

Université Abdelmalek Essaâdi
ENSA Tétouan



1ère année Cycle d'Ingénieur – BDIA
Module : Big Data

Rapport du projet Analyse de données Twitter en temps réel

Préparé par :

BOULAALAM YASSINE
BOUSKINE OTHMANE
BOUHAFI TAHA

Encadré par:

Prof.Faouzi Marzouki

SOMMAIRE

I.	Introduction.....	2
II.	Architecture des outils utilisés.....	3
III.	Implémentation sur Ubuntu en machine virtuelle.....	5
IV.	Collecte des flux de tweets avec Apache Kafka.....	7
V.	Stockage des tweets dans HDFS.....	11
VI.	Analyse des tendances et détection des sentiments avec Apache Spark et TexBlob.....	12
VII.	Conclusion.....	14

Introduction

■ Contexte et Problématique

À l'ère numérique, les réseaux sociaux génèrent d'énormes quantités de données chaque jour.

Ces données, souvent qualifiées de "big data", offrent des opportunités uniques pour analyser les comportements des utilisateurs, identifier les tendances émergentes et comprendre les sentiments exprimés dans les tweets. Cependant, traiter et analyser ces vastes volumes de données en temps réel représente un défi majeur en termes de stockage, de traitement et d'analyse.

Pour relever ces défis, il est crucial de mettre en place une architecture capable de gérer des flux de données en temps réel, de les stocker efficacement et de réaliser des analyses complexes.

Cette étude se concentre sur l'utilisation de Kafka pour le chargement des données en temps réel, de HDFS pour le stockage distribué, et d'Apache Spark pour l'analyse, en utilisant Spark-SQL et TextBlob pour extraire les hashtags tendances et analyser les sentiments des tweets.

■ Objectifs de l'Étude

L'objectif principal de cette étude est de concevoir et de mettre en œuvre une architecture de traitement des données à grande échelle permettant d'analyser en temps réel les tweets pour identifier les hashtags tendances et déterminer les sentiments exprimés. Plus précisément, cette étude vise à :

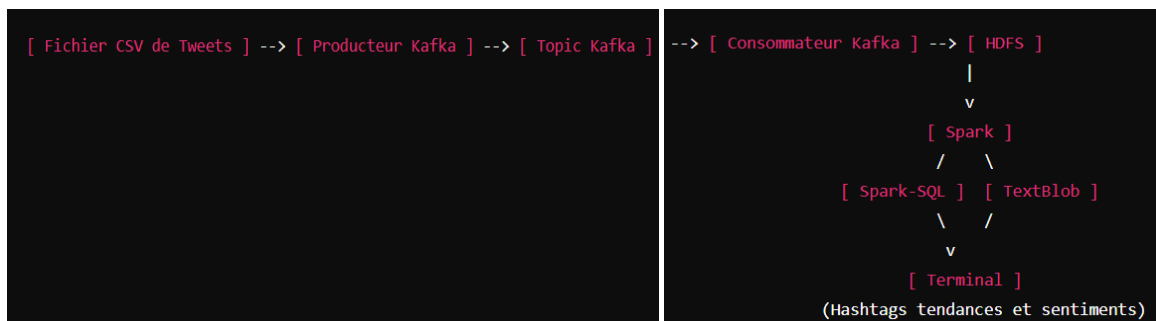
- **Chargement des Données avec Kafka**
- **Stockage des Données avec HDFS**
- **Analyse des Données avec Spark**
- **Affichage des Résultats**

Ce projet illustre comment une combinaison de technologies modernes telles que Kafka, HDFS et Spark peut être utilisée pour traiter et analyser les big data en temps réel. Les insights obtenus, tels que les tendances et les sentiments des utilisateurs, sont essentiels pour des applications dans divers domaines, notamment le marketing, la gestion de la réputation et la prise de décision stratégique.

