Monash University

FIT5202 - Data processing for Big Data

Assignment 2B: Real-time stream processing on big data

Due Date : Sunday, 17th October 2021 11:00 pm

Weight : 10% percent of the total mark

Background

Required datasets (available in Moodle):

- A compressed file **flight-delays.zip.**
- This zip file consists of 21 csv files and a metadata file:
    - csv files
    - airports.csv (we do not use it)
    - metadata.pdf
- Note that in this assignment, the original flights.csv has been sliced and reduced into several csv files in order to accommodate the hardware limitations.
- The complete dataset also can be downloaded publicly at https://www.kaggle.com/usdot/flight-delays

Information on Dataset

The flight-delays and cancellation data was collected and published by the U.S. Department of Transportation's (DOT) Bureau of Transportation Statistics. This data records the flights operated by large air carriers and tracks the on-time performance of domestic flights. This data summarises various flight information such as the number of on-time, delayed, cancelled, and diverted flights published in DOT's monthly in 2015.

Assignment Information

This assignment consists of three parts:

- **Task 1:** Kafka Producer - Producing the streaming data, where you can use csv modules to read and publish the data to the Kafka stream.
- **Task 2:** Kafka Consumer - Consuming the streaming data using Kafka consumer, use the csv module in any python libraries (e.g. Pandas) to process the ingested data from Kafka.
- **Task 3:** Streaming application - Using Spark Structured Streaming together with Spark ML/SQL to process data streams. In task 3, pandas can only be used for plotting steps. The excessive usage of Pandas for data processing is not recommended.

You need to simulate the streaming data production using Kafka, then show some basic streaming classification to display the accumulated average of accuracy (***accumMeanAccuracy***) and total number of flight records for each timestamp (***countFlightRecords*)** after consuming the data. Build a streaming application that integrates the machine learning model (provided to you) that can classify the flight-delays data stream.

Getting Started

- Download the datasets from Moodle namely **flight-delays.zip**.
- There is no template for this assignment, please organize your answer in order so that it is easy to mark.
- You will be using Python 3+ and PySpark 3+, Kafka (kafka-python), and any other python libraries for this assignment such as : numpy, pandas, scipy, and matplotlib. Consult the tutors or ask Ed Forum if you are using other packages.
- Create an Assignment-2B-Task1_flight_producer.ipynb file for data production
- Create an Assignment-2B-Task2_flight_consumer.ipynb file for consuming process data using Kafka
- Create an Assignment-2B-Task3_streaming_application.ipynb file for

consuming and processing data using Spark Structured Streaming

1. Producing the streaming data (30%)

In this section, you will need to implement an Apache Kafka producer to simulate the real-time streaming of the data.

Important:

- In this task, you need to generate the event timestamp in UTC timezone for each data record in the producer, and then convert the timestamp to unix-timestamp format (keeping UTC timezone) to simulate the "ts" column. For example, if the current time is 2021-9-28 12:39:45 UTC, it should be converted to the value of ***1632796806***, and stored in the "ts" column
- Please do not use Spark in this task

Event Flight Producer

Write a python program that loads all the data from "***flight*.csv***". Save the file as **Assignment-2B-Task1_flight_producer.ipynb**.

Your program should send X number and Y number of records from **each producer following the sequence** to the Kafka stream **every 5 seconds.** X represents the records to send in a particular batch, whereas Y represents the records to send in the next batch (pending records). The sequence of the batch is depicted in Fig.1.

