# Cloud Computing and Big Data Ecosystems Design

## 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, which are processed in order to provide valuable information to stakeholders. Applications identify speed violations, abnormal driver behaviors, and/or other extraordinary vehicle machine 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*

Where *Time* is 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 (kms/h) of the vehicle,

*XWay, Lane, Dir, Seg and Pos* are the following functions over the vehicle's (x, y) coordinates:

- *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

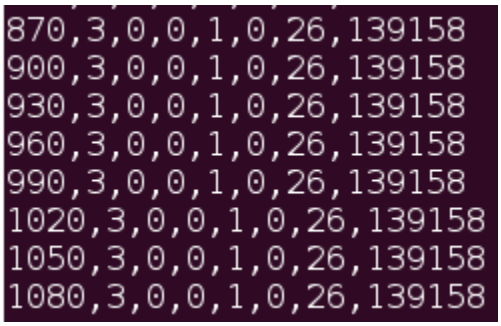
- *Seg* (0 . . . 99) identifies the segment from which the position report is emitted, and
- *Pos* (0 . . . 527999) identifies the horizontal position of the vehicle as the number of meters from the westernmost point on the highway (i.e., Pos = x)

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

- **Speed Radar:** detect cars that overcome the speed limit of 90 kms/h
- **Average Speed Control:** detects cars with an average speed higher than 60Km/h between segments 52 and 56 (both included) in both directions.
- **Accident Reporter:** 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 for 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:




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*

Output to be generated:

The program must generate 3 output CSV files

- speedfines.csv: to store the output of the *speed radar*

    - format: Time, VID, XWay, Seg, Dir, Spd
- avgspeedfines.csv: to store the output of the *average speed control*
    - 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: to store the output of the *accident detector*.
    - 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 8, Flink 1.3.2 and deployed using Ubuntu.

The program must be optimized to run on a *Flink* cluster with 10 task manager slots available.

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

The main class of the project must be named **master2017.flink.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 10 -c master2017.flink.VehicleTelematics $PATH_TO_INPUT_FILE $PATH_TO_OUTPUT_FOLDER target/$YOUR_JAR_FILE

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

**Submission**:

- **Deadline:** 22nd December 2017 at 23:55
- **Where: All the required files must be uploaded to Moodle by the deadline.** The file must be named **ID.rar** (ID is the id of the students provided by the instructor). The structure of your delivery will be:
    - ID.tar.gz
        - flinkProgram
            - pom.xml
            - src/
- **Groups:** The project is implemented by 2 persons of the same master program. Groups must be registered sending an email to mpatino@fi.upm.es and vvianello@fi.upm.es with subject: "master2017 flinkProject" by 30/11/2017. The group id will be assigned replaying to this email.
    - In the email you must provide: full name of the two students of the group and name of their master program.