MS Big Data

INF727

Rapport du projet SHAVADOOP

Encadrés par:
Rémi Sharrock

Réalisé par :
Sami Bargaoui
Mohammed Benseddik

Contents

1 Contexte du projet

Le but du projet est d'implémenter une fonction de Wordcount en MapReduce, analoguement à la fonction implémentée sur Hadoop. La fonction Wordcount comptera le nombre d'occurences de chaque mot présent dans un fichier texte donné en entrée, l'approche MapReduce pour le Wordcount répartira les opérations (Splitting, Mapping, Reducing, Shuffling, ...) de manière distributée sur un ensemble de machines connectées.

L'approche MapReduce consiste à cet ensemble de traitements :

- **Une phase de Mapping :** Chaque machine appelle la fonction Map() sur le jeu de données qu'elle a reçu. La fonction Map() calculera le nombre d'occurences de chaque mot présent dans les données.
- **Une phase de Shuffling :** Chaque machine redistribue les données à partir du résultat de l'étape de Mapping. On peut par exemple rassembler un ensemble de mots ou de clés sur une machine en particulier, de façon à répartir l'ensemble des clés sur toutes les machines.
- **Une phase de Reducing :** Chaque machine ayant reçu un ensemble de clés chacune ayant une certaine valeur, appliquer la fonction Reduce() consiste à rassembler toutes les clés et à faire la somme de toutes les valeurs associées, on aura donc un ensemble de clés (les mots présents dans le fichier en entrée), auxquelles on associe des valeurs (le nombre d'occurences de chaque mot dans le fichier), d'où le principe du MapReduce.

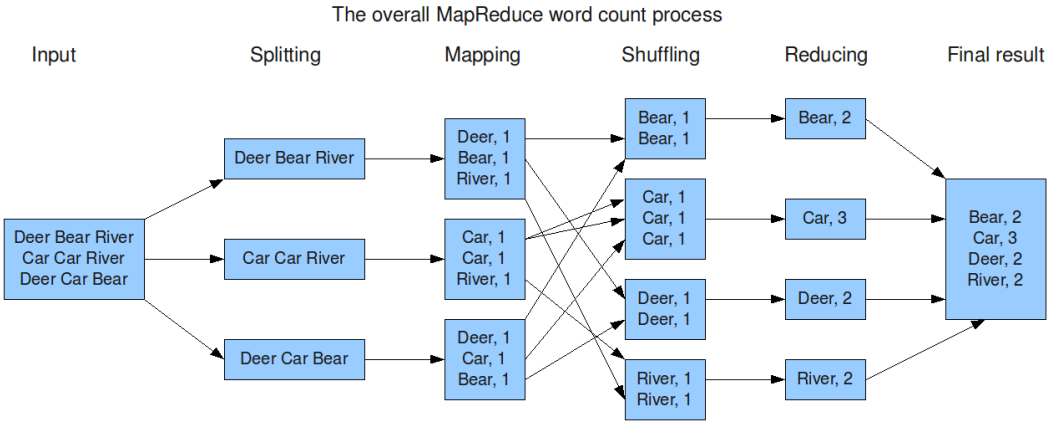

Figure 1: Schema MapReduce

Cette opération fait intervenir un master et plusieurs slaves ou workers. Le rôle du master est de surveiller toutes les opérations et lancer sur les workers les opérations requises, tandis que le rôle des workers est d'éxécuter les opérations demandées par le master, toute la communication entre les différentes parties se fera en SSH.

