Hidoop

Point d'avancement 1

Rapport du groupe Guillaume Claverie, Corentin Dominguez, Théo Petit, Gautier Ben Aïm

Synthèse des corrections / modifications

Le système HDFS supporte des download et des uploads sans limite de taille et à des débits conséquents : jusqu'à 550 Mbps en upload et 370 Mbps en download (tests effectués sur des fichiers de 10 Go).

Cela a été rendu possible par une réécriture complète du fonctionnement de l'upload :

- Avant : envoi d'un buffer de 65 000 KV par des *ObjectStream*s
- Après : envoi d'un buffer de 4 millions d'octets par des *BufferedStream*s

Grâce à cette modification, la mémoire n'est plus saturée par la mise en cache effectuée par les *ObjectStream*s et le débit est multiplié par 100.

Par ailleurs, de nombreuses petites corrections de stabilité ont été faites, comme la suppression de *race conditions* sur les nœuds de données.

Tests effectués (correction & passage à l'échelle)

Sur les fichiers lorem ipsum (375Mo, 750Mo, 1500Mo, 3000Mo) : on a vérifié que le fichier résultat en séquentiel et en hidoop sont identiques (taille exactement identique, et compte de mots identiques).

De même, nous avons testé sur d'autres fichiers textes générés aléatoirement et le résultat est conforme à la version séquentielle. Enfin, lorsque nous remplissons un fichier d'un seul et unique mot, nous avons bien le bon compte pour ce mot dans le fichier résultat.

Outils développés pour le déploiement et tests

TODO : rediger

- Script de déploiement : pas essentiel mais permet de mettre instantanément le dossier Hidoop sur les disques dur (répertoire /work)
- Script de lancement (à terminer → cd puis executer commande pas possible & problème commande pas possible partout)
- Parler du fait que l'outil doit modif 2 3 trucs dans le script
- exec crée nameserver et rmi server (contient liste des serveurs distants) donc doit être la machine principale

Nous avons développé deux scripts permettant de déployer notre application. Les deux scripts sont situés dans le répertoire src/application du dossier Hidoop.

Le premier script, *deploy.sh,* permet de placer le dossier Hidoop sur les disques durs (dans le répertoire /work) des machines qui vont être utilisées pour HDFS.

Le deuxième script, *exec.sh*, doit être lancé sur la machine principale. En effet, il va permettre de lancer, sur la machine d'exécution du script, le NameServer ainsi que le RmiRegistry. Le script va ensuite lancer les Binodes (HDFSnode et Worker) sur chaque machine du HDFS. Enfin, le script permet aussi de stopper le RmiRegistry et les Binodes si l'utilisateur appuie sur la touche `!`.

Le dernier script, *force_stop.sh*, est présent en cas de soucis si exec.sh n'arrête pas Hidoop (à cause d'un soucis).

Comment utiliser les scripts

Les scripts se trouvent dans le dossier Hidoop qui contient les fichiers compilés.

- Placer ce dossier sur une machine de l'ENSEEIHT.
- Dans **deploy.sh**, mettre dans la variable HOST la liste des machines sur lequelles vous voulez déployer. Puis executer le script deploy (./deploy.sh)
- De la même manière, lister les HOST dans le script **exec.sh**, puis executer le script(./exec.sh)
- Vous pouvez maintenant utiliser une application telle que MyMapReduce en executant : "java application.MyMapReduce <nomDuFichier>" ou uploader un fichier sur le HDFS avec "java hdfs.HdfsClient write line <cheminDuFichier>"
- Lorsque vous avez fini, appuyez sur une touche pour arrêter le script exec, puis lancer le script **force_stop.sh** (en renseignant la liste dans HOST des machines sur lesquelles les binodes se trouvent)

