Université Libre de Bruxelles
École polytechnique de Bruxelles

ULB ECOLE POLYTECHNIQUE DE BRUXELLES

INFO-H515 - Big Data

Part I : Distributed Data Management

Authors:
Maximin FOTSING KUETCHE
Vladimir KOZLOV
Julien RIZK
Aliaksei VAINILOVICH

Professors:
Dimitris Sacharidis
Antonios Kontaxakis

2022-2023

Contents

1 Introduction

This project focuses on the application of distributed data management techniques to analyse correlations within large time series. The data is taken from the Brussels Mobility Bike Counts API, with an interval of 15 minutes, for the period from 6 December 2018 to 31 March 2023.

The objective is to identify the five most correlated sensor pairs, using the Pearson correlation coefficient. The management of missing values in the data requires preliminary data preparation.

The project is divided into three parts: batch processing, continuous processing and sliding window processing. These tasks require the use of Spark for the calculation of correlation coefficients, the simulation of a streaming environment, and the implementation of a sliding window to calculate these coefficients.

The successful implementation of these techniques on large-scale real-time data is the main challenge of this project.

2 Batch Processing

At the architecture level, the present algorithm is designed to process a large amount of data coming from the Brussels Mobility Bicycle Counters API. The objective is to analyse the correlation between the different sensors through distributed data management techniques.

The architecture of the algorithm can be divided into several segments:

1. **Data download and acquisition:** Historical data for each sensor is downloaded from the API using GET requests. The device names are first retrieved as a JSON file. Then, for each device, the historical data is downloaded and saved as individual CSV files.

2. **Data mining and pre-processing:** The algorithm then checks the size of the data sets for missing values. The dates between the given range are also generated to allow correct temporal alignment of the data from each sensor.

3. **Merge and normalise the data:** All CSV files are merged into a single table, renaming the columns for each sensor. Missing values are replaced with predetermined values (count at 0, average speed at -1).

4. **Correlation calculation:** For each pair of sensors, the Pearson correlation coefficient is calculated for each 15-minute time step over the given date range. These calculations are performed for temporary data sets distributed throughout the day.

5. **Identification of the most correlated pairs:** Finally, for each timestamp, the algorithm determines the 5 sensor pairs with the highest correlation coefficients.

In sum, this algorithm provides a complete solution for downloading, pre-processing and analysing large-scale temporal data from multiple sensors, focusing on detecting correlation patterns.

3 Stream Processing

In our project, we used a flow processing process. Stream processing is a method of processing data that allows for real-time analysis of data as it arrives. The stream processing process consists of two main parts: the producer and the receiver.

3.1 Data Transmission

In our case, the streaming of data is regulated by two parameters, Π and Δ. Π represents the time period for which the measurements are sent in a single batch by the producer. Δ, on the other hand, represents the regular time interval after which the producer sends these batches of measurements. For example, if Π is 30 days and Δ is 5 seconds, the producer sends one month's data for each sensor every 5 seconds. The receiver then uses Spark Streaming to process these batches of data with a mini-batch interval corresponding to Δ.

3.2 Producer

The producer is responsible for sending real-time data to a server for a given time period. This data includes information from various sensors, including the number of events detected by each sensor and the average speed recorded by each sensor during that period. Here is an excerpt of the producer's output for Π = 30 days:

```
Sending data for period: 2023-03-15 00:00:00 - 2023-04-14 00:00:00
             Date  Time gap  Count_CAT17  Average_speed_CAT17  Count_CVT3
149760 2023-03-15         1            1                   11
149761 2023-03-15         2            2                   16
149762 2023-03-15         3            0                   -1
149763 2023-03-15         4            0                   -1
```

```
149764 2023-03-15         5             0                    -1

        Average_speed_CVT387  Count_CB1699  Average_speed_CB1699
149760                    -1             0                    -1  \
149761                    -1             0                    -1
149762                    -1             0                    -1
149763                    15             0                    -1
149764                    -1             0                    -1
```

3.3 Receiver

The receiver receives the data transmitted by the producer. It analyzes this data and performs various transformations and calculations. This includes the calculation of the Pearson correlation coefficient between the different sensors.