Architecture des outils utilisés

Pour répondre aux exigences de traitement et d'analyse des big data en temps réel, l'architecture mise en place utilise une combinaison de technologies modernes. Voici une description détaillée de cette architecture :

- ❖ Ingestion des Données avec Apache Kafka
- ❖ Stockage Distribué avec HDFS
- ❖ Traitement et Analyse avec Apache Spark
- ❖ Affichage des Résultats



- Fichier CSV de Tweets :
Point de départ des données sous forme de fichier CSV.
- Producteur Kafka :
Lit les tweets du fichier CSV et les envoie vers le topic Kafka pour ingestion en temps réel.
- Topic Kafka :
Espace de stockage temporaire pour les tweets, permettant une ingestion en continu et scalable.
- Consommateur Kafka :
Lit les tweets du topic Kafka et les stocke dans HDFS pour un stockage durable et distribué.
- HDFS :
Stockage distribué et scalable des tweets, permettant un accès rapide et fiable pour le traitement ultérieur.

➤ **Spark :**

Plateforme de traitement de données distribuée pour analyser les tweets stockés dans HDFS.

➤ **Spark-SQL :**

Composant de Spark utilisé pour exécuter des requêtes SQL et extraire des informations structurées, comme les hashtags tendances.

➤ **TextBlob :**

Bibliothèque de NLP intégrée dans Spark pour l'analyse des sentiments des tweets.

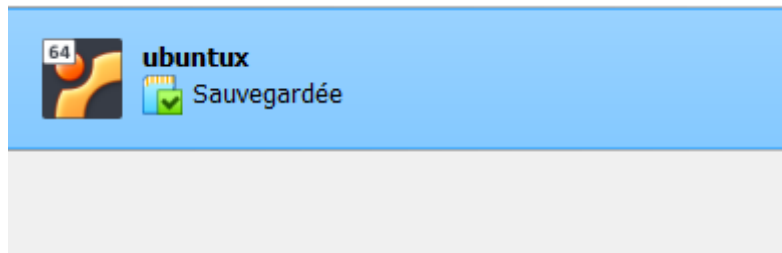
➤ **Terminal :**

Interface de sortie où les résultats de l'analyse sont affichés, permettant une visualisation rapide des tendances et des sentiments.

- ✓ Cette architecture intégrée permet de gérer efficacement le flux de données en temps réel, de stocker et de traiter de grands volumes de données, et d'extraire des insights précieux à partir des tweets.

Implémentation sur Ubuntu en machine virtuelle

- Le système d'exploitation sous lequel on a travaillé est : Ubuntu



- Le téléchargement de :

- Hadoop version :3.3.6
- Kafka version : 2.12-3.3.2
- Apache Spark version : 3.5.1
- Java

- Les commandes de configuration :

- **Pour Installation de Java :**

```
sudo apt-get update
```

```
sudo apt-get install default-jdk -y
```

On installe Java car Apache Kafka et Apache Spark nécessitent Java pour fonctionner, étant donné qu'ils sont construits sur la JVM (Java Virtual Machine).

- **Pour installation de Kafka :**

```
wget https://archive.apache.org/dist/kafka/3.3.2/kafka_2.12-3.3.2.tgz
```

```
tar -xzf kafka_2.12-3.3.2.tgz
```

On télécharge et extrait Kafka pour configurer un système de messagerie distribué, essentiel pour le traitement de données en temps réel.

- **Pour installation de Hadoop :**

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz  
tar -xzf hadoop-3.3.6.tar.gz
```

On télécharge et extrait Hadoop pour mettre en place un Framework de traitement distribué et de stockage de données

- **Pour installation de Spark :**

```
wget https://www.apache.org/dyn/closer.lua/spark/spark-3.5.1/spark-3.5.1-bin-hadoop3.tgz  
tar -xzf spark-3.5.1-bin-hadoop3.tgz
```

On télécharge et extrait Spark pour disposer d'un moteur de traitement de données rapide et polyvalent, essentiel pour l'analyse et le traitement

➤ **La configuration :**

- Configure core-site.xml, hdfs-site.xml, yarn-site.xml, and mapred-site.xml .
- Configure spark-env.sh
- Bashrc doit contenir Path de Spark et Hadoop et Java

➤ **Création d'un environnement et installer les bibliothèques nécessaires :**

- ✓ Pip Install pyspark
- ✓ pip install textblob
- ✓ pip install confluent_kafka
- ✓ Pip install pandas
- ✓ Pip install kafka-python

Collecte des flux de tweets en temps réel

avec Apache Kafka

1) Lancer le serveur pour Kafka :

- D'abord on commence par lancer le serveur zookeeper :

