Streaming Data Processing TD 5

Minh NGUYEN

November 27th, 2023

Abstract

In this programming assignment, we will continue to study the streaming data processing with Spark Structured Streaming and Kafka source.

1 Summary

2 Introduction

2.1 Prerequisites

- Programming language and SDK:
  - Python 3
  - Java 11/13/15/17
- IDE such as IntelliJ (Java) or VS Code / PyCharm with Python.
- Spark 3.5.0

3 Install Spark

3.1 Windows 11:

Follow this tutorial .

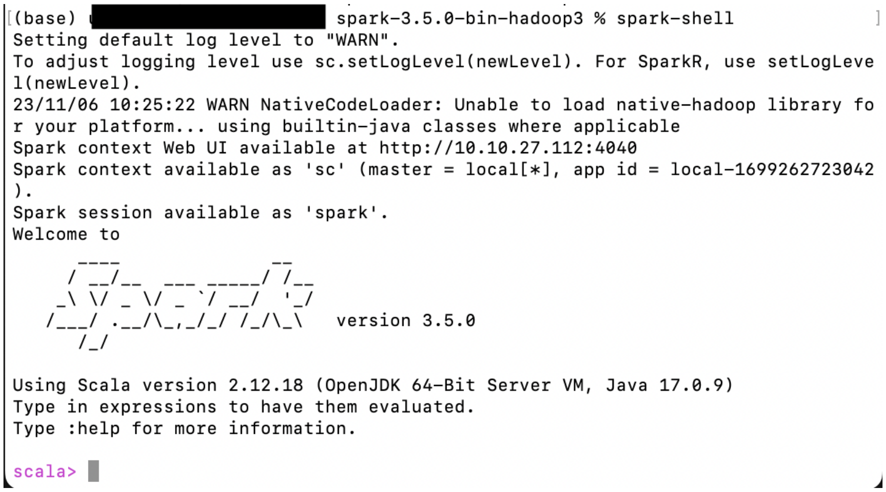
3.2 Other than Windows:

- Download Spark from this link: spark-3.5.0-bin-hadoop3.tgz.
- Extract the downloaded compressed file to a folder as you like.
- Config the PATH environment:
  - If you're using MacOS: do the following steps:
    1. Open˜/*.zshrc* by an editor (can use Vim)
    2. Add the following lines in the end of the file:

       ```
       export SPARK_HOME="<your-path-to-spark-folder>/spark-3.5.0-bin-hadoop3"
       export PATH="$SPARK_HOME:$SPARK_HOME/bin:$PATH"
       ```

    3. Apply this change by running this command in the terminal:

       ```
       source ~/.zshrc
       ```



```
(base) [redacted] spark-3.5.0-bin-hadoop3 % spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve
l(newLevel).
23/11/06 10:25:22 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Spark context Web UI available at http://10.10.27.112:4040
Spark context available as 'sc' (master = local[*], app id = local-1699262723042
).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.0
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 17.0.9)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Figure 1: Spark shell interface

4. Testing by opening another terminal and running this command. If Spark shell appears, then the installation can be said as done.

```
spark-shell
```

4 Setup environment for Python

A virtual environment is a directory that contains a specific collection of Python packages that you have installed. For example, you may have one environment with NumPy 1.7 and its dependencies, and another environment with NumPy 1.6 for legacy testing. If you change one environment, your other environments are not affected. You can easily activate or deactivate environments, which is how you switch between them. Working with separate environments help you:

- Profesionalize your workflow, in which Python dependencies can be fulfilled inside an environement
- Separate one environement from another, to avoid the dependencies conflict.

To create virtual environments, you can use either **venv** or **conda**. In this LAB, we will use conda.

4.1 Install Conda

- Install Miniconda from this link: https://docs.conda.io/projects/miniconda/en/latest/

4.2 Managing environments

Full list of tutorial can be found here :

Creating an environment with required packages (pyspark):

```
conda create -n <environment_name> python=3.10 pyspark
```

Activate the environment:

```
conda activate <environment_name>
```

5 LAB 5

5.1 Context

- In this LAB we will generate the data streams as requested, and publish them to Kafka cluster (as we did in Lab 4) in 2 different topics.
- Write another Spark Structured Streaming application for further analysis.
- Visualization using jupyter-dash is welcomed and there will be a bonus for this task.

5.2 Lab report

You are required to **submit a Lab report** for the following assignments:

EXERCISE 1: From TD4, you've already known how to connect to Kafka cluster and Kafka topic. If you don't remember, refer to TD4 and finish it.
For this question, the sample code is provided in TD5_sample.py. The context is as follow:
Do the following steps and write down your observation in Lab report:

- [**Code**] Write a program with **Kafka Producer** to publish to your Kafka topic at various rate from 1-10 message per second. The program should publish a pseudo e-commerce transactional data with **2 topics** as follow:

  1. **TD5_users**
     - user_id
     - name
     - country
  2. **TD5_transactions**
     - user_id
     - item_name
     - item_type
     - unit_price
     - amount
     - timestamp

  Attach the program code to your lab report.

  Now do the following steps as mentioned in the python sample file:

  1. Read the *usersstream* from Kafka:
     - kafka topic: *TD5_users*
     - Write stream to memory, and name this temporary table as user_stream. Given that df_stream is your stream:
       df_stream.writeStream.format("memory")
       .queryName(" < put_your_table_name > ")
  2. Same request to read *transactionstream* from Kafka topic *TD5_transactions*
  3. Answer the following questions by querying the data streams using Spark, and explain what you do:
     - Question 1: What is the proportion (in percentage) of users having the age under 30 buying something at the store in the last 30 minutes?
     - Question 2: Top 3 item types that have been bought in the last 10 minutes
     - Question 3: Top 3 item types that have been bought by the teenagers (from 13-19 year old) in the last 10 minutes
     - Question 4: Do we have any teenager buying alcohols in the last 30 minutes?
     - Question 5: Top 3 items having the highest revenue by age of buyers.
     - Question 6: Top 3 countries having the most item bought in our system.