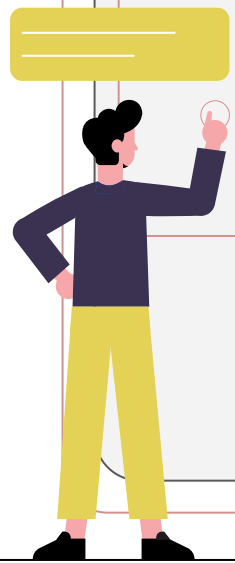
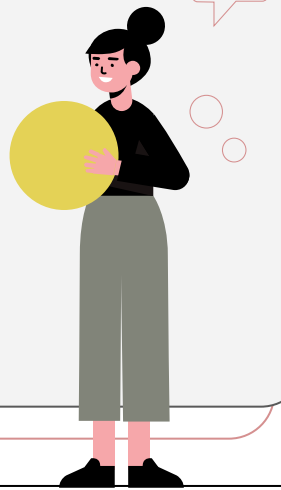
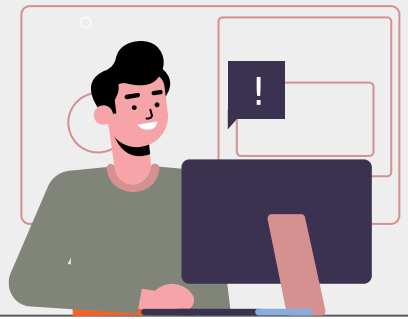
Construction d'une base de données à destination d'une banque

Caillau Yohan - Guillot Cécile





Sommaire



01

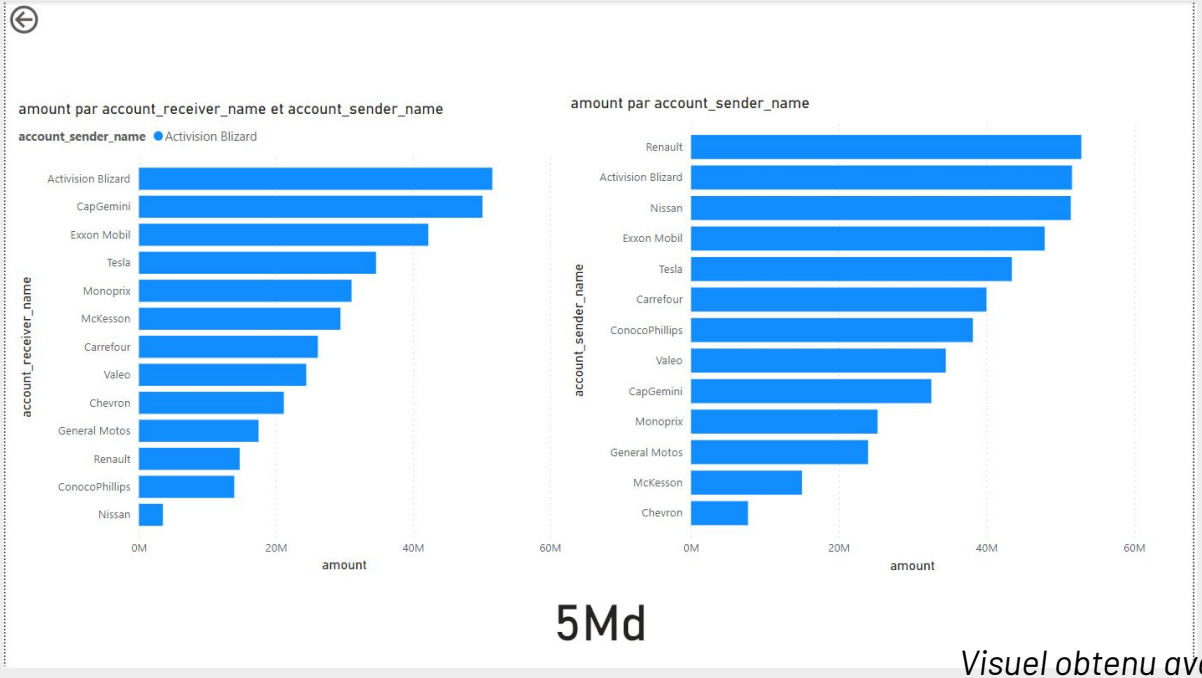
Conceptualisation du projet

Problématique rencontrée, conceptualisation des différents éléments de la solution envisagé