```
sh bin/zookeeper-server-start.sh config/zookeeper.properties
```

Lancement du serveur Zookeeper en utilisant le script de démarrage et le fichier de configuration zookeeper.properties.

- Ensuite Lancer le serveur Kafka :

```
sh bin/kafka-server-start.sh config/server.properties
```

2) Configuration de Kafka:

- Creation du Topic:

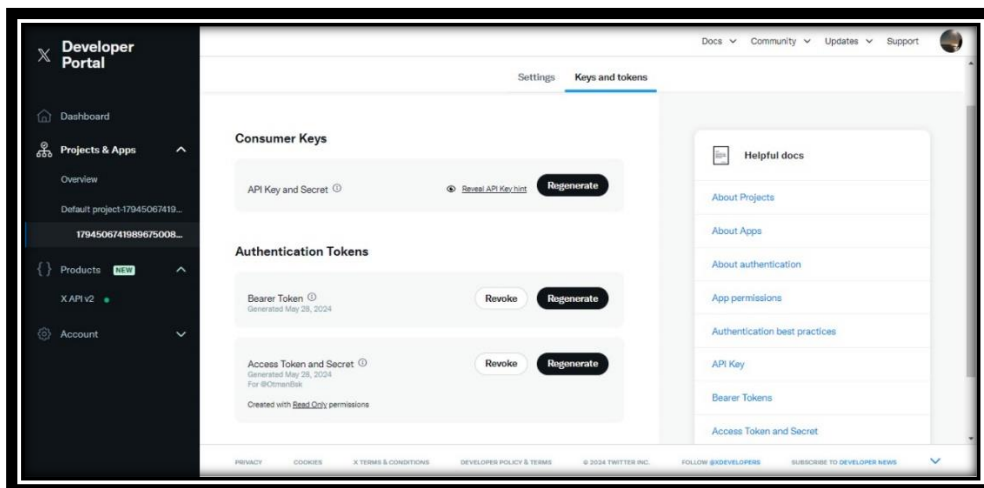
```
bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1  
--partitions 1 --topic twitter_topic
```

Création d'un nouveau topic nommé testTopic avec un facteur de réplication de 1 et une seule partition en utilisant le script kafka-topics.sh.

- Lister les topics qui existent :

```
bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```


3) Test de collection utilisant 'API' :



```
import tweepy
from kafka import KafkaProducer
import json

# Twitter API
api_key = '#####'
api_secret_key = '#####'
access_token = '#####'
access_token_secret = '#####'

auth = tweepy.OAuthHandler(api_key, api_secret_key)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)
producer = KafkaProducer(bootstrap_servers='localhost:9092',
                        value_serializer=lambda x: json.dumps(x).encode('utf-8'))

class MyStreamListener(tweepy.StreamListener):
    def on_status(self, status):
        tweet = {
            'text': status.text,
            'user': status.user.screen_name,
            'created_at': str(status.created_at)
        }
        producer.send('twitter_topic', tweet)
        print(f'Tweet sent: {tweet}')

stream_listener = MyStreamListener()
stream = tweepy.Stream(auth=api.auth, listener=stream_listener)
stream.filter()
```

Ce script Python utilise l'API Twitter via la bibliothèque Tweepy pour capturer des tweets en temps réel et les envoyer à un sujet Kafka.

Tout d'abord, il configure l'authentification pour accéder à l'API Twitter en utilisant les clés et les jetons d'accès nécessaires.

Ensuite, un producteur Kafka est configuré pour envoyer les données vers un serveur Kafka local. Une classe personnalisée, `MyStreamListener`, est définie pour écouter les flux de tweets provenant de l'API Twitter. À chaque nouveau tweet capturé, les informations pertinentes telles que le texte du tweet, le nom d'utilisateur et l'horodatage sont extraites et formatées dans un dictionnaire.

Ces données sont ensuite envoyées au sujet Kafka spécifié.

La méthode `stream.filter()` est utilisée sans paramètres pour capturer et envoyer tous les tweets disponibles en temps réel, garantissant ainsi que chaque tweet est transmis au système de traitement ou d'analyse en continu.

Ce script offre un moyen efficace de collecter et de traiter en continu les données Twitter via Kafka pour une analyse ultérieure.

