l'application WordCount en variant le nombre de nameNodes à chaque exécution.
La figure suivante montre la variation du temps de calcul en fonction du nombre de name nodes dans le cas itératif et map-reduce. On voit bien que pour un grand fichier, plus les noeuds sont nombreux, plus le temps de calcul diminue.

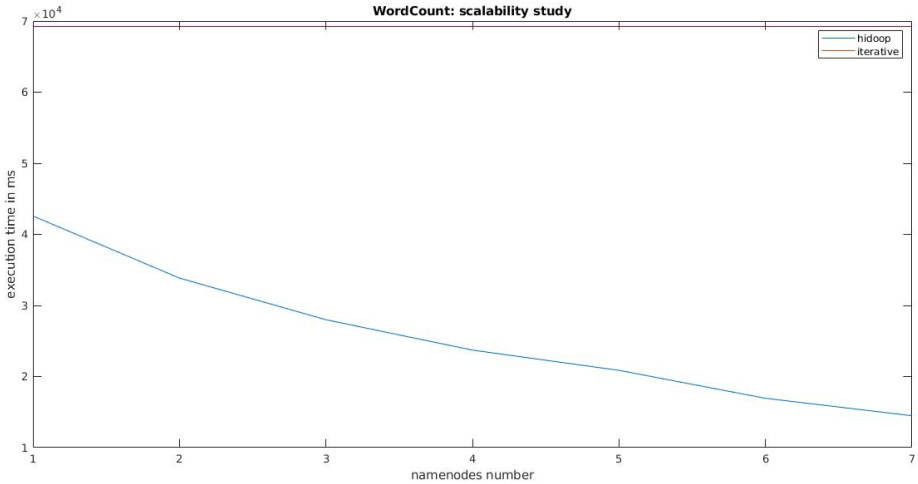

Figure 3: Graphe résultant de l'etude de scalabilité

7 Les améliorations envisagées

On envisage améliorer la tolérance aux pannes.

8 l'application choisie pour évaluerla future version améliorée

On envisage étudier l'application du pageRanking, vue qu'elle est l'une des applications classiques de hadoop, donc on veut bien la tester avec notre version simplifiée de hadoop.