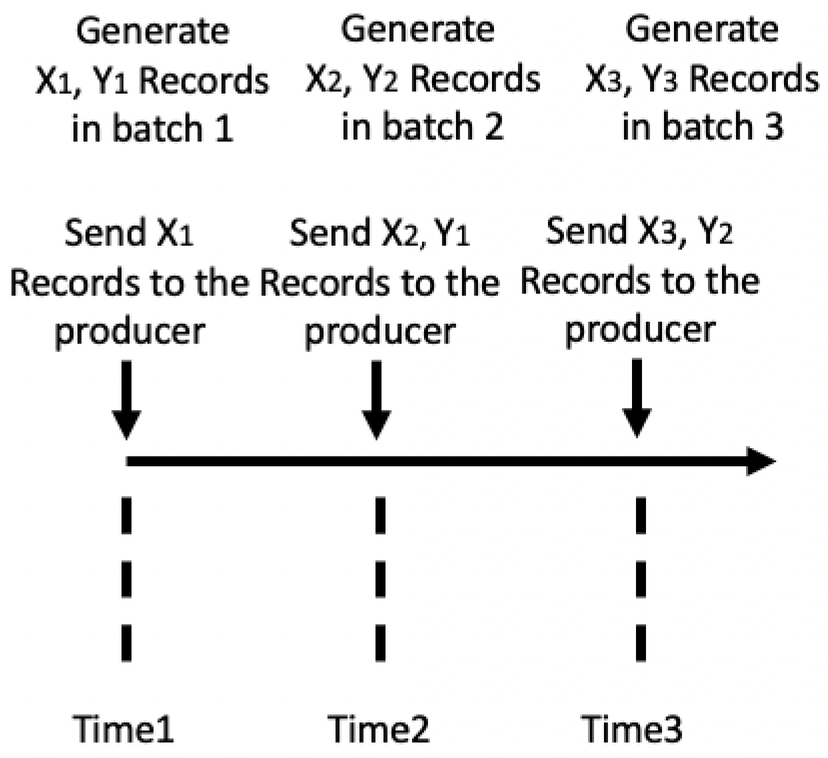
Generate X1, Y1 Records in batch 1
Generate X2, Y2 Records in batch 2
Generate X3, Y3 Records in batch 3
Send X1 Records to the producer
Send X2, Y1 Records to the producer
Send X3, Y2 Records to the producer
Time1
Time2
Time3

Fig.1. Sequence of data generation in flight-data into stream

There are some steps need to be carried out for this task:

1. Generate random numbers A and B, whose values are between 70~100 (inclusive) and 5~10 (inclusive) respectively, which are regenerated for each *keyFlight*. The *keyFlights* are generated from the column 'DAY_OF_WEEK' in the dataset which has 7 unique keys. These values 1, 2, 3, 4, 5, 6, and, 7 represents 'sunday','monday', 'tuesday', 'wednesday','thursday','friday','saturday'
2. You will need to append event time in unix-timestamp format (as mentioned above) for each key. Assuming that there are 7 keys in the flight-dataset as mentioned above, there will be 7 unix-timestamp for each batch.
3. Each batch data contains 7 group (7 sub batches) instances generated from each key. All of them are concatenated in the form of the list of dictionaries.
   a. Explanation of a group instances/records
      i. If A1 represents a group of instances/records generated from key = '1' and B1 represents a group of pending instances/records generated from key = '1', thus batch 1 (X1) contains [A1; A2; A3; A4; A5; A6; A7] and batch 1 pending (Y1) contains [B1; B2; B3; B4; B5; B6; B7].
      ii. A1 and B1 have the same **'ts'** as it is generated from the same batch at

the same time. The same case is also the same for A_2 and B_2 and so on.

iii. Given random numbers *A* and *B*, the number of instances in A_1, A_2 and B_1, B2 and so on vary.

b. Explanation of a dictionary.

Dictionary represents an instance of data which output can be seen as follow

i. Example of a dictionary with key = '1' {'ts':***1632796806,..,*** 'DAY_OF_WEEK':1, 'month':1,...}

ii. Example of a dictionary with key = '7' {'ts':***1632744322,...,*** 'DAY_OF_WEEK':7, 'month':3,...}

iii. Dictionary is a part of a sub batch data. A sub batch data is a part of a batch data X. This also applies for pending data B.