⚠ **Initialement, nous avons l'intention d'utiliser l'API de Twitter pour collecter des tweets en temps réel. Cependant, l'accès à l'API de Twitter n'est pas gratuit et nécessite des autorisations.**

Pour contourner cette contrainte, nous avons opté pour une **approche alternative en téléchargeant un fichier .csv préexistant contenant des tweets**, nous permettant ainsi de travailler avec des données réelles malgré les limitations de l'API Twitter. La dataset utilisée : 'viral_tweets.csv'.

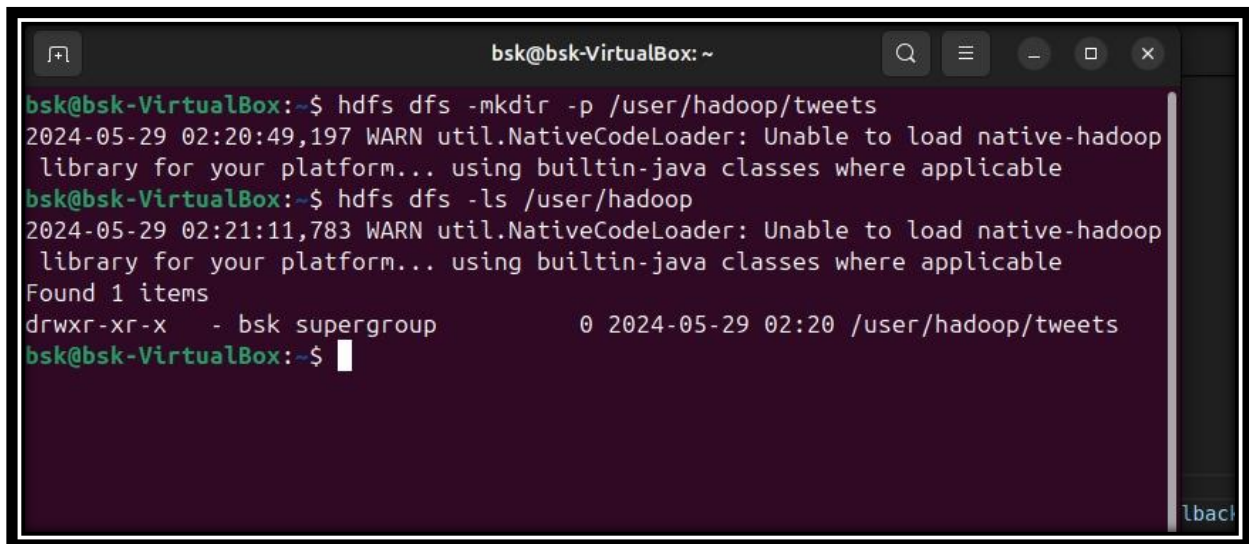
Donc Après on exécute le fichier Consumer.py premierement et après, exécuter TwitterProducer.py :

```
TwitterProducer.py Python: TwitterProducer X Python: Consumer X + [ ] ...  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
Message delivered to twitter topic  
  
rect communication with the Space Station #Dragon'  
Received message: {"id": "205502360112476161", "time stamp": "2012-05-24 03:35:55", "tweet text": "b'The President just called to say congrats. Caller ID was blocked, so at first I thought it was a telemarketer :)" }"  
Received message: {"id": "205314559542693888", "time stamp": "2012-05-23 15:09:40", "tweet text": "b'Dragon fly by of Space Station planned for 12:47 am California time. All systems green. #dragonlaunch'"}  
Received message: {"id": "204911319260987392", "time stamp": "2012-05-22 12:27:20", "tweet text": "b'Lanuch video of Falcon 9 from Cape Canaveral tracking cameras http://t.co/QnBne3pa #dragonlaunch'"}  
Received message: {"id": "204883910654570496", "time stamp": "2012-05-22 10:38:25", "tweet text": "b'Navigation bay pointing to deep space and star map being generated by star tracker one. Yes! #DragonLaunch'"}  
Received message: {"id": "204882126884188160", "time stamp": "2012-05-22 10:31:20", "tweet text": "b'Dragon spaceship opens the navigation pod bay door without hesitation. So much nicer than HAL9000 :) #DragonLaunch'"}  
Received message: {"id": "204858156801732608", "time stamp": "2012-05-22 08:56:05", "tweet text": "b'Allo big thanks to the Air Force, FAA and all of our partners for their support of this mission.'"}  
Received message: {"id": "204857362782224384", "time stamp": "2012-05-22 08:52:56", "tweet text": "b'Huge appreciation for @NASA, without whom we could not even have started, let alone reached this far.'"}  
Received message: {"id": "204845069772128256", "time stamp": "2012-05-22 08:04:05", "tweet text": "b'Falcon flew perfectly!! Dragon in orbit, comm locked and solar arrays active!! Feels like a giant weight just came off my back :)'" }
```