Problématique

- Croissance importante du groupe suite à de nombreux rachats
- Aucune donnée centralisée dans un système d'information commun
- Besoin d'avoir une architecture Big Data pour traiter un volume important de données
  - 20 millions de clients sur 10.000 agences
  - Entre 1 et 20 millions de transaction par heure

Exemple de rendu souhaité

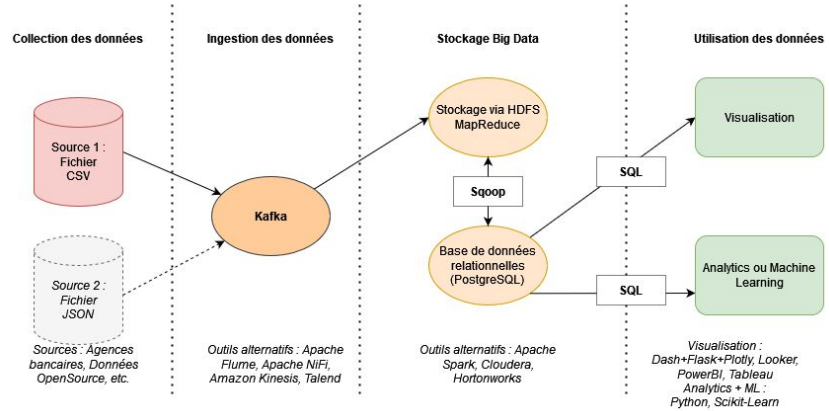


Visuel obtenu avec Power BI

Pipeline de données

Data Pipeline du projet 1

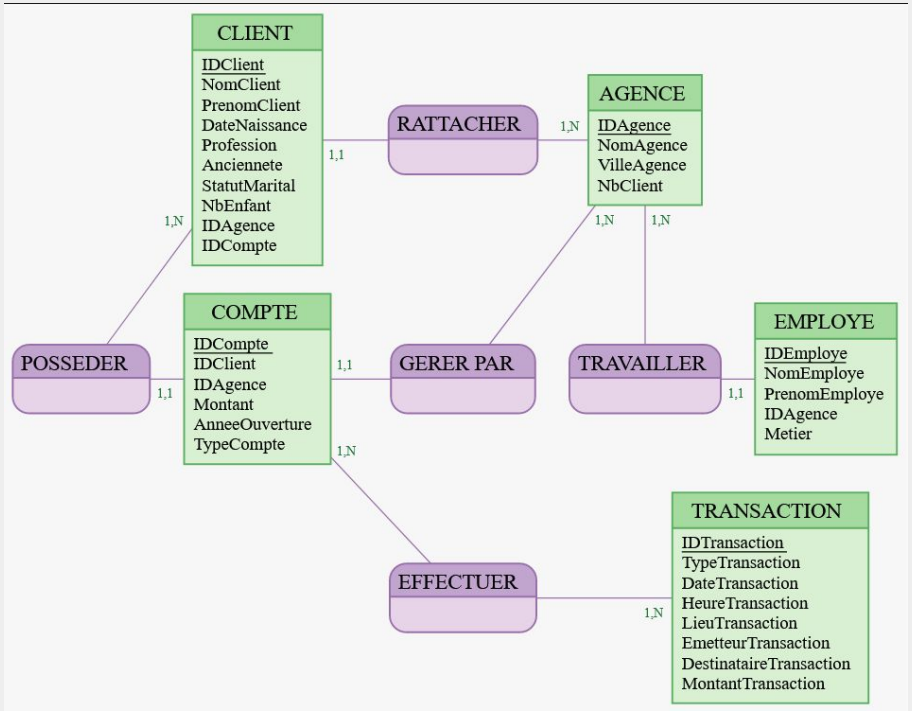
Caillau Yohan - Guillot Cécile



Autres :
+ Maintenance et orchestration via Apache Airflow ou Oozie
+ Gestion des accès via Apache Ranger ou Apache Sentry
+ Transformation des fichiers sources en .parquet

Base de données relationnelles

Script SQL



Réalisé via Mocodo

Autres besoins ?