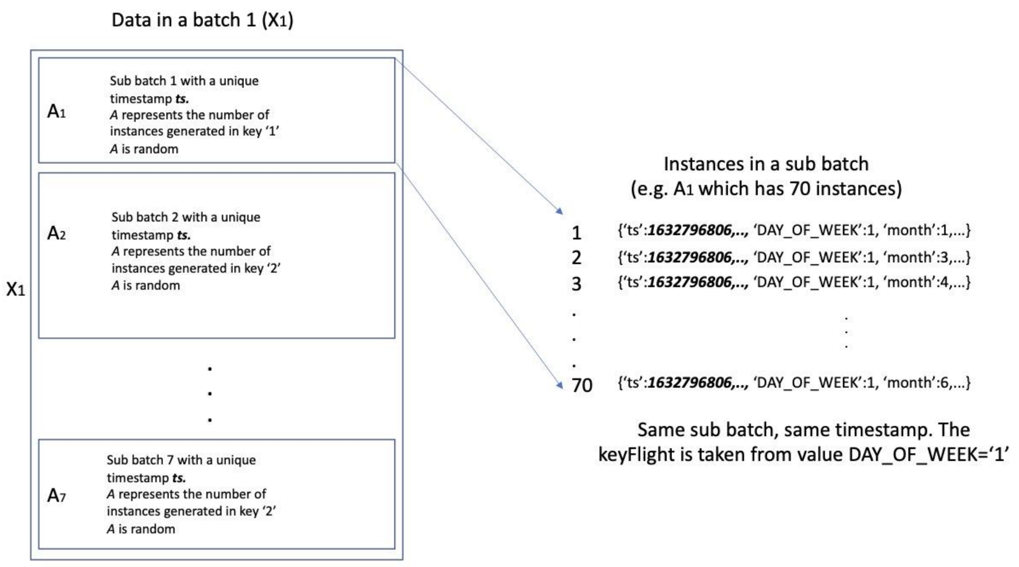
Data in a batch 1 (X1)
X1
A1
Sub batch 1 with a unique timestamp ts.
A represents the number of instances generated in key '1'
A is random
A2
Sub batch 2 with a unique timestamp ts.
A represents the number of instances generated in key '2'
A is random
A7
Sub batch 7 with a unique timestamp ts.
A represents the number of instances generated in key '2'
A is random
Instances in a sub batch (e.g. A1 which has 70 instances)
1 {'ts':1632796806,.., 'DAY_OF_WEEK':1, 'month':1,...}
2 {'ts':1632796806,.., 'DAY_OF_WEEK':1, 'month':3,...}
3 {'ts':1632796806,.., 'DAY_OF_WEEK':1, 'month':4,...}
70 {'ts':1632796806,.., 'DAY_OF_WEEK':1, 'month':6,...}
Same sub batch, same timestamp. The keyFlight is taken from value DAY_OF_WEEK='1'

Fig.2. Illustration of sub batches data (A_1 to A_7) in a batch data 1 (X_1)

4. Simulation of data stream in Fig.1.

   a. At time1: X_1 and Y_1 are generated on the producer side, but only X_1 is sent.

   b. At time2: X_2 and Y_2 are generated on the producer side, but only X_2 and pending data from the previous batch (Y_1) are sent to the consumer.

5. If the data in each key is exhausted, restart from the first sequence again.

6. Pseudocode for this task:

   a. Take the *DAY_OF_WEEK* column as the key, name a variable *KeyFlights* which contains the set of keys (7 keys).

b. Create a function *getFlightRecords*, which returns a variable named *flightRecords***,** which is a dictionary that contains all flight data with their associated keys (step 3).

c. Create a topic called *'flightTopic'*

d. Create an instance variable called *'flightProducer'*

e. for each *keyFlight* in *KeyFlights*

   i. Generate A['*keyFlight*'] and B['*keyFlight*'] and give both the timestamp as formatted in 3.b or Fig.2.

   ii. Concatenate all A and B. It will form the data batch X and Y respectively, see Fig.2.

f. Send X and Y to the consumer following the rule in step 4 or Fig.1.

2 Consuming data using Kafka (10%)

In this task, we will implement multiple Apache Kafka consumers to consume the data from task 1.

Important:

- **In this task, use Kafka consumer to consume the data from task 1.**
- **Do not use Spark in this task**

Event Flight Consumer

Write a python program that consumes the process events using kafka consumer, visualising the ***countFlightRecords*** in real time based on the timestamp. Save the file as **Assignment-2B-Task2_flight_consumer.ipynb**.

Your program should get the count of ***countFlightRecords*** captured by the consumer in the last 2-minutes (**use the processing time**) for each ***keyFlight***. For this, print the number of flights for keyFlight = '1', keyFlight = '2', and keyFlight = '3' only. Then, use line charts to visualize it. Note, in this task, please use ts as timestamp generated from producer step (disregard DAY, DAY_OF_WEEK, MONTH, and YEAR as the real timestamp)

- Hint - x-axis can be used to represent the timestamp, while y-axis can be used to represent the number of ***countFlightRecords*** data can be represented in different color legends.