Les scripts démontrent l'utilisation de Kafka pour produire et consommer des messages représentant des tweets. D'abord, il faut démarrer les services Zookeeper et Kafka pour gérer les échanges. Ensuite, un topic nommé ``twitter_topic`` est initialisé dans Kafka. Le script du producteur lit des tweets depuis un fichier CSV et envoie ces messages au topic Kafka, en utilisant des fonctions pour configurer le producteur, lire le fichier CSV et convertir chaque tweet en JSON avant de l'envoyer. Simultanément, le script du consommateur s'abonne au même topic pour lire les messages entrants. Il est configuré pour récupérer les messages depuis le début du topic et traite les erreurs éventuelles en cours de route. Chaque message reçu est décodé et affiché. Cette solution permet de simuler la production et la consommation de tweets en temps réel, ce qui est utile pour le traitement et l'analyse de gros volumes de données.

Stockage des tweets dans HDFS

- Créer un fichier dans hdfs ou on va enregistrer le tweet envoyé à Kafka-Consumer

A terminal window titled 'bsk@bsk-VirtualBox: ~' with standard window controls. It shows two commands being executed: 'hdfs dfs -mkdir -p /user/hadoop/tweets' and 'hdfs dfs -ls /user/hadoop'. The first command is successful. The second command shows a directory listing for '/user/hadoop' containing one item: 'tweets' with permissions 'drwxr-xr-x', owner 'bsk', group 'supergroup', size '0', and timestamp '2024-05-29 02:20'. There are warning messages about native-hadoop library loading for both commands.

```
bsk@bsk-VirtualBox:~$ hdfs dfs -mkdir -p /user/hadoop/tweets
2024-05-29 02:20:49,197 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
bsk@bsk-VirtualBox:~$ hdfs dfs -ls /user/hadoop
2024-05-29 02:21:11,783 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x  - bsk supergroup          0 2024-05-29 02:20 /user/hadoop/tweets
bsk@bsk-VirtualBox:~$
```