- Ajouter une étape de transformation du format des fichiers
  - Actuellement csv mais possibilité de format différent par la suite. Uniformisé en .parquet
- Collecter des données provenant d'autres sources pour des projets de ML (par exemple)
  - API, Bases de données Open Source
- Sécuriser le système via Apache Ranger ou Apache Sentry

02

Hadoop

Architecture BigData, MapReduce,
Performances & Orchestration

Présentation HDFS & MapReduce

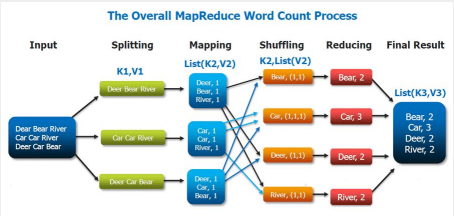
HDFS:

Système de fichiers distribués fiable stockant des fichiers volumineux

Fournit un accès aux données à travers un cluster Hadoop

MapReduce:

Système de calcul permettant de traiter un grand volume de données



Effectue le traitement Big Data

=> Tolérant aux pannes, adaptés à un traitement d'un grand volume de données

Démonstration

```python
from mrjob.job import MRJob
from mrjob.step import MRStep

class Sum_transactions_receiver(MRJob):
    def steps(self):
        """Création d'une première méthode contenant les étapes à réaliser.
           On commence par une étape de mapping puis une étape de reduce"""
        return [
            MRStep(mapper = self.mapper_transactions_receiver,
                   reducer = self.reducer_transactions_receiver)
        ]

    def mapper_transactions_receiver(self, _, line):
        """Cette méthode contient le code pour réaliser l'étape de mapping.
           La sortie obtenue lors de cette étape est une liste
           contenant des clés et des valeurs associés. """
        agency_id,account_sender_name,country_sender,\
        account_receiver_name,country_receiver,amount,\
        payment_type,datetime_timestamp = line.split(',')
        if amount != 'amount':
            yield (account_receiver_name, int(amount))

    def reducer_transactions_receiver(self, key, values):
        """Cette méthode contient le code pour réaliser l'étape de reducing.
        La sortie obtenue est le résultat
        d'une agrégation sous la forme clé-valeur."""
        yield key, sum(values)

if __name__ == "__main__":
    Sum_transactions_receiver.run()
```

Exemple de code de MapReduce

```
"Activision Blizard"    464288223
"CapGemini"     415480161
"Carrefour"     442907850
"Chevron"       374504416
"ConocoPhillips"        429380137
"Exxon Mobil"   433516012
"General Motos" 333153987
"McKesson"      370449094
"Monoprix"      351214584
"Nissan"        351627025
"Renault"       389075272
"Tesla" 311865247
"Valeo" 371589138
```

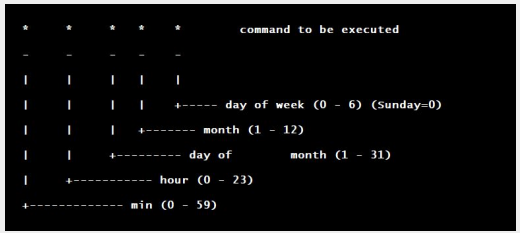
Exemple de sortie obtenue

Performance (MR sur HDFS)

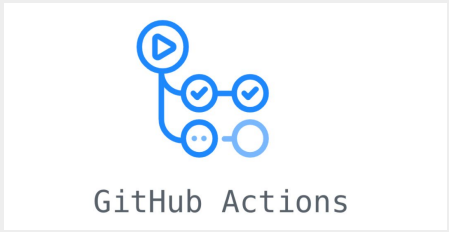
Nombre de lignes	Temps d'exécution
1000	22 secondes
10 000	24 secondes
100 000	31 secondes

Orchestration du workflow

- Création de crontab pour lancer les scripts Python
- Utilisation de GitHub Actions
- Utilisation d'outils d'orchestration de workflow : Apache Airflow, Luigi, Prefect, Apache Oozie