3. Streaming application using Spark Structured Streaming (40%)

In this task, we will implement **Spark Structured Streaming** to consume the data from task 1 and perform streaming classification.

Important:

- **You may use Spark Structured Streaming together with Spark SQL and ML.**
- **You are also provided with a set of pre-trained pipeline models to classify the binary classification flight-delays (to be provided soon).**

Write a python program that achieves the following requirements. Save the file as **Assignment-2B-Task3_streaming_application.ipynb.**

1. **SparkSession is created** using a SparkConf object, which would use two local cores with a proper application name, and use UTC as the timezone 3.
2. From the Kafka producers in Task 1, ingest the streaming data into Spark Streaming.
3. Then the streaming data format should be transformed into the proper formats following the file schema in the metadata.
4. Persist the transformed streaming data in parquet format for flight data. Flight data should be stored in "flight.parquet" in the same folder of your notebook.
5. Load the machine learning models given, and use the models to classify whether each flight records are delayed. This is based on the assumption that the data has been labelled.
6. Using the classification results, monitor the data following the requirements below. For each key in keyFlight = '1', keyFlight = '2', and keyFlight = '3', keep track of the accumulated accuracy for every timestamp in the 2-min window for a total of 6 minutes.
    i. Your results should include, number of records flight and for each key, including their accumulated accuracy in each timestamp.
    ii. Visualise the data in line charts. Prepare a line chart plot to show the number of flights from the start to the most recent.
    For this visualisation, You need two subplots. First subplot, the x-axis can be used to represent the timestamp, while y-axis can be used to represent the number of ***countFlightRecords.*** For the second subplot,

x-axis can be used to represent the timestamp, whereas y-axis can be used to plot the ***accumMeanAccuracy*.** For each subplot, the results from all **keyFlights** (key = ‘1’, key = ‘2’, and key = ‘3’) should be represented in different color legends.

Assignment Marking

The marking of this assignment is based on the quality of work that you have submitted rather than just quantity. The marking starts from zero and goes up based on the tasks you have successfully completed and its quality, for example how well the code submitted follows programming standards, code documentation, presentation of the assignment, readability of the code, reusability of the code, and organisation of code.

Submission Requirements

You should submit your final version of the assignment solution online via Moodle; You must submit the following:

- **A PDF** file (created from the notebook) to be submitted through Turnitin submission link. Use the browser’s print function to save the notebook as PDF. Please name this pdf file based on your authcate name (e.g. **psan002.pdf**)
- **A zip file** of your Assignment-2A folder, named based on your authcate name (e.g. **psan002.zip**). This should be a ZIP file and **not** any other kind of compressed folder (e.g. .rar, .7zip,   .tar). Please do not include the data files in the ZIP file. Your ZIP file should only contain Assignment-2A.ipynb

Where to Get Help

You can ask questions about the assignment on the Assignments section in the Ed Forum accessible from the on the unit’s Moodle Forum page. This is the preferred venue for assignment clarification-type questions. You should check this forum regularly, as the responses of the teaching staff are "official" and can constitute amendments or additions to the assignment specification. Also, you can visit the consultation sessions if the problem and the confusions are still not solved.

Plagiarism and Collusion

Plagiarism and collusion are serious academic offences at Monash University. Students must not share their work with any other students. Students should consult the policy linked below for more information.

https://www.monash.edu/students/academic/policies/academic-integrity

See also the links under *Academic Integrity Resources* in Assessments block on Moodle.

Students involved in collusion or plagiarism will be subject to disciplinary penalties, which can include:

- The work not being assessed
- A zero grade for the unit
- Suspension from the University
- Exclusion from the University

Late submissions

Late assignments will be penalised. If a student wants to apply for an extension up to 5 days, send an email to the lecturer (david.chengzarate@monash.edu) with the request and reasons to have an extension. Greater than 5 days extension requires Special Consideration which is submitted centrally. This means that students MUST submit an online Special Consideration form. For more details please refer to the **Unit Information** section in Moodle.

There is a **10% penalty per day including weekends** for the late submission.