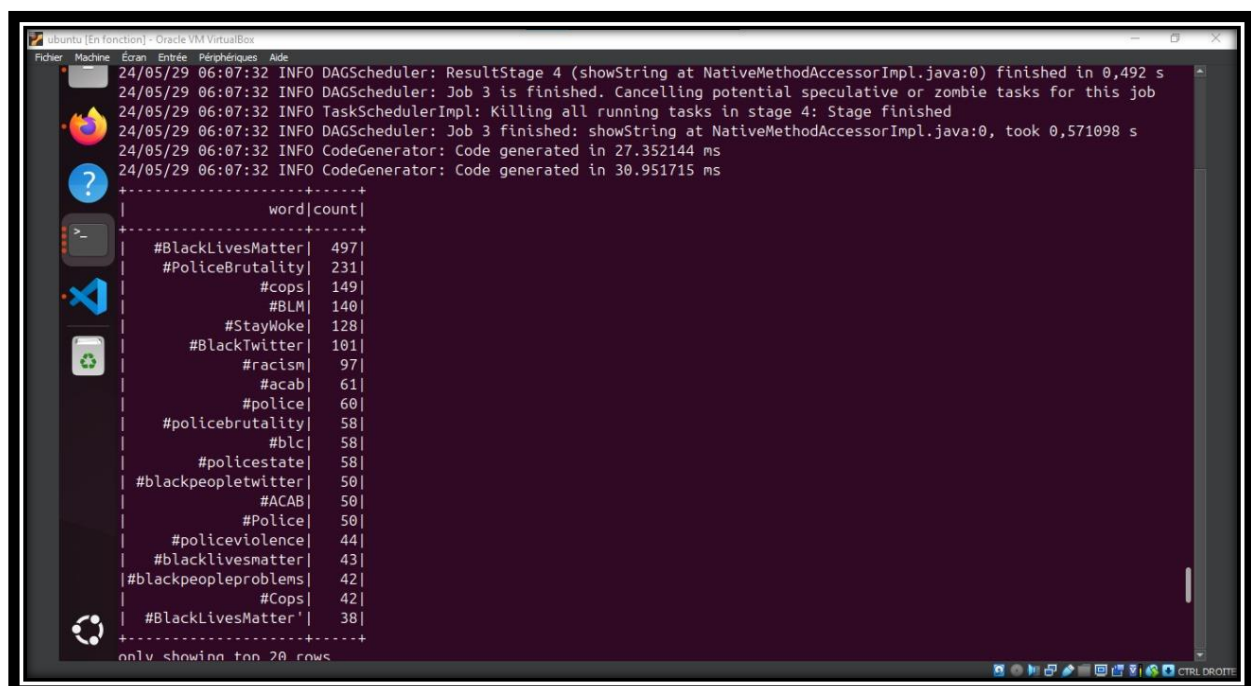
- Après que les données ont envoyés vers Twitter Consumer, on Convertit les RDD en DataFrame, et les enregistrent le DataFrame au format JSON dans HDFS.
- Le script TwitterConsumer.py utilise Apache Spark pour consommer des messages de tweets depuis un topic Kafka et les sauvegarder dans HDFS (Hadoop Distributed File System). Après avoir importé les modules nécessaires, le script configure une session Spark et un contexte de streaming avec un intervalle de 5 secondes.
- Un consommateur Kafka est créé pour lire les tweets du topic `twitter_topic`, et les messages reçus sont décodés, transformés en RDD, puis convertis en DataFrame Spark. Les tweets sont ensuite sauvegardés dans HDFS au format JSON.
- Pour exécuter ce script, il faut démarrer Zookeeper et Kafka, créer le topic Kafka, préparer le répertoire HDFS avec `hdfs dfs -mkdir -p /user/hadoop/tweets`, et lancer le script avec la commande `spark-submit`.

Analyse des tendances et détection des sentiments avec Apache Spark et TextBlob

On a créé le script `twitter_analyse.py` qui utilise Apache Spark pour analyser les tweets stockés dans HDFS. Il extrait les hashtags les plus courants en utilisant Spark-SQL et calcule la polarité moyenne des sentiments des tweets en utilisant TextBlob. Les étapes comprennent le chargement des données des tweets depuis HDFS, l'extraction des hashtags, le calcul de la polarité des sentiments et l'affichage des résultats. Avant de l'exécuter avec Spark, assurez-vous d'avoir démarré Zookeeper et Kafka, créé le topic Kafka "twitter_topic", et préparé le répertoire de destination dans HDFS.

Une fois ces étapes effectuées, utilisez la commande `spark-submit` pour exécuter le script.

■ Les résultats obtenus et leur interprétation :



```
24/05/29 06:07:32 INFO DAGScheduler: ResultStage 4 (showString at NativeMethodAccessorImpl.java:0) finished in 0,492 s
24/05/29 06:07:32 INFO DAGScheduler: Job 3 is finished. Cancelling potential speculative or zombie tasks for this job
24/05/29 06:07:32 INFO TaskSchedulerImpl: Killing all running tasks in stage 4: Stage finished
24/05/29 06:07:32 INFO DAGScheduler: Job 3 finished: showString at NativeMethodAccessorImpl.java:0, took 0,571098 s
24/05/29 06:07:32 INFO CodeGenerator: Code generated in 27.352144 ms
24/05/29 06:07:32 INFO CodeGenerator: Code generated in 30.951715 ms

+-----+
| word|count|
+-----+
| #BlackLivesMatter| 497|
| #PoliceBrutality| 231|
| #cops| 149|
| #BLM| 140|
| #StayWoke| 128|
| #BlackTwitter| 101|
| #racism| 97|
| #acab| 61|
| #police| 60|
| #policebrutality| 58|
| #blc| 58|
| #policestate| 58|
| #blackpeopletwitter| 50|
| #ACAB| 50|
| #Police| 50|
| #policeviolence| 44|
| #blacklivesmatter| 43|
| #blackpeopleproblems| 42|
| #Cops| 42|
| #BlackLivesMatter'| 38|
+-----+
only showing top 20 rows
```



```
24/05/29 06:07:36 INFO DAGScheduler: Job 5 finished: collect
:42, took 0,194485 s
Average sentiment polarity: 0.03
24/05/29 06:07:36 INFO SparkContext: SparkContext is stoppi
```