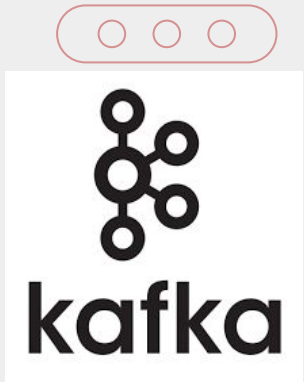
Exemple de crontab



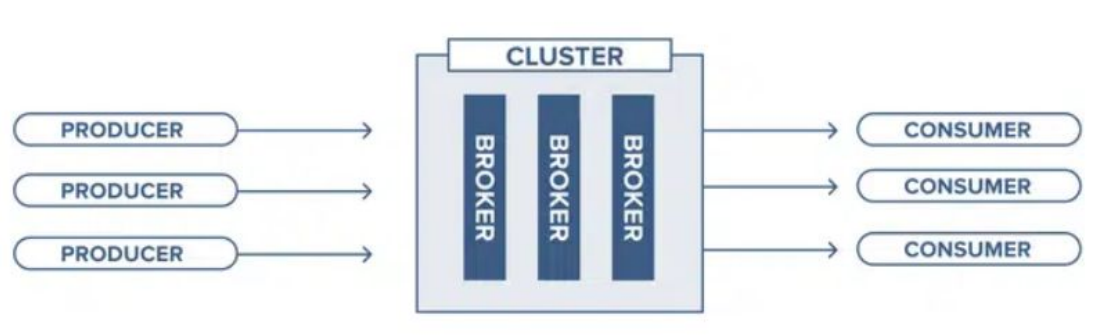
Ingestion via Kafka

03

Utilisation d'une technologie d'ingestion streaming



Considération technique



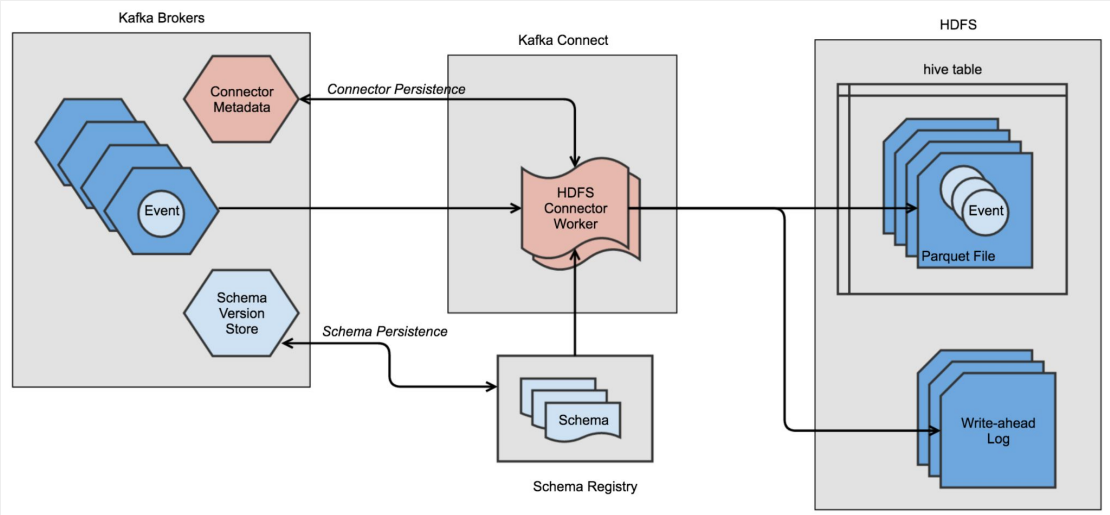
- Quantité de données par heure : 1,1 To (20 millions de transactions)
- Rétention : 1h dans Kafka
- Facteur de réplication : 3

- Taille du cluster : **3,3 To** avec réplication incluse
- Nombre de brokers : **5**

https://eventsizer.io/simple

Connexion Kafka & HDFS

1. Utilisation d'un connecteur Sink.
2. Description des données par le connecteur et division du travail en tâche attribué aux workers.
3. Copie de chaque tâche dans un sous-ensemble de données via Kafka



