Projet PLE
Univerité de Bordeaux 2020-2021

Alexandre Erard,
Maxime Gresse

January 25, 2021

1 Infrastructure

- Les utilisateurs de twitter produisent environs 504.10^6 tweets par jours. Sachant que 1 tweet fait environ 5Ko.

  $$504.10^6 \times 5 \times 3 = 756.10^7 Ko = 7,56 To$$

  Notre Cluster possède actuellement 18.16To de stockage. Donc on peut stocker:

  $$\frac{18,16}{7,56} \approx 2,40$$

  un peu plus de 2 jours de tweet.

- Pour 2 jour de tweet on obtient un total de $7,56 \times 2 = 15To$. Un blocs faisant 128Mo on obtient 117187 blocs disponnibles.

  $$\frac{1,5.10^7}{128} \approx 117187$$

- 5ans de tweets équivaut à $7,56 \times 365 \times 5 = 13797To$ de données. Nous disposons actuellement de 20 machines possédant en moyenne

  $$\frac{18.16}{20} = 0.93 = 930Go$$

  En gardant un stockage de 930Go par machine il nous faudrait alors 14816 machines.

  $$\frac{13797 - 18.6}{0.93} \approx 14815.5$$

2 Technologie utilisé

Pour réaliser notre batch layer nous avons choisis d'utiliser MapReduce. Nous avons fais ce choix du a divers problèmes qu'on a eu avec Spark qui ne fonctionner pas toujours même avec un programme simple. Ce choix entraîne donc de plus faible performance dû à l'écriture de fichiers. Nous stockons nos résultat dans une base HBase.

Pour notre application web nous avons choix choisie d'utiliser Vue pour le front et express/Node pour l'api backend.

3 Analyse de performance

3.1 Evolution du nombre de tweets par jour contenant un mot spécifique sur le mois

Cet algorithme est le 1er que nous avons implémentés. Il va d'abord trier les tweets pour ne récupérer que ceux "valables" et donc ne prendre en compte que ceux qui ont un champ "text" qui contient le mot entré en paramètre. On va aussi mettre de côté tous les tweets qui sont seulement des Retweets de tweet contenant le mot et non pas des citations. On ne garde donc que les tweets pour lequels les gens ont écrit le mot en question. Une fois ce filtrage effectué on renvoie un couple Text / IntWritable. Le Text contient le jour (ex: Mar 01) et le IntWritable est simplement un 1. Ce qui signifie que ce jour là on a eu un tweet avec le mot en paramètre. Ensuite notre combiner va récupérer les couples Jour / 1 et les additionner afin d'éviter de faire passer autant de messages sur le réseau que le nombre de fois que le mot a été tweeté sur le mois. Le combiner va renvoyer le couple Text (Jour) / IntWritable (Total sur le jour) au reducer. Le reducer va donc faire la somme des totaux pour chaque jour (si nécessaire). Il va ensuite écrire le résultat dans HBase avec comme clé le mot en paramètre, comme colunm family "number" et comme colonne le jour avec en value le total du nombre de tweets contenant le mot ce jour là.

- Avec les 22 premiers jours et les 23 machines du cluster on peut donc estimer le nombre maximum de messages à : 23 x 22 messages au maximum
- Avec une exécution sur tout le cluster avec le mot "minecraft" le job est exécuté en 3min04

3.2 Evolution du nombre de tweets par jour d'un hastag sur le mois

Pour cet algorithme nous avons filtrés les tweets contenant des hahstags. Ensuite pour chaque hashtag contenu on renvoie un couple Text ("Date,Hashtag") / IntWritable (1). Un combiner va récupérer ces couples et additioner les "1" afin de commencer à compter combien de fois le hashtag a été tweeté par jour pour éviter de faire passer un trop grand nombre de messages sur le réseau. En effet sans combiner on ferait passer un message sur le réseau à chaque fois qu'un hashtag a été utilisé. Ici on arrive à réduire au nombre de hashtags différents multiplié par le nombre de jours et par le nombre de mappeurs. Cela reste un nombre important de messages mais sachant que nous souhaitions avoir les données pour tous les hashtags et non pas un seul par paramètre c'est la solution pour laquelle nous avons optés. Le reducer va ensuite finir le calcul du total pour chaque jour par hashtag si nécessaire et écrire le résultat dans HBase avec en clé le Hasthtag, en colunmn family "number", en colonne le jour et en value le nombre de tweets contenant ce hashtag pour ce jour là.