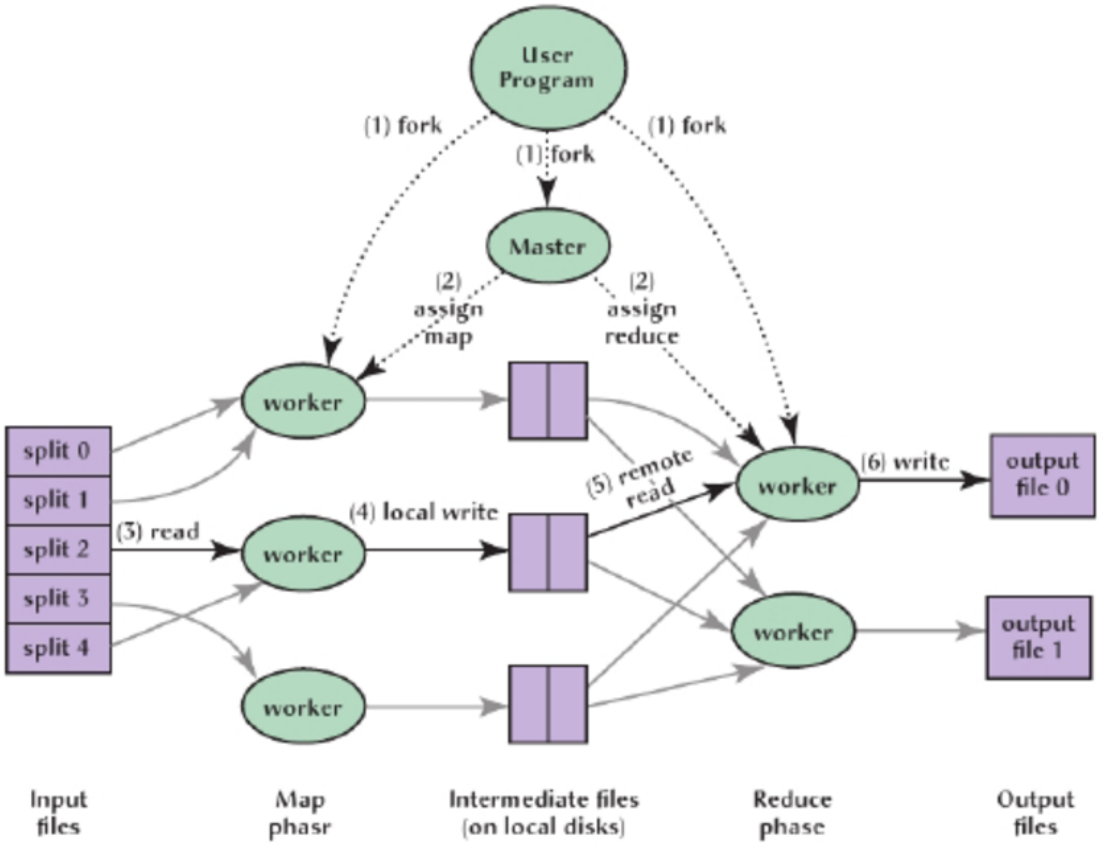

Figure 2: Schema MapReduce

2 Organisation du code

Le programme est organisé en deux principales parties :

- Les traitements du Master :
  - CheckPcsThread : Contient les fonctions liées à la connection du master aux différentes machines reliées au réseau.

  - Master : Contient les procédures et fonctions liées au traitement du master (Création des dictionnaires, Splitting, Merging, ...)
  - ShavadoopThread : gère l'éxécution des threads sur les différents slaves, fonctionne en deux modes : SxUMx (Mapping) et UmxSmx (Reducing).
  - ShavadoopUtils : Contient des sous-fonctions utilisées par les méthodes invoquées dans le master.

- Les traitements du Slave :
  - Slave : exécute les fonctions du mapping et reducing sur les différentes machines.

3 Fonctionnement du code

3.1 Fonctionnement du master

Le master fait appel principalement à trois grandes fonctions, qui font appel à des sous fonctions. D'abord le master opère le split sur le fichier donné en entré et crée les fichiers Sx à partir de la procédure splitSxBloc, qui recoit en entrée le fichier à splitter et un nombre de lignes choisi par l'utilisateur : si par exemple le fichier contient 2000 lignes et que l'utilisateur donne en paramètre une division chaque les 100 lignes, on aura 20 fichiers Sx qui correspondent à 20 partitions du fichier, le résultat renvoyé est un dictionnaire des fichiers Sx.