Démonstration (1)

```python
from kafka import KafkaConsumer
import json

topic='caillau'

consumer=KafkaConsumer(topic,
                       bootstrap_servers=['127.0.0.1:9092'],
                       value_deserializer=lambda m: json.loads(m.decode('utf-8')))

for message in consumer:
    print("%s:%d:%d: key=%s value=%s" % (message.topic,
          message.partition, message.offset, message.key, message.value))
```

Code du consumer Kafka

Code du producer Kafka

```python
from kafka import KafkaProducer
import random
from single_transaction_generator import generate_receiver_transaction
import json

producer= KafkaProducer(bootstrap_servers=['127.0.0.1:9092'],
                        value_serializer=lambda v: json.dumps(v).encode('utf-8'))

topic="caillau"

for i in range(10):

    data = generate_receiver_transaction(random.randint(1, 3))
    print(data)
    message = data
    producer.send(topic, message)

#Permet d'envoyer le message sans attendre que le buffer soit rempli
producer.flush()
```

Démonstration (2)

```
C:\Users\yohan\OneDrive\Bureau\Formations\Projets
λ python consumer_proj.py
caillau:0:592: key=None value={'account_receiver_name': 'Renault', 'country_receiver': 'FR', 'agency_id': 2, 'account_sender_name': 'Nissan', 'country_sender': 'JAP', 'amount': 339968, 'payment_type': 'transfer'}
caillau:0:593: key=None value={'account_receiver_name': 'CapGemini', 'country_receiver': 'FR', 'agency_id': 3, 'account_sender_name': 'EDF', 'country_sender': 'FR', 'amount': 117633, 'payment_type': 'transfer'}
caillau:0:594: key=None value={'account_receiver_name': 'Monoprix', 'country_receiver': 'FR', 'agency_id': 1, 'account_sender_name': 'XXXX', 'country_sender': 'SWI', 'amount': 9677, 'payment_type': 'cheque'}
caillau:0:595: key=None value={'account_receiver_name': 'Thales', 'country_receiver': 'FR', 'agency_id': 3, 'account_sender_name': 'XXXX', 'country_sender': 'SWI', 'amount': 533964, 'payment_type': 'card'}
caillau:0:596: key=None value={'account_receiver_name': 'Thales', 'country_receiver': 'FR', 'agency_id': 3, 'account_sender_name': 'Tesla', 'country_sender': 'US', 'amount': 353959, 'payment_type': 'transfer'}
caillau:0:597: key=None value={'account_receiver_name': 'Valeo', 'country_receiver': 'FR', 'agency_id': 2, 'account_sender_name': 'XXXX', 'country_sender': 'SWI', 'amount': 234170, 'payment_type': 'card'}
caillau:0:598: key=None value={'account_receiver_name': 'Carrefour', 'country_receiver': 'FR', 'agency_id': 1, 'account_sender_name': 'EDF', 'country_sender': 'FR', 'amount': 924297, 'payment_type': 'transfer'}
caillau:0:599: key=None value={'account_receiver_name': 'Valeo', 'country_receiver': 'FR', 'agency_id': 2, 'account_sender_name': 'Tesla', 'country_sender': 'US', 'amount': 763317, 'payment_type': 'transfer'}
caillau:0:600: key=None value={'account_receiver_name': 'Thales', 'country_receiver': 'FR', 'agency_id': 3, 'account_sender_name': 'Nissan', 'country_sender': 'JAP', 'amount': 855616, 'payment_type': 'transfer'}
caillau:0:601: key=None value={'account_receiver_name': 'Monoprix', 'country_receiver': 'FR', 'agency_id': 1, 'account_sender_name': 'Tesla', 'country_sender': 'US', 'amount': 229240, 'payment_type': 'transfer'}
```

