TP5
Hadoop

GIF 4104
Programmation parallèle et distribuée

Pier-Luc Auger (PLAUG2 - 910 098 011)
Mathieu Garon (MAGAR220 - 111 005 164)
Colosse USER46

Université Laval

Rapport TP5

Mise en contexte

Pour ce TP, on demandait d'écrire un programme Hadoop permettant de traiter l'ensemble des documents contenus dans un répertoire et de construire un index inversé des termes rencontrés dans ces documents. Le programme créé accepte en entrée deux arguments sur la ligne de commande: 1) le chemin du répertoire contenant les documents et 2) le nom du fichier d'index à créer.

Méthode utilisée

Dans ce TP, les librairies Hadoop distribuées par Apache ont été utilisées. Celle-ci permettent de formaliser l'algorithme de Map-Reduce pour résoudre le problème. Essentiellement, l'algorithme consiste en deux principales parties distinctes:

Mapper

Le Mapper est en soit le premier élément qui effectue un traitement sur les données. Il associe des paires clé/valeur en un ensemble de clé/valeur intermédiaire. De manière générale, la clé est "l'offset" de la valeur lue dans un fichier. Dans le programme développé, une structure incluant l'indice du fichier et l'offset a été utilisée. Cela donne des ensembles [MOTS (ndf offset)].

Reducer

Une instance de Reducer est créer pour chaque tâche. C'est la méthode reduce qui est appelée pour effectuer le traitement. Cette étape permet de rassembler les éléments avec la même clé, ici le même mot, et de reformer un ensemble ordonné. En effet, à ce stade, l'ordre n'est pas encore valide. À la sortie du traitement, on remarque que toutes les données ont été indexées mais aussi triées.

Formatage

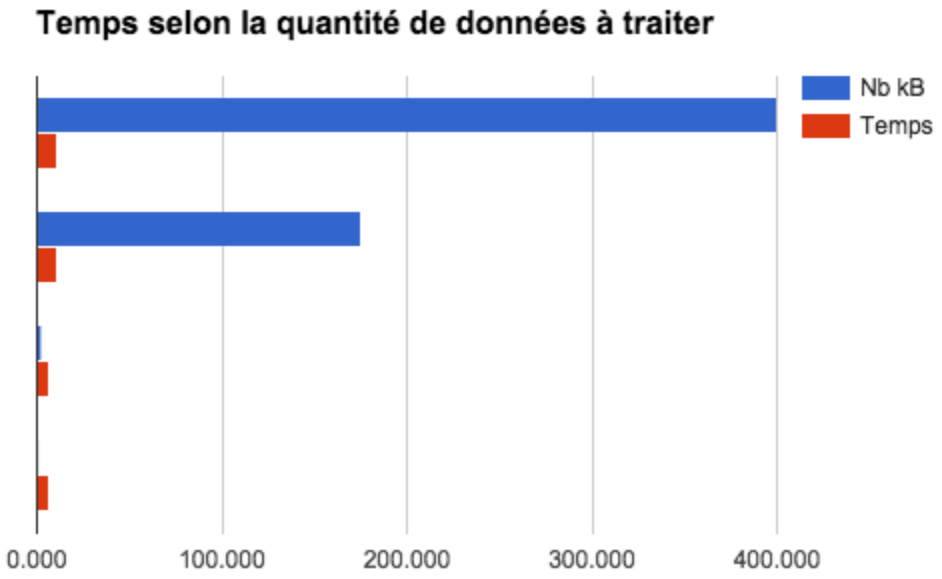
Enfin, un script Python complète le formatage du fichier de sortie pour le rendre conforme aux exigences. Celui ajoute entre autre la liste des fichiers parcourus en entête du fichier de résultats. Celui est à l'emplacement `/results/indexInverse.txt`.

Analyse des résultats

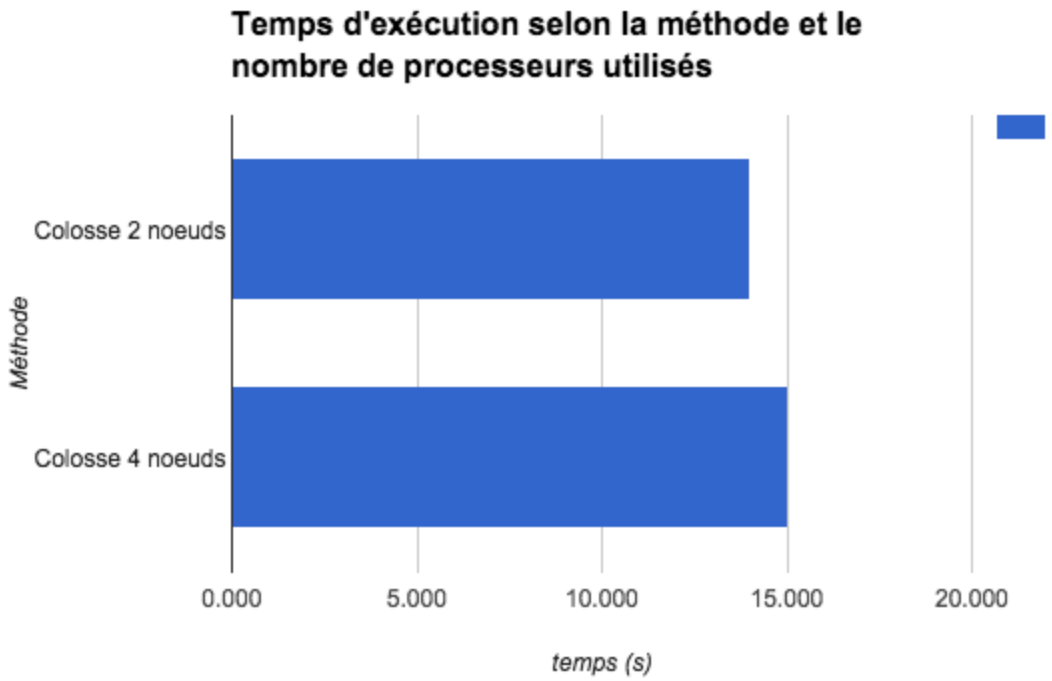
Nous devions nous attendre a ce que l'utilisation de Colosse ne permettent pas nécessairement un speedup considérable. En effet, Hadoop est conçu pour traiter des données en quantité massive. Par quantité massive, on réfère à un autre de grandeur tout autre que celui du présent contexte. De plus, l'infrastructure de Colosse n'est pas non plus prévu pour être optimale dans ce genre de contexte.

Ceci dit, voici les résultats obtenus avec le jeu de données présent sous `/user46/tp5/wordcount/input`.

En traitement local, il a été facile de l'observer. En effet, on voit que le temps de traitement est principalement affecté par les overheads qu'implique Hadoop. En effet, le temps de traitement n'est absolument pas linéaire avec le quantité de données à traiter.



Sur Colosse, il a été possible de voir si l'utilisation de plusieurs noeuds permettrait d'accélérer le traitement. Après essais, il est facile de percevoir que cela n'est pas en utilisant plus d'infrastructure matérielle que l'on accélère le traitement.



Lancement de l'exécutable

Sur colosse, le code est déjà disponible et exécutable. Le script à l'emplacement `/home/user46/tp5/wordcount/clean_hdfs_and_submit.sh` permet de lancer le traitement.

À noter qu'il faut exécuter le formatage manuellement. Nous avons tenté de mettre en place l'exécution automatique d'un script de post traitement avec magpie. Hors, les résultats se sont avérés inconstants.

Pour ce formater manuellement, il faut simplement appeler le script `/home/user46/tp5/wordcount/scripts/post.sh`