Puis on opère le MapReduce sur les slaves à travers la fonction runOnSlaves : on recoit le dictionnaire de fichiers de la phase précédente, on établtit une liste de machines sur lesquelles lancer notre traitement, on opère la phase de Mapping à travers la méthode splitOnUmx qui retrourne un dictionnaire, on passe à la phase de shuffling et on termine par lancer le Reducing sur le dictionnaire obtenu en faisant appel à la méthode splitOnSmXAndRmx. On sauvegarde après le fichier obtenu où on retrouvera les mots du fichier et leur occurence.

La classe Master fait appel à des méthodes intermédiaires qu'on retrouve sur dans ShavadoopUtils. Cette classe de méthodes regroupe les traitements utiles au lancement du programme comme la suppression des fichiers crées lors de lancements du code antérieurs, des procédures d'affichage, les fonctions qui interviennent dans le split du fichier en plusieurs sous fichiers, etc...

La classe ShavadoopThread est la classe qui lance et gère les traitements sur les différents slaves et fonctionne en deux modes distincts : mapping pour traiter la

phase de mapping et reducing pour traiter de la phase du même nom. Elle contient aussi des méthodes qui gèrent l'exécution de tâches sur des machines distantes.

3.2 Fonctionnement du slave

La classe Slave contient des méthodes d'éxécution des tâches lancées par le main. Parmi les méthodes implémentées dans le Slave, on retrouve des traitements sur les fichiers : éliminations de caractères spéciaux, éliminations des mots de liaisons et des conjonctions, etc.

On retrouve aussi les méthodes liées à la phase de Shuffling, Reducing et gestion des traitements sur les fichiers intermédiaires et dictionnaires (Um, Sm, Rm). La méthode main quant à elle est la méthode où on éxécute les méthodes citées au-dessus sur les différentes machines de la liste pré-établie.

4 Lancer le code

Le code ne fait pas appel à des librairies extérieures. Pour lancer le code, il importer les fichiers sources dans Eclipse. Il faut ensuite changer le répertoire qui contient les fichiers en entrée dans le Slave (SlaveShavadoop) et paramétrer le bon chemin pour les fichiers nécessaires. Le lancement se fait à partir du Master (MasterShavadoop). En sortie on verra l'éxécution des différentes étapes, ainsi que les temps impartis pour chaque étape du traitement.

5 Problèmes confrontés et Optimisations

Lors de la réalisation du projet, on a été confrontés à quelques problèmes. Un exemple a été les délais de connexions SSH et l'éventualité de threads qui ne répondaient plus causait qu'on perdait des données en les envoyant à des slaves qui ne répondaient plus, donc on aboutissait à un fichier après traitement où il nous manquait plusieurs clés/valeurs. Pour remédier à ce problème, on a pensé à relancer les threads morts après le premier lancement de threads, et en relancant à chaque fois, on s'assure de ne pas avoir perdu de données en cours de traitement, et que tous les fichiers soient répartis et traités. Aussi, une recherche de machines sur le réseau est effectuée au début du processus puis stockée sous forme de liste. Cette recherche est unique, il n'y a donc pas de vérification par la suite, ce qui pose le problème des threads perdus et des longs délais de timeout qui ralentissent l'ensemble de l'exécution.

6 Github, Javadoc

Le projet est disponible sous Github : https://github.com/BenseddikM/Shavadoop ou https://github.com/Sbargaoui/Shavadoop . Dans le projet aussi, on a créée une Javadoc pour toutes les classes et les méthodes implémentées. La javadoc se trouve sur ce siteweb créée pour le projet sous Github :
https://sbargaoui.github.io/Shavadoop/docs/