Flink Project

Vehicle Telematics

Description: Drivers, fleet owners, transport operations, insurance companies are stakeholders of vehicle monitoring applications which need to have analytical reporting on the mobility patterns of their vehicles, as well as real-time views in order to support quick and efficient decisions towards eco-friendly moves, cost-effective maintenance of vehicles, improved navigation, safety and adaptive risk management. Vehicle sensors do continuously provide data, while on-the-move, they are processed in order to provide valuable information to stakeholders. Applications identify speed violations, abnormal driver behaviors, and/or other extraordinary vehicle conditions, produce statistics per driver/vehicle/fleet/trip, correlate events with map positions and route, assist navigation, monitor fuel consumptions, and perform many other reporting and alerting functions.

In this project we consider that each vehicle reports a position event every 30 seconds with the following format: *Time, VID, Spd, XWay, Lane, Dir, Seg, Pos*

Being *Time* a timestamp (integer) in seconds identifying the time at which the position event was emitted,

VID is an integer that identifies the vehicle,

Spd (0 - 100) is an integer that represents the speed in mph (miles per hour),

XWay (0 . . .L−1) identifies the highway from which the position report is emitted

Lane (0 . . . 4) identifies the lane of the highway from which the position report is emitted (0 if it is an entrance ramp (ENTRY), 1 − 3 if it is a travel lane (TRAVEL) and 4 if it is an exit ramp (EXIT)).

Dir (0 . . . 1) indicates the direction (0 for Eastbound and 1 for Westbound) the vehicle is traveling.

Seg (0 . . . 99) identifies the segment from which the position report is emitted

Pos (0 . . . 527999) identifies the horizontal position of the vehicle as the number of meters from the westernmost point on the highway (i.e., Pos = x)

The goal of this project is to **develop a Java program using *Flink* implementing the following functionality:**

- **SpeedRadar:** detects cars that overcome the speed limit of 90 mph**.**

- **AverageSpeedControl:** detects cars with an average speed higher than 60 mph between segments 52 and 56 (both included) in both directions. If a car sends several reports on segments 52 or 56, the ones taken for the average speed are the ones that cover a longer distance.
- **AccidentReporter:** detects stopped vehicles on any segment. A vehicle is stopped when it reports at least 4 consecutive events from the same position.

Notes:

- All metrics must take into account the direction field.
- A given vehicle could report more than 1 event in the same segment.
- Event time must be used for timestamping.
- Cars that do not complete the segment (52-56) are not taken into account by the average speed control. For example 52->54 or 55->56.
- A car can be stopped on the same position for more than 4 consecutive events.
  An accident report must be sent for each group of 4 events. For example, the next figure shows 8 events for the car with identifier VID=3:

```
870,3,0,0,1,0,26,139158
900,3,0,0,1,0,26,139158
930,3,0,0,1,0,26,139158
960,3,0,0,1,0,26,139158
990,3,0,0,1,0,26,139158
1020,3,0,0,1,0,26,139158
1050,3,0,0,1,0,26,139158
1080,3,0,0,1,0,26,139158
```

  The accident reporter should generate 5 accident alerts. (870->960, 900->990, 930->1020, 960->1050, 990->1080).

Input: The Java program will read the events from a CSV with the format: *Time, VID, Spd, XWay, Lane, Dir, Seg, Pos*

An example file is available at https://dl.lsdupm.ovh/CLOUD/2122/sample-traffic-3xways.csv

Output to be generated:

The program must generate three output CSV files.

- speedfines.csv: stores the output of the ***Speed Radar*** functionality.
  - format: Time, VID, XWay, Seg, Dir, Spd
- avgspeedfines.csv: stores the output of the ***Average Speed Control*** functionality**.**
  - format: Time1, Time2, VID, XWay, Dir, AvgSpd, where *Time1* is the time of the first event of the segment and *Time2* is the time of the last event of the segment.
- accidents.csv: stores the output of the ***Accident Reporter***.

- format: Time1, Time2, VID, XWay, Seg, Dir, Pos, where *Time1* is the time of the first event the car stops and *Time2* is the time of the fourth event the car reports to be stopped.

Requirements:

The application must be developed using the versions of the software: Oracle Java 11, Maven, Flink 1.14.0.

The program must be optimized to run on a *Flink* cluster with **3 task manager slots** available. **The parallelism for the write operation to the files must be 1.**

The project must be configured with the **flink-quickstart-java** maven artifact.

The program of the project must be named master.**VehicleTelematics**, your application will be tested using the following procedure from the root folder of your project:

- mvn clean package -Pbuild-jar
- flink run -p 3 -c master.VehicleTelematics target/$YOUR_JAR_FILE $PATH_TO_INPUT_FILE $PATH_TO_OUTPUT_FOLDER

The input file and the output folder will exist on all nodes of the cluster running the *Flink* Task Managers.

Each project will be developed by a group of two students from the same master program.

Submission:

- **Deadline:** 1st December 2021 at 23:55
- **Only one submission per group**
- **Where: All the required files must be uploaded into Moodle by the deadline.** The file must be named **ID.zip** (**ID is the last name of the two members of the couple separated by a hyphen**). The structure of the file must be:
  - ID.zip
    - flinkProgram
      - pom.xml
      - src/

Evaluation:

- **Correctness and efficiency both in terms of space and time will be taken into account.**