- Avec les 22 premiers jours et les 23 machines du cluster on peut donc estimer le nombre maximum de messages à : 23 x 22 x nb de hastags messages au maximum
- Avec une exécution sur tout le cluster le job est exécuté en 4min15

3.3 Evolution du nombre de tweets par langue spécifique sur le mois

Afin de visualiser l'évolution du nombre de tweets par jour pour une langue on filtre d'abord seulement les tweets de la langue passée en paramètre. Notre mappeur renvoie ensuite le couple Text (Date) / IntWritable (1). Un combiner va faire la somme pour chaque date afin de réduire le nombre de messages envoyés au reducer via le réseau. Le reducer termine si nécessaire la somme pour chaque jour et écrit le résultat dans HBase. On aura en clé la langue en paramètre, en column family "number", en column la date et en value le nombre de tweets dans la langue pour ce jour là.

- Avec les 22 premiers jours et les 23 machines du cluster on peut donc estimer le nombre maximum de messages à : 23 x 22 messages au maximum
- Avec une exécution sur tout le cluster le job est exécuté en 3min20

3.4 Langues les plus utilisées sur le mois

Pour récupérer les langues les plus utilisées sur le mois on va utiliser le pattern topk. On doit d'abord faire le compte du nombre de tweets par langue sur le mois. Le mappeur va donc pour chaque tweet dont la langue est renseignée renvoyer un couple Text (Langue) / IntWritable (1). On exclu la langue "und" qui correspond à "undifined" qui serait sinon Top 5. Un combiner va ensuite faire la somme des IntWritable pour chaque langue et renvoyer le couple Text (Langue) / IntWritable (Total) au reducer. Le reducer va lui terminer la somme si besoin et ajouter dans une TreeMap le couple Langue / Total avec le total en clé et la langue en valeur afin de pouvoir profiter du tri des clés par ordre croissant de la TreeMap.
Dans le cleanup on va récupérer le k renseigné en paramètre. Nous avons rencontrés des problèmes avec le parcours de la TreeMap et notamment le remove de la firstKey si la taille de la TreeMap est supérieure à k. Action qui devrait être effectuée juste après l'ajout dans la TreeMap. Ne trouvant pas la source de se problème malgré l'exemple proposé en TD nous avons décidés de parcourir la map et quand nous arrivons sur les k dernières paires de clé / valeur et donc les k plus grandes on écrit dans HBase. Cela n'est évidemment pas aussi optimisé que la solution proposée en TD puisque qu'on doit parcourir une TreeMap de taille nb de langues au lieu d'une TreeMap de taille k. Dans hbase les clés seront les indices de k (ex: de 1 à 10 si k=10), la column family est "total" et ensuite deux column, une lang qui contient le nom de langue et une "count" qui contient le nombre de tweets dans le mois pour cette langue.

- Avec les 22 premiers jours et les 23 machines du cluster on peut donc estimer le nombre maximum de messages à : 23 x nb de langues messages au maximum
- Avec une exécution sur tout le cluster le job est exécuté en 3min13

3.5 Localisations les plus populaires pour une langue sur le mois

Ici aussi afin de récupérer les locations depuis lesquelles on tweete le plus pour une langue en paramètre on utilise le pattern topk après avoir effectué un filtrage et un compte pour chaque localisation. Le mapper va filtrer les tweets de la langue en paramètre et renvoyer un couple Text (Localisation) / IntWritable (1) si la localisation est renseignée. Le combiner va ensuite effectuer une somme des IntWritable pour chaque localisation. Il va envoyer le couple Text (Localisation) / IntWritable (Total) au reducer. Le reducer va lui finir de compter le total si nécessaire et ajouter le couple clé (total) / valeur (localisation) dans une TreeMap afin de profiter du tri des clés. Ici aussi nous avons eu le même problème que pour les topk langues les plus utilisées au niveau de la fonction map.remove(map.firstKey()) , nous avons donc utilisés la même solution pour réussir à en extraire les topk localisations pour la langue. Dans hbase on écrit la langue en clé, l'index de k en column family et deux columns par CF : name qui contient le nom de la localisation et total qui contient le nombre de tweets pour cette localisation sur le mois.

- Avec les 22 premiers jours et les 23 machines du cluster on peut donc estimer le nombre maximum de messages à : 23 x 22 messages au maximum
- Avec une exécution sur tout le cluster le job est exécuté en 3min24