

# Rapport Projet BigData

William Arnault

[https://github.com/Warnault/Projet\\_BigData.git](https://github.com/Warnault/Projet_BigData.git)

January 2021

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Parseur de donnée</b>	<b>2</b>
<b>3</b>	<b>Traitement de l'information</b>	<b>2</b>
3.1	Partie User . . . . .	4
3.1.1	Résultat . . . . .	4
3.2	Partie Hashtag . . . . .	5
3.2.1	Présentation . . . . .	5
3.2.2	Les Résultats . . . . .	5
<b>4</b>	<b>Mes Difficultés</b>	<b>6</b>

## 1 Introduction

Pour la réalisation de ce projet j'ai décidé d'utiliser hadoop pour faire des MapReduces.

Les testes et exécutions de programme on était sur le petit échantillon données(10 000 tweets).

## 2 Parseur de donnée

**Pour parser les données présentes dans le fichier j'ai choisi d'utiliser les librairies :**

- *com.fasterxml.jackson.databind.JsonNode*,
- *com.fasterxml.jackson.databind.Object*

me permettant ainsi de parser facilement chaque ligne de nljson dans le mapper:

```
JsonNode x = new ObjectMapper().readTree(<data>.toString())
```

Avec ce nouvel objet **x** je veux m'en servir comme un json pour extraire les informations souhaitées, par exemple pour récupérer la langue d'un tweet :

```
String lang = x.get("lang").asText();
```

## 3 Traitement de l'information

Pour la réalisation de ce projet, je suis parti sur le schéma de Mapper/Combiner/Reducer me permettant ainsi via le Combiner de minimiser grandement le nombre de données envoyé au Reducer tout en maximisant la rapidité de calcul.

Puisque toutes les données sont parsées et triées par les mappers cela nous permet donc de ne garder que le strict nécessaire pour l'algorithme, on envoie ces données au combiner afin que celui-ci concatène les informations (regroupe les doublons en incrémentant le compteur : la valeur de la key). Une fois la concaténation de toutes les données, chaque combiner envoie au reducer les informations. Le reducer recevra dans le pire des cas  $X * N$  fois chaque messages ( $X$ =étant le nombre de combiner et  $N$ =nombre de données différentes).

On peut observer sur l'image 1 ci-dessous des différentes quantités de données aux entrées et sorties de chacun des éléments.

Au commencement, il y avait *1000* messages(couple clef/valeur).

A la sortie des Mappers il nous en reste 110, qui seront envoyés aux combineurs.

A la sortie des Combiners il ne nous en reste plus de 20.

Et le Reducer concatène tous ces messages et nous retourne les 20 messages réunis et sommés.

```
Map input records=10000
Map output records=110
Map output bytes=1557
Map output materialized bytes=30
Input split bytes=125
Combine input records=110
Combine output records=20
Reduce input groups=20
Reduce shuffle bytes=304
Reduce input records=20
Reduce output records=20
```

Figure 1: Analyse des entrées/sorties des Mappers/Combiners/Reducers pour la recherche des pays

### 3.1 Partie User

Pour cette partie on utilise la partie "user" du tweet.

#### 3.1.1 Résultat

**Country** Il a fallu 15s à l'algorithme pour s'exécuter au complet et sortir le résultat ci-dessous : la figure 2.

```
Argentina 10
Australia 6
Brasil 528
Brazil 10
Canada 6
Chile 3
Colombia 1
India 1
Indonesia 10
Malaysia 10
Mexico 1
México 21
Peru 1
Republic of the Philippines 10
Singapore 1
Thailand 3
United States 496
Vietnam 1
日本 10
```

Figure 2: résultat d'une recherche de pays dans un fichier

**Language** Il a fallu 16s à l'algorithme pour s'exécuter au complet et sortir le résultat ci-dessous : la figure 3.

```
ar 14878
bn 3
ca 45
cs 1
cy 10
da 15
de 91
el 3
en 3334653
es 460320
et 78
eu 28
fa 66
fi 21
fr 1711
hi 3321
ht 171
hu 10
in 28920
is 1
it 136
iw 1
ja 827541
ko 52003
lt 3
lv 1
my 3
ne 1
nl 21
no 3
or 1
pl 10
ps 3
pt 709836
ru 78
sl 1
sv 6
ta 45
th 83021
tl 3655
tr 630
und 147696
ur 45
vi 45
zh 465
```

Figure 3: résultat d'une recherche de langue dans un fichier

## 3.2 Partie Hashtag

### 3.2.1 Présentation

La partie Hashtag dans son fonctionnement est identique à la partie User.

**Pour faire le TopK** sur les meilleurs hashtags j'ai du faire deux jobs à la suite. Un premier qui récupère la fréquence des hashtags et un deuxième qui conserve seulement les meilleurs. Pour réaliser cela, j'ai fait une sauvegarde du premier job dans un fichier temporaire, afin de m'en servir pour le deuxième.

### 3.2.2 Les Résultats

Les résultats présentés ci-dessous ont été réalisés sur le hashtag "BestMusicVideo".

Pour récupérer la fréquence de cet hashtag il a fallu 18s à l'algorithme pour s'exécuter et récupérer l'information ci-dessous : la figure 4.

```
BestMusicVideo 9
```

Figure 4: résultat de la recherche du nombre d'utilisation du hashtag BestMusicVideo

Pour récupérer la liste utilisateur, il a fallu 18s à l'algorithme pour s'exécuter et récupérer l'information ci-dessous : la figure 5.

```
Ffahsa113
InnerCypher
Nat11a45032443
beashaky
brlchy
lenileo2
milky_kookies
pmkjsj
vkookmaschido
```

Figure 5: résultat des utilisateurs du hashtag BestMusicVideo

Via ces deux résultats, on peut constater qu'ils se confirment entre eux puisque le 1er résultat annonce 9 utilisateurs et dans le second on obtient bien une liste de 9 utilisateurs.

**Le topk5** a mis 42s à s'exécuter et à récupérer l'information ci-dessous : la figure 6.

```
RedeBBB 13
BTS 14
iHeartAwards 15
bbb20 27
BBB20 41
```

Figure 6: résultat d'une recherche de topk5 dans un fichier

## 4 Mes Difficultés

Pour moi, le plus compliquer était de me servir du cluster. Ayant eu des difficultés au démarrage de l'année avec l'impossibilité de me connecter au cremi, j'ai du aussi rattraper le retard accumulé, d'où la seule présence d'hadoop sur ce projet.