Citi Bike share analysis

[Project Report - 061]

Aruna Kumaraswamy
Indiana University
akumaras@iu.edu

ABSTRACT

Purpose of this project is to analyze the NYC Citi Bike share system data in batch and streaming mode. Following report summarizes the finding based on the September 2016 bike share data. Report will also provide instructions on installing the necessary software, build and deploy the software created for this project.

1. INTRODUCTION

Apache Flink is used in this project to process the data in batch and streaming mode. There are 3 main components in this project

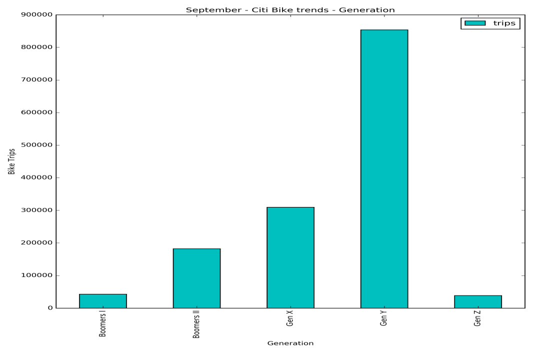
- BikeRentTrendByAge
- BikeRentTrendByGender
- LiveStationStatusStream

Components are built in Java. Maven is used for building the components.

1.1 Bike Rent Trend - By Age

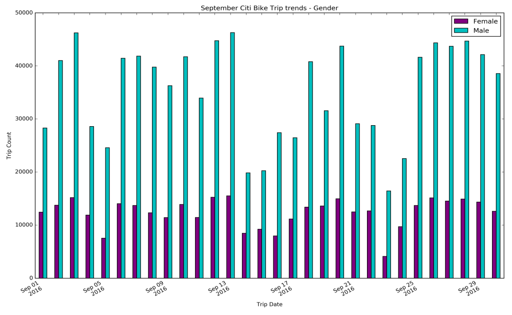
Objective of this analysis is to find in which age group bike share is popular. This component takes csv file as input. For age group analysis, we are interested only in the age, so we skip all the other columns in the spreadsheet. Based on the age group, we decide if the bike rider is Boomer I, Boomer II, Gen X, Gen Y or Gen Z. Invalid age in the data is filtered. Final result is the count of the age groups for September 2016. Following is the visualization for this batch program. Visualization is created using a python program.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235



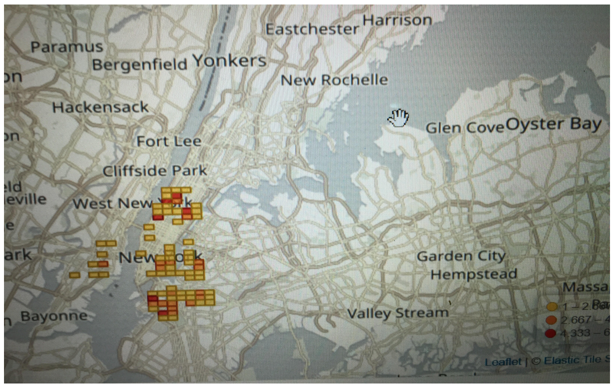
1.2 Bike Rent Trend - By Gender

Objective of this analysis is to find whether bike ride is popular among men or women. This component takes CSV file as input and creates a CSV output. We are interested in trip date, gender column of the spreadsheet, so we skip other columns. We also filter unknown gender. We transform the input data using tables and join them to produce the desired CSV output. Following is the visualization for this batch program. Visualization is created using python program.



1.3 Live Station Status Stream

Live station status feed is used to check the availability of bikes across all stations. If the bike availability of a station is less than 2 then those stations are filtered. Result is pushed to elastic search. Kibana is used for visualization of real-time alerts across geographic map.



2. TECHNOLOGY

Flink - Install flink version 1.1.3
Elastic search - Install version 2.3.5
Kibana - 4.5.4 (with appsense plugin)
Python 2.7

3. BUILD AND DEPLOY

Clone GIT repository

3.1 STEP1 Build Java code

Build java code using
mvn clean package

3.2 STEP2 Prepare data

Use wget or bitsadmin to get trips data from https://s3.amazonaws.com/tripdata/201609-citibike-tripdata.zip
Flink is unable to process ZIP directly. So unzip the data file.

3.3 STEP 3 Elastic Search Index

Start Kibana
Go to http://localhost:5601/appsense
Create index:
PUT citi-bikes
PUT /citi-bikes/$_mapping/bike-alerts''bike-alerts'' : ''properties'' : ''cnt'' : ''type'' : ''integer'', ''location'' : ''type'' : ''geo_point'', ''name''$
Start Elastic Search

3.4 STEP 4 Run Flink

Start flink server
flink run -c com.citibike.batch.BikeRentTrendByAge /pathtojar/flink-java-project-0.1.jar –input <location of tripcsv> –output <output csv file location>
flink run -c com.citibike.batch.BikeRentTrendByGender /pathtojar/flink-java-project-0.1.jar –input <location of tripcsv> –output <output csv file location>
flink run -c com.citibike.stream.LiveStationStatusStream /pathtojar/flink-java-project-0.1.jar –input https://feeds.citibikenyc.com/stations/stations.json

4. CONCLUSION

Apache Flink offered ease of use and better performance to perform analysis in batch environment. SQL processing in batch is seem less.

Stream Execution environment hs rich set of features and connector. Even though Table API is same for batch and stream, I had difficulty getting Kafka stream to dump to a table.