Algoritmos para Big Data
Lab class handout #6

Data streaming with Apache Spark and Apache Kafka **2024/2025**

Learning objectives

This lab class is about processing streams of data. It relies on the structured streaming library of Apache Spark and builds upon the previous lab class handout, particularly in the part related to Apache Kafka as a messaging system.

Note: It is acceptable that the work to be undertaken may require some time beyond the class. If that is the case, work should continue outside of class. And if needed, assistance will be provided during the office hours established or to be agreed with the lecturer.

Supporting information

- Course slides
- Apache Spark – Structured streaming programmimg guide
- Apach Spark structured streaming and Apache Kafka integration guide
- Apache Kafka
- Streamlit

Introduction

A key aspect of a data streaming process is, besides the transforming operations on data, how to manage the flow of incoming data (from sources) and outgoing data (to sinks).
Apache Spark provides methods to read/write from/to a stream, accordingly to some formats we may select from. For example, there are the usual file-based formats like json, parquet, csv, text, and so on. We can also use socket connections to get/send text data from/to TCP servers and, more importantly, to use advanced messaging systems like Apache Kafka, which can play a sort of buffering role in this context. On the other hand, we have to set an output mode to define how the results of data operations will be delivered. For instance, whether to deliver all every time, or just the updates, etc.

In this lab, we will be using data connectors to Apache Kafka, supported by the work done in the previous lab class. The processing of the streaming data will be managed by Apache Spark structured streaming. Additionally, we will be using Streamlit [1] to present results in a more friendly way, as if it were a data app.
As for the the dataset to be used, it will be the same as in the previous lab class handout regarding Apache Kafka:
`https://bigdata.iscte-iul.eu/datasets/books-amazon.zip`

Finally, alongside this handout describing the tasks to accomplish, Python files are provided for the implementation of the tasks.

[1]Streamlit is an open-source Python framework for data scientists to deliver dynamic data apps with a few lines of code.

Tasks to accomplish

A. Apache Kafka cluster

In this task, the goal is to have an Apache Kafka cluster running and sending a data stream from the data file mentioned above, and related to a particular topic.

Hence, start by setting up and running a Kafka cluster, based on the guidelines introduced in the previous related lab class. If needed and advisable, change the Python code that was provided. Then, execute the following operations:

1. After setting up and running the Kafka cluster, check if it is working properly, by using the command:
   ```
   $ python3 kafka_cluster_info.py
   ```
2. Create a new topic (different from the ones you may have created in the past) by running:
   ```
   $ python3 kafka_cluster_create_topic.py
   ```
3. In one terminal, execute the following command to send/produce the data stream, as json data (instead of byte array):
   ```
   $ python3 kafka_producer.py
   ```
4. Likewise, and just for the purpose of checking if everything is all right, in another terminal execute the command:
   ```
   $ python3 kafka_consumer.py
   ```
5. Once tests were made, execute **Crtl-C** in the terminals to terminate both the running processes of sending/producing data and consuming data.

B. Data stream processing

In this task, the goal is to subscribe the topic that was created in the previous task, and to process incoming stream of data with Apache Spark. Hence, giving the Python code that was provided, execute the following operations:

1. Create a new directory, to be called `Application`, and there upload the files `app_spark.py` and `data_streaming.py`.
2. Check carefully the code written in the files that were just uploaded.
3. In a terminal, send/produce the data stream as before. [2]
4. In another terminal, run the command below to process the incoming data stream:
   ```
   $ python3 app_spark.py
   ```
5. Adjust the code in the file `app_spark.py` to incorporate aggregated information about the fields *Group* and *Format* combined together. Then run again the code.
6. Once experiments were made, make sure the running process of sending/producing data is stopped.

[2]Notice that the new data stream will join previous data that was sent/produced, so one has to be aware of that when analysing results. We leave possible working solutions as an exercise.

C. Data app with Streamlit

In this task, we aim to present the data processing results from the previous task but in a more friendly way, as if it were an app. Hence, execute the following operations:

1. In a terminal, install Streamlit:

   ```
   $ pip install streamlit
   ```

2. As suggested in the documentation of Streamlit, validate the installation by running the *Hello* app in the terminal:

   ```
   $ streamlit hello
   ```

3. Explore the running of the *Hello* app in the browser, e.g., its functionalities and related code.
4. Upload the file `app_streamlit.py` in the directory `Application` and check its code carefully.
5. In a terminal, send/produce the data stream as before. [3]
6. In another terminal, run the command below to process the incoming data stream and to visualize the results in the browser:

   ```
   $ streamlit run app_streamlit.py
   ```

7. Once the visual exploration of results is finished, make sure the running process of sending/-producing data is stopped.

D. Shut down the Apache Kafka cluster

Finally, in this task we are going to terminate the Kafka environment. Hence:

1. Stop the processes that are still running, by executing **Crtl-C** in the terminals of concern. Notice that this also includes the Kafka broker.
2. Run the following command to delete any data of the local Kafka environment:

   ```
   $ rm -rf /tmp/kafka-logs /tmp/kraft-combined-logs
   ```

[3]Recall previous footnote about sending again.