Rapport final Hidoop

(Arthur Jaumet, Adrien Le Roux, Jiayu Luo, Ai Li)

Améliorations / Changements

Nous avons ajouté la possibilité de supprimer un fichier de HDFS (la méthode delete() de HdfsClient).

Nous n'avons pas fait de changement sur l'architecture de Hidoop car les premiers

Outils

Dans le répertoire principal, nous avons deux scripts qui servent au déploiement d'Hidoop et un fichier de configuration (config.sh) dans lequel il y a les différentes machines sur lesquelles Hidoop sera déployé. RM doit avoir la même valeur que la constante *RESOURCE_MANAGER_HOSTNAME* dans le fichier src/config/Project.java (actuellement "dragon").

Une fois que l'on peut se connecter en SSH sur les machines de l'école sans mot de passe, il suffit de lancer le script "starthidoop.sh" qui se charge de compiler le code et de déployer sur les différentes machines Hidoop.

Ensuite pour arrêter Hidoop il suffit de lancer le script "stophidoop.sh" qui stop tout les processus lancé par "starthidoop.sh" et qui suppriment tout les fichiers présents sur les machines des datanodes.

Pour lancer une application, il suffit de la lancer comme src/application/MyMapReduce.java .

Quant aux tests, nous avons mis en place une classe java pour les réaliser src/tests/PerformancesTest.java, cependant c'est très lourd et peu flexible.

Applications Hidoop

Pour réaliser les tests pour Hidoop, nous avons conçu deux applications en plus de l'application WordCount. Nous avons réalisé les applications PageRank et Monte Carlo.
L'application PageRank sert à calculer les pageranks d'un ensemble de pages. Elle calcule les pageranks de ces pages ainsi que des liens présents sur la page. L'utilité de Hidoop est de diviser le nombre d'url à traiter en les partageant entre toutes les nodes. Ainsi cela permet d'avoir une plus grande efficacité de calcul. Le map s'occupe de recueillir ces URL ainsi que les liens de ces URL afin de les mettre sous forme de données traitables pour le Reduce. La méthode Reduce quant-à-elle s'occupe de calculer le PageRank de chaque URL données par le Map. Le Reduce donne des données traitables par le Map afin de réitérer le MapReduce plusieurs fois et affiner les valeurs des pageranks.

L’application MonteCarlo s’occupe, quant à elle, d’approximer la valeur de Pi. L’algorithme fait apparaître des points au hasard dans le pavé [0,1]x[0,1] et vérifie s'ils appartiennent au disque de centre (0,0) et de rayon 1. Cela permet d’avoir une approximation de Pi puisque la probabilité que le point soit dans le quart de disque est de PI/4. Donc avec un grand nombre de points et en ayant le taux de point dans le quart de disque on peut connaître PI en multipliant le résultat par 4. Le Map s’occupe de voir si les points préalablement créés sont dans le disque et calcule le nombre de points à l’intérieur et à l’extérieur du disque. Ensuite, le Reduce récupère les nombres en sortie des Maps et les additionne. A la fin, il peut réaliser l’approximation de Pi en calculant : 4x(nombrePointsDedans/nombrePointsTotal);

