Projet de Big Data sur le Twitter

Auteur :
DAURIAC Lucile
LESNE Nathan

Chargé de TD :
AUBER David

Table des matières

1 Infrastructure

Les données :

- 1 tweet : 5Ko
- 330 Million d'utilisateur
- 504 Million de Tweet par jour
- 1 block : 128 Mo
- Chaque donnée est répliquer en 3 Blocks
- Le cluster a 24 machines
- 1 machine : 1 To de mémoire

Combien de jours complets de collecte de données pouvons-nous stocker sur notre infrastructure de test (la salle de TP) ?
Il y a 24 machines de 1To de mémoire donc on peut garder 24To de mémoire.
Il y a 504 Million de tweet par jour et un tweet est de 5Ko donc il y a 2 520Go de donnée par jour. avec une réplication de 3, on a alors 7 56OGo de donnée par jour à sauvegarder.
Sachant qu'on a 24To de mémoire et qu'il y 7 560Go de donnée par jour alors on peut sauvegarder environ 3 jours de donnée.

Pour ce nombre de jour de collecte complet, combien de blocs de données seront disponibles sur chaque machine en moyenne ?
Il y 3 jours de donnée pour 24To de mémoire, sachant qu'un block fait 128Mo, il y a alors 187 500 blocks réparti dans les 24 machines. Il y a donc environs 7 813 blocks par machine.

Afin de planifier nos achats de matériels futurs, calculez le nombre de machines total que nous devrons avoir dans notre cluster pour stocker 5 ans de tweets ?
Il y a 7 560Go de donnée par jour donc environ 2 759To de donnée par ans. Donc pour 5 ans, il y aura 13 797To de donnée.
Sachant qu'il y a 1To de mémoire par machine alors il faudra 13 797 machines pour sauvegarder les données sur 5ans.

2 Projet

On a choisi de faire le projet en mapreduce et d'utiliser les programmes drivers pour executer les codes.

Chaque mapper créé, parse le JSON et récupère les données qu'on a besoin de traiter uniquement.
Pour les tris, on a choisis de faire un système de mapper, combiner, reducer pour traiter les informations.
Pour le tri avec le top K, on a procéder différemment. On applique deux jobs qui ont un mapper et réducer. On a du créer un fichier intermédiaire pour sauvegarder le premier job et qui est repris en entrée dans le deuxième Job. Le premier permet de compter les hashtags et le deuxième de faire le top k avec la treeMap.
Dans le reducer, les données sont sauvegardées dans Hbase, chaque ligne commence par le jour du job.

Pour la partie web, notre api utilise les filtres Hbase pour récupérer les données de chaque jour et les renvoyer.

3 Performance

On a testé nos jobs sur le jour 4 car c'était l'un des plus gros jour, le mapReduce lit environs 23 Go de données.

Partie Hashtag

	CountHastags	top 100	UserHashtag
temps min	52s	1mn 40	48s
temps max	1mn 2s	1mn 56	1mn

Partie User

	AllHastagUser	CountTweet	TweetCountry	TweetLanguage
temps min	54s	1mn 14s	49s	48s
temps max	1mn 1s	1mn 34s	1mns	49s

On a pu voir que le top k était particulièrement long soit environs deux fois plus long car il exécute deux jobs.
Nos fonctions utilisent des combiners après les mappers, ce qui permet de réduire le nombre de messages qu'il y a sur le réseau. Le nombre de messages sur le réseau dépend aussi du nombre de mapper, on a pu voir en faisant les tests qu'il y en avait 50 des mappers. D'après nos codes comme on renvoie un maximum de données, on a donc 50 fois le nombre de données à renvoyer.
Par exemple : pour la mapReduce qui cherche combien il y a de nombre d'hashtag par pays, on va se retrouver alors au maximum avec 50 fois le nombre total de pays en message qui circule sur le réseau.
Celui qui va faire circuler le plus de message sur le réseau va être l'algorithme avec les top k puisqu'il exécute deux jobs. Pour le premier job, on a en entrée environs 23Go de données mais il écrit 3Go de données dans le fichier en sortie puis il est récupéré par le deuxième job.

4 Difficultés

Pour adapter le mapper et le combiner, au début, on avait un mapper qui parsait les tweets et qui écrivait dans une classe Tweet qui implement Writable pour le passer à un combiner. Mais après plusieurs recherches, on a du réadapter le mapper car le combiner devait avoir les mêmes entrées et sorties pour être fonctionnel.

Pour le top k, on a du d'abord compter les hashtags puis faire la treeMap pour faire un top k, et on a du alors exécuter deux jobs et créer un fichier intermédiaire à faire passer entre les deux jobs.

Pour le parser, on a essayer différent parser pour trouver un qui soit fonctionnel selon nos besoins.

Pour Hbase, on a eu des problèmes de connexion, on a du modifier les entiers en string car sinon ils étaient sauvegardés en bytes et donc difficile à récupérer pour la partie web.

Pour la partie web, on a pu récupérer les données dans hbase avec le back pour les visualiser les données mais on pense que le temps de chargement des données est trop long par rapport au temps d'affichage et donc qu'il y a des problèmes pour voir le résultat. Le nombre de données à récupérer est importante.