At the end of the processing, the receiver displays the five highest correlations for the number of events and the average speed recorded by each sensor. Here is an part of the receiver output for the first batch of data with $\Pi = 30$ days:

```
Top 5 correlations for Count:
CB1143 and CEK18: 0.7657636106436612
CB1143 and CB02411: 0.750007462332869
CB02411 and CEK18: 0.7341503240487556
CB02411 and CEV011: 0.70713147535422
CB1143 and CEV011: 0.694323348165607
Top 5 correlations for Average speed:
CB1143 and CB02411: 0.5962738164502057
CB1143 and CEK18: 0.5719238078866581
CB1143 and CEK049: 0.5710604482129724
CB02411 and CEK18: 0.5676817671321441
CB1143 and CEK31: 0.5447576602008446
```

In addition, the receiver checks if the data stream is empty, i.e. if there is no data available for processing. If this is the case, it interrupts the stream processing until new data is available.

3.4 Calculation of the Pearson correlation coefficient

The Pearson correlation coefficient is used to measure the strength and direction of the association between two continuous variables. In our case, we use it to measure the correlation between data from different sensors.

For example, the receiver calculates the Pearson correlation coefficient between the CAT17 sensor and the CB1101 sensor. To do this, it uses the data on the number of bikes captured and the average speed recorded by each sensor. If the correlation coefficient is close to 1, it indicates a strong positive correlation between the two sensors. If the coefficient is close to -1, it indicates a strong negative correlation. If the coefficient is close to 0, it indicates no correlation. It is important to note that while correlations may indicate a relationship between two sensors, they do not prove a causal relationship.

4 Sliding Window Processing

Architecturally, the code uses the PySpark library to analyse a real-time stream of bicycle count data in Brussels. The code implements a sliding window to process the data, which is a common way of analysing stream data in real-time processing.

1. **Initialization:** The code sets up necessary variables, including a list of bike sensors in Brussels (sensors), and defines a schema for incoming data. The schema comprises the date, time offset, bike count, and average speed per sensor.

2. **Spark Contexts:** The next step involves creating a Spark context (sc), the central interface for Spark operations, and a streaming context (ssc), which facilitates stream processing.

3. **DStream creation:** A DStream (lines), a series of Resilient Distributed Datasets (RDDs), is created from the socket port data. RDD is a core abstraction in Spark.

4. **Data transformation:** Subsequent processing removes headers and converts the data into a usable format. The transformed data is grouped into a window defined by window_duration and sliding_interval parameters.

5. **Sliding window:** The sliding window technique is used for data analysis. It considers a certain number of preceding data points (window_duration), moving by a specific interval (sliding_interval) at each iteration.

6. **Data processing:** The windowed data undergoes processing to calculate the Pearson correlation coefficients between sensor pairs. After sorting these coefficients, the top five pairs with the highest correlation are outputted.

Using windows in Spark Streaming enables efficient processing of data based on its timestamp and allows various aggregate operations within the window, such as calculating sums, averages, and extremes.

The script operates in two main parts: initially, it reads the incoming stream, processes it and organizes the data into a Sliding Window. Secondly, it converts the window data into a dataframe and applies a correlation function to it, as defined in the first part of the project. And here are the results:

```
Checking for empty...
Processing batch...
+-----------+------------+------------------+
|   Device 1|    Device 2|       correlation|
+-----------+------------+------------------+
|Count_CAT17| Count_CJM90|0.8777253555119251|
|Count_CAT17| Count_CB1142|0.8659487638547783|
|Count_CAT17| Count_CJE181|0.8438760598477546|
|Count_CAT17| Count_CB1699|0.8333281074422983|
|Count_CAT17|  Count_CEK31|0.8249462508321062|
+-----------+------------+------------------+
```