Processamento de Streams

Lab. Assignment #2
(2017-2018)

The challenge for this second assignment is the same as for the first one. I.e., you are supposed to implement the queries of the DEBS Grand Challenges 2015.
The difference is that this time you should implement them using SiddhiQL of WSO2.

More specifically, you are supposed to:

- Use the Apache Kafka event producer developed in assignment #1 as an event producer for the WSO2 Complex Event Processor (see https://docs.wso2.com/display/CEP420/Kafka+Event+Receiver for instruction on the setup and installation). See also https://wso2-extensions.github.io/siddhi-io-kafka/api/latest/#kafka-source.
- Develop the relevant adapters in WSO2 for accessing the event streams
- Register the SiddhiQL queries that implement both queries plus the ones mentioned below

Queries

Besides the two queries defined in the DEBS Grand Challenge 2015 with the simplification defined below, you further must implement continuous queries for solving each of the following challenges:

- To control possible periods of the day when the number of taxis available is too much, the city wants to be alerted whenever the average idle time of taxis is greater than a given amount of time (say 10 minutes). The idle time of a taxi is the time mediating between the drop off of a ride, and the pickup time of the following ride. It is assumed that a taxi is available if it had at least one ride in the last hour.
- The city wants to know about the areas where, when the taxis enter there, the rides increase in their duration. For that, there should be alerts when a taxi has a peak in the duration of the ride that is followed by at least 3 rides all increasing in their duration. The alert should contain the location where the taxi started the ride which had the peak duration.
- To distinguish the most pleasant taxi drivers, it should be nice to have an event, emitted once a day, signalling the taxi driver with the highest total amount of tips in that day.

Since there is no aggregation function for the median in SiddhiQL, you can simplify the second query of the Grand Challenge to use the average rather than the median.

Delivery

You are supposed to deliver a short report (8 pages max) on your implementation to jmp@fct.unl.pt.

You are also supposed to show a little demo of the implementation. The demo should be done until the end of the semester, in a time arranged by mail convenient for both you and the lecturer. As soon as you have it, just contact by mail or in person to schedule it.

The deadline is the end of the semester, though there is no reason to leave it for the last minute…

Extras

As extras you can:

- Implement the second query of the Grand Challenge with the median rather than with the average. For this, you should implement a median aggregation function in SiddhiQL.
- In the above description of the assignment, nothing is said about how the detected event are presented. The simplest way is just to show them on the console. Of course, it would be much nicer to show them in a dashboard…
- If you decide to make a dashboard, you could include in it some monitoring feature. For example, a continuously updated graph which shows the number of taxi rides over the last hour (updated e.g. every minute), or a continuously updated histogram with the number of rides per sector, for the sector with the most rides at the moment, or even a map painting the sectors according to the "density" of taxis in it…
- You can also make a throughput evaluation, as you've done for the assignment #1. Of course, the advantage of using a DSMS or CEP system won't certainly be on efficiency. But you can test how far it goes
- As you'll find out, the most difficult part of this assignment won't be writing the queries. So, once you have it all set up, one more query won't certainly hurt…
  You may come up with further interesting queries on the Taxi's subject and implement them. Be creative!