Université Paris Cité

Master 2 Informatique Parcours RSA

2023/2024

Traitement de flux de données (streaming)

Rapport de projet :

« Topic 2 : IoT Sensor Data Processing »

Réalisée par :

Jules CHERION – Mathéo DESSAUVAGES – Ibrahima DRAME – Idriss KOURBANHOUSSEN

Encadrant : **Hoang Nhat Minh NGUYEN**

I. Présentation du sujet

L'essor des technologies liées à l'Internet des objets (IoT) a complètement changé notre façon de surveiller et de comprendre notre environnement. Cette transformation s'accompagne par l'utilisation en temps réel des données recueillies par des capteurs spécialisés de l'IoT, tels que ceux qui mesurent les données météorologiques. Ces données ne sont plus simplement des nombres abstraits, ce sont des éléments clés qui nous aident à anticiper des événements météorologiques, comme des catastrophes naturelles. Il est donc crucial de traiter ces données en temps réel afin d'assurer une réponse plus adaptée que ce soit pour aider la population dans leurs quotidiens ou pour les prises de décisions des gouvernements.

C'est dans ce contexte que nous avons choisis de traiter le sujet 2, qui consiste à utiliser des données de capteurs IoT, telles que la température et l'humidité, et à les acheminer efficacement vers **Kafka** en flux continu d'informations.

En utilisant **Spark Streaming**, nous avons essayé de traiter ces données en temps réel. Notre objectif est de mettre en œuvre des mécanismes qui permettent de déclencher des notifications en cas de dépassement de seuils spécifiques. Cela signifie que nous ne ferons pas que collecter des données, mais nous cherchons également à obtenir des informations instantanées et pertinentes qui pourraient indiquer des conditions exceptionnelles ou des événements significatifs dans l'environnement surveillé.

En résumé, ce projet montre à quel point il est crucial de gérer en temps réel les données provenant des capteurs IoT. On utilise Kafka pour collecter ces données et Spark pour les traiter.

II. Données utilisées

Parlons maintenant du choix des données, nous avons fait le choix d'utiliser des données issues d'API afin de traiter des données réelles de différents capteurs IOT en France et dans le monde.

Nous avons utilisé des données de 2 API différentes, la première est issue d'OpenWeatherMap qui est un service en ligne, propriété d'OpenWeather Ltd, qui fournit des données météorologiques mondiales via API, notamment des données météorologiques actuelles, des prévisions, des prévisions immédiates et des données météorologiques historiques. Afin d'envoyer ces données sur un topic Kafka, nous avons tout simplement fait une requête vers l'API en précisant le nom de la ville dont nous voulions les données ainsi que notre clé d'accès.

La deuxième API que nous avons utilisée provient d'**Infoclimat**, une association à but non lucratif qui met à disposition sur son site les données de plusieurs stations météorologiques collaborant avec eux. Les données récupérables via l'API, en revanche, ne concernent que des stations de particuliers ou d'organismes décidant de participer (par exemple, les données de Météo France ne sont pas disponibles dans l'API). Ces données diffèrent car elles ne contiennent pas exactement les mêmes champs que celles de la première API. Nous récupérons toutes les données disponibles en Île-de-France (soit 34 stations) sur une plage horaire définie. Une fois reçues, elles sont adaptées afin que les deux puissent être utilisées de manière identique. Ensuite, elles sont envoyées dans Kafka pour simuler un flux continu de données.

OpenWeather

INFOCLIMAT

- Format des données
  - -Location
  - -Température
  - -Température_ressentie
  - -Température_min O
  - -Température_max O
  - -Pression
  - -Humidité
  - -Vent
  - -Description O
  - -Qualité_air A
  - -Timestamp
  - -Pluie_1h I
  - -Pluie_3h I

O: Openweather
I: Infoclimat
A: Aléatoire

III. Analyse des données

Dans le Producer, après avoir lu les données du topic Kafka, nous avons appliqué différentes requêtes sur ces données telles que les risques de pluie, le risque de chaleur élevé, le risque de gel, l'intensité des rafales, …

Nous vous donnerons dans ce rapport un exemple qui va décrire une requête mais puisque les requêtes sont faites de la même manière, cela vous permettra de comprendre aussi les autres requêtes.

1) On récupère les données puis on définit un schéma qui donne un sens à ces données.

```
# Définir le schéma
schema = StructType([
    StructField("location", StringType(), True),
    StructField("temperature", FloatType(), True),
    StructField("temperature_ressentie", FloatType(), True),
    StructField("temperature_min", FloatType(), True),
    StructField("temperature_max", FloatType(), True),
    StructField("pression", FloatType(), True),
    StructField("humidite", StringType(), True),
    StructField("vent", FloatType(), True),
    StructField("description", StringType(), True),
    StructField("qualite_air", FloatType(), True),
    StructField("timestamp", StringType(), True),
    StructField("pluie_3h", FloatType(), True),
    StructField("pluie_1h", FloatType(), True),
])

    stream_data:DataFrame = lines.select(
        F.from_json(lines.value, schema).alias("data")
    ).select("data.*")
```

2) On crée pour chaque requête que l'on va faire une colonne dans laquelle on va donner les conditions qui font que l'on obtient ce qu'on recherche dans les données.
   Dans la l'image ci-dessous on cherche à savoir s'il risque de pleuvoir donc on vérifie d'abord si le temps est pluvieux ensuite que le taux d'humidité est élevé c.-à-d. au-dessus de 70% et que le temps est nuageux.

```
result = stream_data\
            .withColumn("risque_pluie", when((col('humidite') >= 70) & (col('description').like('%cloud%')), "probable")
                        .when((col('description').like('%rain%')), "probable")
                        .otherwise("aucun")) \
```

IV. Notification des résultats

L'envoi de notification se fait par le biais du module *Tkinter* qui crée une fenêtre dans laquelle il est possible d'ajouter du texte. Une notification est envoyée après une opération de traitement si le seuil pour une nature de risque donné est atteint par le dernier rapport météorologique reçu. Si le risque d'une nature donnée déterminé pour le dernier rapport météorologique traité n'indique "aucun risque" alors aucune notification ne sera envoyée, dans le cas contraire une notification sera envoyée en précisant le niveau de risque et la nature.


```
query = result \
    .writeStream\
    .outputMode('complete')\
    .format('console')\
    .queryName("result_table") \
    .trigger(processingTime="5 second")\
    .foreach(lambda row: send_notification(row.asDict()))\
    .start()
```

V. Bibliographie

- Documentation Kafka : https://kafka.apache.org/documentation/
- Documentation Spark Streaming: https://spark.apache.org/docs/latest/streaming-programming-guide.html
- Documentation Python : https://docs.python.org/3/
- API pour les données météorologiques : https://openweathermap.org/api
- API pour les données météorologiques : https://www.infoclimat.fr/api-previsions-meteo.html?id=2988507&cntry=FR
- Cours magistraux et travaux dirigés de l'UE *"Traitement de flux de données (streaming)"*