Les résultats fournis comprennent la polarité moyenne des sentiments des tweets et les hashtags les plus tendance dans les tweets analysés. Voici une interprétation de ces résultats :

a) Polarité moyenne des sentiments : 0.03

- La polarité moyenne des sentiments est légèrement positive (0.03). Cela suggère que dans l'ensemble, les tweets analysés ont tendance à exprimer des sentiments positifs. Cependant, la valeur est proche de zéro, ce qui indique une certaine neutralité dans les sentiments exprimés.

b) Hashtags tendance :

Les hashtags les plus fréquents dans les tweets sont listés avec le nombre de fois qu'ils apparaissent.

Les hashtags comme #BlackLivesMatter, #PoliceBrutality, #BLM, #StayWoke sont parmi les plus fréquents, indiquant que les discussions portent sur des sujets liés aux mouvements sociaux, à la brutalité policière et à la sensibilisation.

D'autres hashtags comme #racism, #acab, #policestate, et #policeviolence sont également présents, mettant en lumière les problèmes sociaux et les préoccupations liées à la justice et à l'égalité.

La présence de hashtags tels que #blackpeopletwitter et #blackpeopleproblems suggère une discussion spécifique sur les expériences et les défis rencontrés par la communauté noire.

Ces résultats indiquent une forte présence d'activisme et de sensibilisation sociale dans les discussions Twitter analysées, ainsi qu'une tendance générale vers des sentiments neutres à légèrement positifs.

Conclusion

Cette étude a démontré comment une architecture intégrée utilisant Apache Kafka, HDFS et Apache Spark peut efficacement résoudre le problème de traitement et d'analyse des big data en temps réel, en particulier pour les données issues des réseaux sociaux comme les tweets.

1) Ingestion en Temps Réel :

L'utilisation d'Apache Kafka a permis une ingestion fluide et scalable des tweets à partir d'un fichier CSV. Cette solution garantit une gestion efficace des flux de données continus, permettant une mise à jour en temps réel des informations collectées.

2) Stockage Distribué et Fiable :

En stockant les données dans HDFS, nous avons assuré une gestion robuste et distribuée des grands volumes de tweets. HDFS offre non seulement une haute disponibilité et une redondance des données, mais aussi un accès rapide et fiable pour les étapes de traitement ultérieures.

3) Analyse Avancée des Données :

Apache Spark a été utilisé pour traiter et analyser les tweets stockés. Grâce à Spark-SQL, nous avons pu extraire les hashtags tendances efficacement, et avec TextBlob, nous avons réalisé une analyse des sentiments pour évaluer les émotions exprimées dans les tweets.

Cette combinaison d'outils a permis une analyse approfondie et rapide des données.

4) Affichage et Interprétation des Résultats :

Les résultats des analyses, incluant les hashtags tendances et les sentiments des tweets, ont été affichés directement dans le terminal. Cette méthode a offert une visualisation immédiate des résultats, facilitant une interprétation rapide sans nécessiter de solutions de visualisation complexes.

Cette étude a apporté une solution pratique et efficace pour traiter et analyser en temps réel les données massives provenant des réseaux sociaux. En intégrant Kafka pour l'ingestion, HDFS pour le stockage et Spark pour l'analyse, nous avons montré qu'il est possible de gérer efficacement de

grands volumes de données tout en obtenant des insights précieux sur les tendances et les sentiments des utilisateurs.

Les insights obtenus grâce à cette architecture peuvent être utilisés pour diverses applications, comme la surveillance des tendances sur les réseaux sociaux, l'analyse de la satisfaction des clients, et la prise de décisions marketing stratégiques. Cette approche peut également être adaptée à d'autres types de données et d'autres domaines d'application, démontrant ainsi la flexibilité et la puissance de l'architecture mise en place.

En conclusion, notre travail a fourni une solution robuste pour surmonter les défis du traitement et de l'analyse des big data en temps réel, offrant ainsi une base solide pour des applications futures et des améliorations potentielles dans le domaine de l'analyse des données massives.