Etude de scalabilité sur WordCount

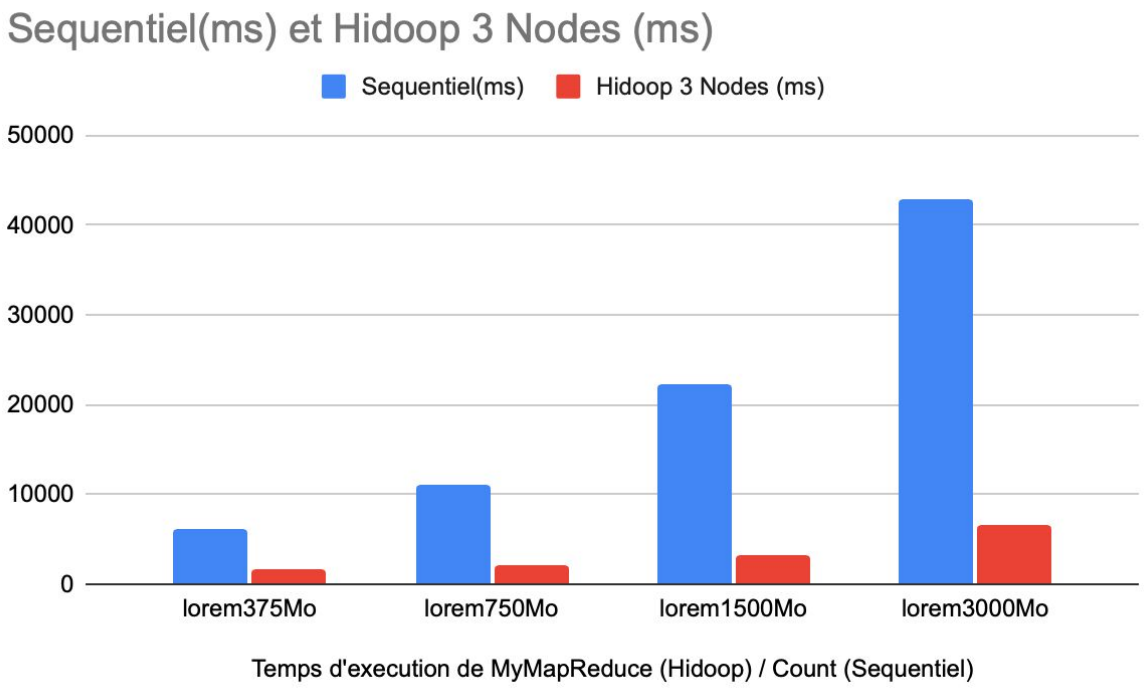
Les performances ont été mesurées en utilisant la commande *time* de Linux (et comparées aux valeurs qui sont print par le programme Java).

Paramètres pouvant influencer les performances

- Le processeur de la machine : nos nodes vont utiliser un thread par fragment à traiter. Généralement, une machine a bien plus de fragments que de coeurs, donc ils seront tous exploités. Ainsi selon les machines utilisées, pour un même nombre de node, on peut avoir des performances différentes.
- Le nombre de node… (qui est corrélé au nombre de processeurs disponibles)
- La taille du fichier

Comparaison des performances

Tout d'abord, on teste avec différentes tailles de fichier, en séquentiel, puis avec un Hidoop de 3 nodes.

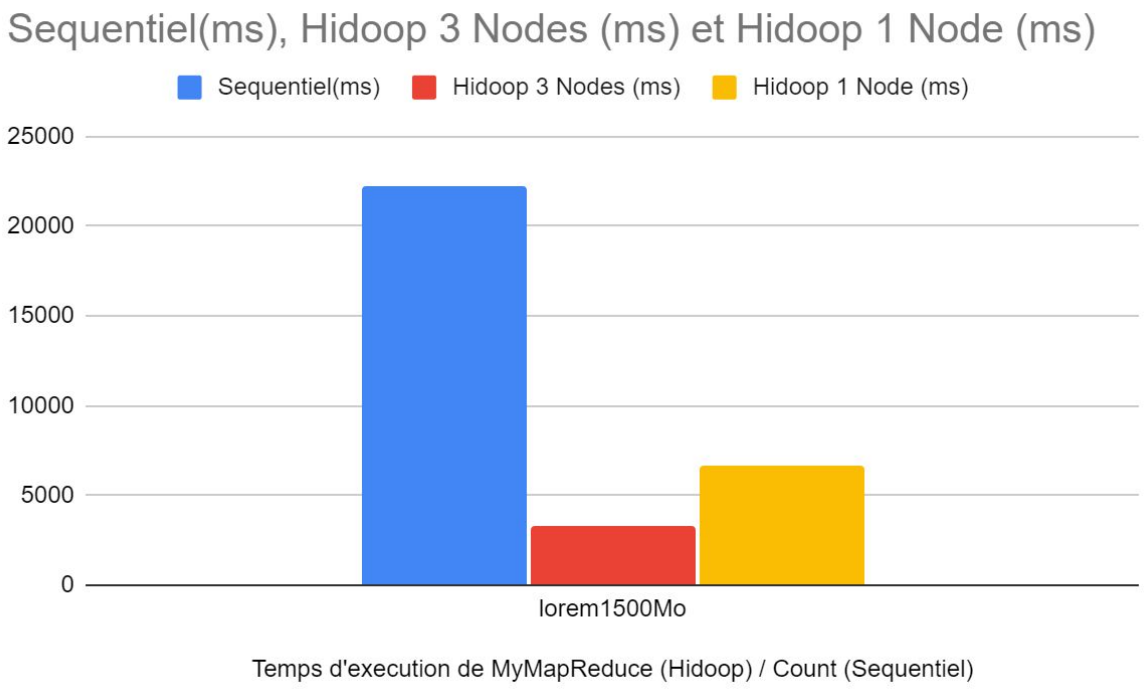


Données brutes obtenues

Calculs	Sequentiel(ms)	Hidoop 3 Nodes (ms)
lorem375Mo	6068	1585
lorem750Mo	11113	2166
lorem1500Mo	22231	3315
lorem3000Mo	42867	6610