Sortie du consumer

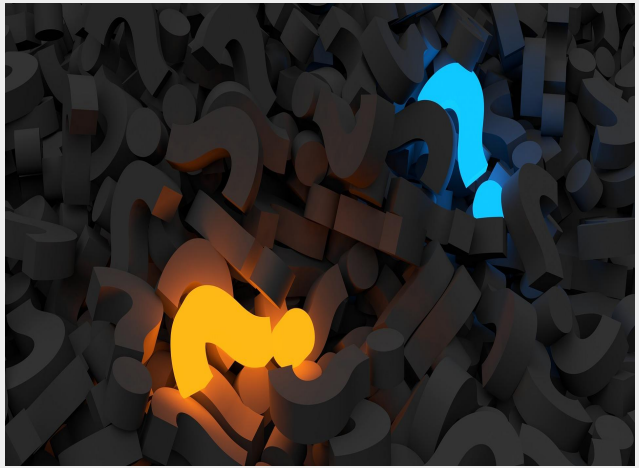
```
C:\Users\yohan\OneDrive\Bureau\Formations\Projets
λ python producer_proj.py
{'account_receiver_name': 'Monoprix', 'country_receiver': 'FR', 'agency_id': 1, 'account_sender_name': 'EDF', 'country_sender': 'FR', 'amount': 939936, 'payment_type': 'transfer'}
{'account_receiver_name': 'Renault', 'country_receiver': 'FR', 'agency_id': 2, 'account_sender_name': 'Tesla', 'country_sender': 'US', 'amount': 356324, 'payment_type': 'transfer'}
{'account_receiver_name': 'Carrefour', 'country_receiver': 'FR', 'agency_id': 1, 'account_sender_name': 'EDF', 'country_sender': 'FR', 'amount': 329886, 'payment_type': 'transfer'}
{'account_receiver_name': 'Valeo', 'country_receiver': 'FR', 'agency_id': 2, 'account_sender_name': 'Nissan', 'country_sender': 'JAP', 'amount': 119089, 'payment_type': 'transfer'}
{'account_receiver_name': 'Carrefour', 'country_receiver': 'FR', 'agency_id': 1, 'account_sender_name': 'Nissan', 'country_sender': 'JAP', 'amount': 607491, 'payment_type': 'transfer'}
{'account_receiver_name': 'Renault', 'country_receiver': 'FR', 'agency_id': 2, 'account_sender_name': 'XXXX', 'country_sender': 'US', 'amount': 474619, 'payment_type': 'cheque'}
{'account_receiver_name': 'CapGemini', 'country_receiver': 'FR', 'agency_id': 3, 'account_sender_name': 'Nissan', 'country_sender': 'JAP', 'amount': 856824, 'payment_type': 'transfer'}
{'account_receiver_name': 'Thales', 'country_receiver': 'FR', 'agency_id': 3, 'account_sender_name': 'XXXX', 'country_sender': 'US', 'amount': 707311, 'payment_type': 'card'}
{'account_receiver_name': 'Carrefour', 'country_receiver': 'FR', 'agency_id': 1, 'account_sender_name': 'EDF', 'country_sender': 'FR', 'amount': 743860, 'payment_type': 'transfer'}
{'account_receiver_name': 'Carrefour', 'country_receiver': 'FR', 'agency_id': 1, 'account_sender_name': 'XXXX', 'country_sender': 'US', 'amount': 823171, 'payment_type': 'card'}
```

Génération d'une transaction

04 Conclusion

Rappel de la problématique

- Accroissement du nombre de données et nécessité de centraliser les données
- Mise en place d'une architecture Big Data
- Répondre à des besoins de visualisation des données pour différents acteurs (clients, directeurs d'agence, conseillers bancaires, business analysts)



Limites

- **HDFS** : Problème de latence d'accès ; impossibilité d'écriture par plusieurs utilisateurs en même temps ; peu pratique pour une grande quantité de petits fichiers
- **MapReduce** : Opération limitée
- **Kafka** : Peu d'outils de monitoring ; réduction de performances (compression/décompression)

Perspectives

- Utilisation de Spark pour avoir accès à un large panel d'opération
- Création d'un entrepôt de données se basant sur HDFS via Hive par exemple
- Ajouter des outils d'ingestion supplémentaires dans le cas d'ajouts de source de données
- Mise en place de différents pipelines pour la partie visualisation et de futurs projets de ML

Merci pour votre attention !

Des questions ?