Hidoop

BONNET Olivier, STEUX Yoann, XAMBILI Robin

***Résumé*—Explication du fonctionnement et de l'implantation de Hidoop et des technologies utilisées.**

I. Introduction

HIDOOP est une plateforme dont le principe est de traiter une grande quantité de données en même temps en utilisant pleinement toutes les ressources possibles. Le travail est séparé sur plusieurs processeurs en même temps.
Le travail réalisé ne concerne que la partie sur la répartition des data.

II. Structure générale

Le client va envoyer la fonction à appliquer sur chaque partie de texte à chaque daemon présent sur des serveurs qui s'occupent d'une partie de texte. On a utilisé pour cela la technologie RMI.

A. Client

La classe Job implémente JobInterface. Il représente le « client ».
La fonction startJob demande à chaque Daemon des machines présentes dans le fichier de configuration de lancer le runMap, en leur donnant le nom du fichier sur lequel lire, celui sur lequel écrire, ainsi que la fonction à exécuter.
On transfert également un objet CallBack qui permet de transférer des informations telles que la fin du map. Dans cette V_0, le CallBack ne sert pas réellement ; on peut facilement récupérer si le map est fini, cela servira dans la V_1 quand nous aurons dupliqué les fragments de fichier (au cas où il y aurait une panne). On applique ensuite la fonction reduce présente dans l'objet MapReduce / Mapper qui recalcule le résultat en utilisant tous les « bouts » de résultats qui ont été récupérés par Hdfs.

B. Serveurs

La classe DaemonImpl implémente Daemon. Les Daemon sont lancés sur les serveurs.
Le main crée un registry sur le serveur. Il y aura donc autant de registry qu'il y a de serveurs.
La fonction runMap lancée par Job ouvre les fichiers présents dans les FormatReader et FormatWritter passés en arguments. La fonction MapReduce passée en argument est executée avec les fichiers précédemment ouverts.

C. Formats

La classe FormatImpl implémente Format, FormatWritter et FormatReader.
Les fonctions open et read lisent et écrivent dans les fichiers, en respectant les formats KV ou LINE.

III. Comment tester

A. Déployer les Daemons

On peut ajouter d'autres Daemons dans le fichier daemons.txt.
Il faut se placer en local dans `/hidoop/src`, et entrer :
`../config/launch.sh`

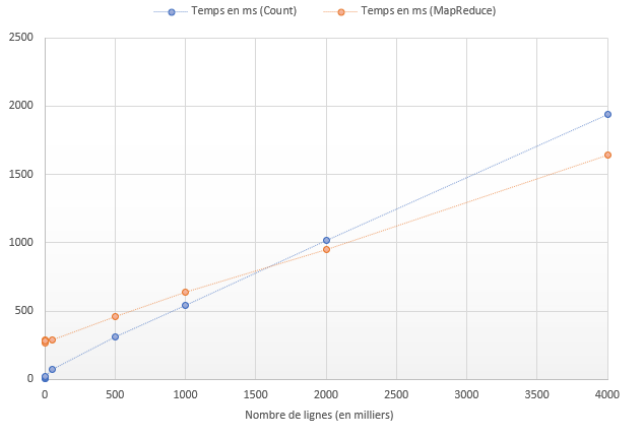
B. Lancer l'application

Dans le terminal client, il faut se placer dans `/hidoop/src`, et entrer :
`java application.MyMapReduce ../data/filesample.txt`
(si l'on souhaite appliquer le MyMapReduce au fichier `filesample.txt`).

IV. Tests de complexité

Pour tester cette partie de Hidoop, nous avons créé 9 fichiers de tailles différentes.

On a mesuré les temps d'exécution de Count et de MyMapReduce, pour chacun des 9 fichiers (ayant entre 40 et 4'000'000 lignes), et obtenu le graphe suivant :



On remarque ici qu'il est préférable d'utiliser MyMapReduce plutôt que Count pour des « gros » fichiers (de nombre de lignes supérieur à 1'600'000 environ). Ainsi, l'utilisation de MyMapReduce est pertinente au vu du traitement de fichiers de grandes tailles.

V. Conclusion

Pour la prochaine version, l'objectif est de rendre plus robustes les délégations de calcul en dupliquant les différentes parties de texte…