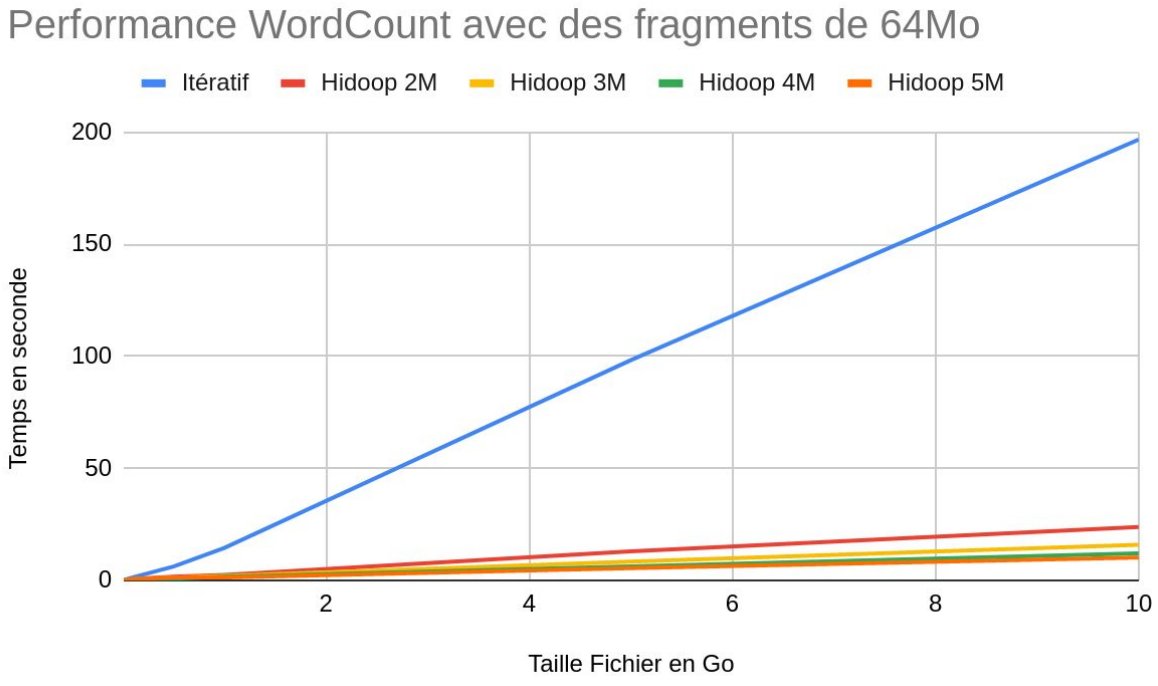
Performances

L’étude de performances se place au coeur du projet. Pour mettre en évidence les performances nous allons nous intéresser au temps pour exécuter les différentes applications. La scalabilité, c’est à dire le fait d’augmenter les performances en augmentant le nombre de machines est l’essence même d’Hidoop, nous devons donc la tester.