Puis on teste le fichier de 1500 Mo en séquentiel (42 secondes), puis avec 1 node (7 secondes) et 3 nodes (1,5 secondes). On remarque qu'il n'y a pas énormément de gain à augmenter le nombre de nodes pour cette taille de fichier et cette application (WordCount).

Pour 3000 Mo, le temps a environ doublé face au 1500 Mo.



Temps d'execution de MyMapReduce (Hidoop) / Count (Sequentiel)

Analyse et interprétation

On remarque que même avec un seul node, le temps est bien meilleur qu'en séquentiel. Cela pourrait être dû au fait que l'on exploite tous les cœurs de la machine (et la version séquentielle non ?).

Les performances de Hadoop sont donc très satisfaisantes, cependant ce test ne prend pas en compte le temps d'upload des fichiers (env 1min30 pour le fichier de 1500 Mo !) (Remarque : Suite à une correction, les temps d'upload sont bien meilleurs : 40 secondes pour 1500 Mo).

A travers le premier test, on remarque que le temps de calcul de la version Hidoop ne croit pas de manière proportionnelle à la taille du fichier comme le fait la version séquentielle. Cela nous indique que nous avons certaines étapes qui « bottleneck » les performances de notre version (temps transfert résultats ? Répartition des tâches entre les machines ?).

Remarque : après avoir corrigé un souci qui nous empêchait de tester sur des fichiers plus importants, on remarque que le temps pour un fichier de 3GB est environ le double que pour un fichier de 1.5GB : on est de nouveau sur un rapport proportionel.

Améliorations Envisagées (pour améliorer perfs)

Transférer directement les résultats depuis les Map vers le Reducer. Problème : Si les données des Map sont très grandes (ordre du Giga), on est limités par la mémoire de la VM de Java ?

Pour améliorer les performances de Hidoop, nous pourrions ne pas utiliser de fichier intermédiaire pour le reduce, ou implémenter un système à plusieurs réducer avant de transférer les résultats partiels (car un node a plusieurs fragment de résultat en général.

Pour la fiabilité de Hidoop, nous allons finir de gérer la redondance des fragments (c'est déjà géré mais nous l'avions désactivé dans la V1 suite à un soucis).

Pour la vitesse : quand un fragment est redondant, utiliser le premier node qui a fini son boulot pour calculer sur ce fragment (attention à ne pas dupliquer du travail!)

Pour la fiabilité : si un node plante pendant un map, il faut qu'un autre node prenne le relais (cf point juste au dessus)

Application(s) choisie(s) pour évaluer la future version améliorée

Nous avons choisi de réaliser principalement deux applications de Map/Reduce pour compléter celle du WordCount.

Dans un premier temps, nous voulons implémenter l'algorithme de Monte-Carlo, méthode pour calculer pi. Cette application nous permettra de vérifier que notre Hidoop ne fonctionne pas simplement sur le WordCount mais que l'on peut l'utiliser pour d'autres problèmes de Map/Reduce.

Ensuite, nous voulons réaliser un algorithme de PageRank. C'est une application qui semble a priori plus ambitieuse que la première mais suscite chez nous un réel attrait.

Enfin, s'il nous reste du temps, nous aimerions nous pencher sur un tout autre type de problème, cette fois-ci plus calculatoire, et nous pensons notamment à un algorithme comme celui de la Couverture Exacte.

README

Le projet s'utilise à travers trois scripts en plus de l'application distribuée. Voici les commandes à exécuter pour :

- **./deploy.sh** : compile le projet et l'installe sur trois machines distantes
- **./exec.sh** : démarre les noeuds et le serveur de nom
- **java votre.ApplicationDistribuee** : uploadez des fichiers et lancez une ou plusieurs applications
- **./force_stop.sh** : arrête les serveurs

Pour uploader un fichier les noeuds, il faut utiliser

java -cp /work/$USER/Hidoop hdfs.HdfsClient write line <fichier>