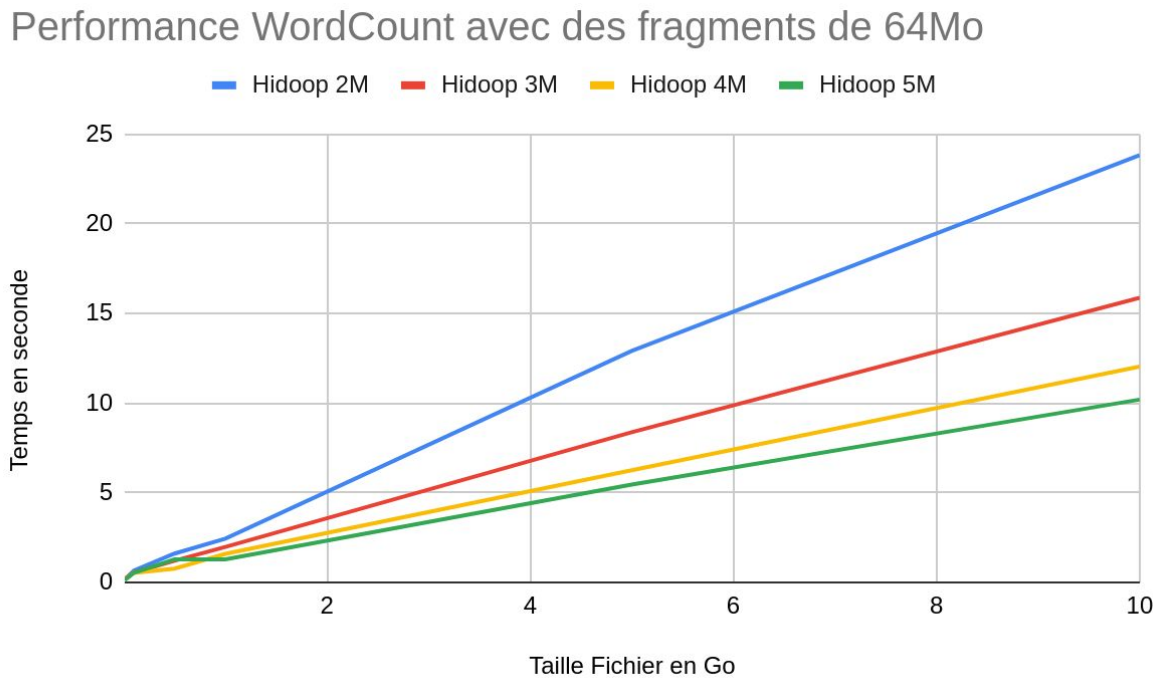
Word Count

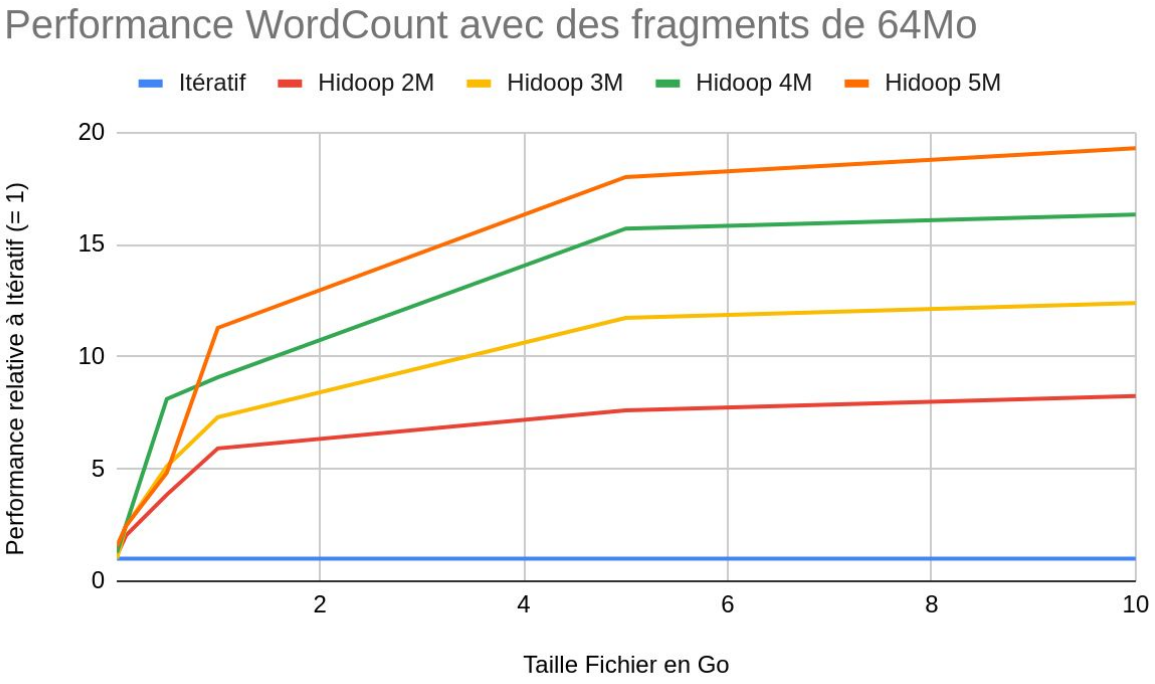
Pour étudier la scalabilité nous allons effectuer l’algorithme WordCount en itératif puis sur Hidoop avec un nombre de machines variables (de 2 à 5) et des fichiers de tailles différentes (de 10Mo à 10Go).

On peut voir déjà qu’avec 2 machines le gain de performances est énorme.



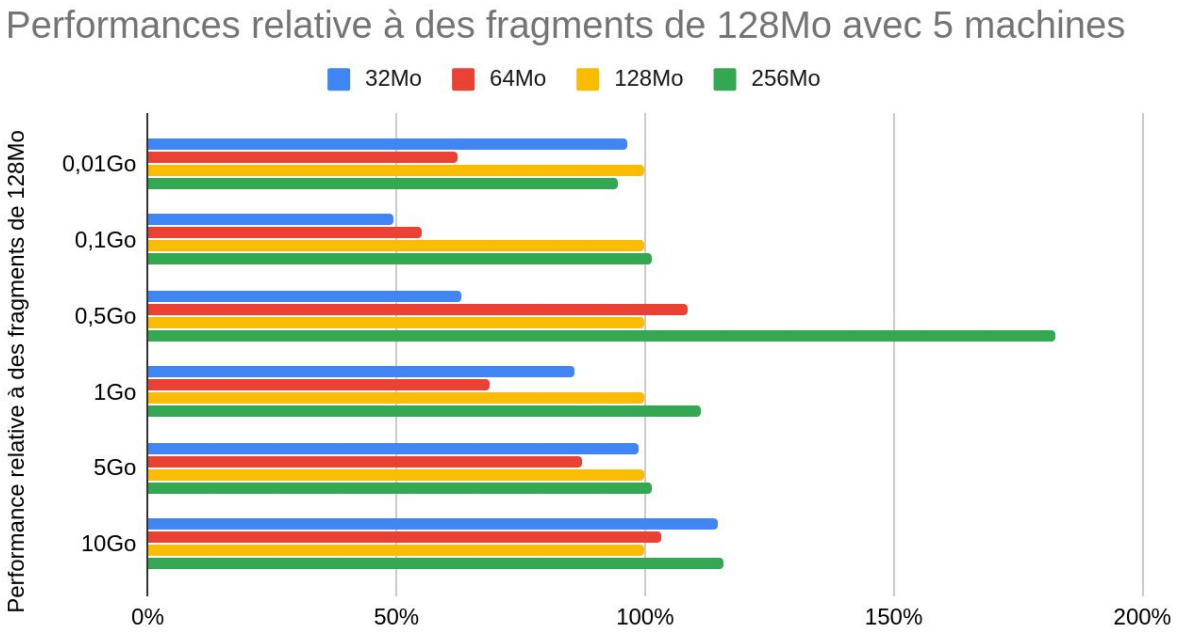
(Zoom sur le temps d'exécution de WordCount)





On peut voir que les temps d'exécution sont linéaires et qu'avec l'utilisation d'Hidoop, même avec seulement 2 machines, on a des performances bien meilleures. Cependant si on prend les performances relative par rapport à l'algorithme itératif, on peut voir que pour des fichiers lourds (supérieur à 1Go) l'augmentation du nombre de machines à un impact

significatif sur les performances. A 10Go, on gagne 33% de performances en passant de 2 à 3 machines, puis 24% de 3 à 4 et 15% de 4 à 5. On voit qu'il y a une limite quand à la pertinence d'augmenter le nombre de machines. D'ailleurs moins le fichier est gros, plus l'écart de performance est petit. Nous allons voir si cela n'est pas dû à la taille des fragments dans HDFS.



On peut voir que globalement augmenter la taille des fragments augmentent drastiquement les performances pour des tailles de fichiers relativement faible tandis que pour des tailles de fichiers très élevés l'écart se réduit.