Project Report

Daniele Montesi, Francesco Staccone

October 27, 2019

1 Problem description

Dock-less vehicles programs are becoming very popular nowadays. These sharing services allow people to move easily from point to point within a city without the need of owning a mean of transportation. At the beginning, the service was limited to cars, then, it expanded opening to bikes, motorcycles and scooters. In order to manage those services, a company should include a GPS tracker on each vehicle and register every trip data including GPS coordinates and time for both pick-up and return. The company staff usually is in charge of analyzing the data of their users and displace the vehicles in the city capturing the demand and increasing their revenues.

In this project, we aim to build a real-time GPS-data analyzer for dock-less vehicles trips with the goal of identifying the most popular zones where the sharing services are used. In particular, the project focuses on 2 types of data:

- **pick-ups**, i.e. start locations of the trip
- **returns**, i.e. end locations of the trip

By receiving in input streaming data from the application, we will be able to analyze the spots where the vehicles are picked up and possibly clusters of trips on the map.
The goal of the project is to answer these kind of questions:

- Which are the most popular places where a bike is picked up during specific hours of a specific day?
  i.e. at 13 pm of weekdays bikes are picked up at universities
- Are there any specific areas that identify where bikes are often returned?
  i.e. in the evening bikes are picked up in the city center and returned in the suburbs

2 Tools

Data pre-processing: Jupyter Notebook
Data Storage and Services: Kafka, Elasticsearch.

Processing: Apache Spark Streaming.
Development: Python, Java, Scala.
Visualization: Kibana.

3 Data

The input stream has been simulated reading from file. The data come from a csv dataset containing dock-less (scooter and bike) trips in the city of Louisville, US.
Every row contains data of:

- the trip_id;
- a timestamp;
- the Geo-point coordinates as latitude and longitude;
- a column for the type of data ("pick_up" or "return");
- the day of the week (1 for Sunday, 7 for Saturday);
- the hour of the day;
- the duration of the trip.

The dataset is available **here**.

4 Methodology and algorithm

The streaming pipeline starts from **Kafka** messaging services. A **Producer** is in charge of periodically writing data read from a file onto the Kafka logs, in the topic *"docklessvehicles"*. Then, a Spark Streaming **Consumer** is in charge of reading the data, apply filters, string manipulations and finally load it in Elasticsearch to the index *"/docklessvehicles"*. For this purpose, we have defined a **tumbling window** with 10 seconds range and 10 seconds slide.

The data have been displayed in a map using the **Kibana** visualization tool. Kibana reads from the */docklessvehicles* index predefined in Elasticsearch and it continuously updates the map whenever there is new incoming data.

Through Kibana we are able to specify interesting filters for answering the goals of the project. Some of the useful filters we have used are:

- specify to show only pickups/returns;
- specify the hour intervals where to focus the research;
- specify the day of the week - ranges.

The results are deduced from an empirical point of view thanks to Kibana's visualization tools.

5 How to run the code

Requirements:

- Java 8 SDK
- Python 3
- Scala
- Apache Kafka
- Apache Spark
- Elasticsearch 7.4.0
- Kibana 7.4.0

1. Set the following environment variable.

```
export KAFKA_HOME="/path/to/the/kafka/folder"
```

2. Kafka uses ZooKeeper to maintain the configuration information, so you need to first start a ZooKeeper server if you do not already have one.

```
zookeeper-server-start.sh $KAFKA_HOME/config/zookeeper.
    ↪ properties
```

3. Start the Kafka server.

```
kafka-server-start.sh $KAFKA_HOME/config/server.
    ↪ properties
```

4. Access your ElasticSearch folder and run ElasticSearch.

```
./bin/elasticsearch
```

5. Access your Kibana folder and run Kibana.

```
./bin/kibana
```

6. Set the indexes on Kibana as editable:

```
curl -XPUT -H "Content-Type:␣application/json" http://
    ↪ localhost:9200/_cluster/settings -d '{"transient
    ↪ ":{"cluster.routing.allocation.disk.
    ↪ threshold_enabled":false}}'
curl -XPUT -H "Content-Type:application/json" http://
    ↪ localhost:9200/_all/_settings -d '{"index.blocks.
    ↪ read_only_allow_delete":null}'
```

7. Declare the index on Kibana specifying the types of the variables. Here we specify 2 types: Geo-point, that tells Kibana that latitude and longitude will appear as attributes, and Date:

```
PUT docklessvehicles?include_type_name=true
{
  "mappings": {
    "_doc": {
      "properties": {
        "location": {
          "type": "geo_point"
        },
        "date": {
          "type": "date"
        }
      }
    }
  }
}
```

8. Access the producer folder and create a topic running the producer.

```
cd producer
tail -F file.txt | kafka-console-producer.sh --broker-
    ↪ list localhost:9092 --topic docklessvehicles
```

9. Compile the java class and run the Java process. It is in charge of filling a file text *file.txt* from which the producer will read in order to produce.

```
javac ReadDataFile.java
java ReadDataFile
```

10. Access the consumer folder and compile and run the consumer.

```
cd consumer
sbt run
```

11. Create the index pattern "docklessvehicles" and the related Map visualization on Kibana, then use filters like "date" and "type" to visualize information on the map, or use custom filters like:

```
{
  "query": {
    "range": {
      "duration": {
        "gte": 10
```

```
          }
        }
      }
    }
```

6 Results - Screenshots

The data are loaded into Elasticsearch periodically. Here is the map shown at the beginning when only few samples are present:

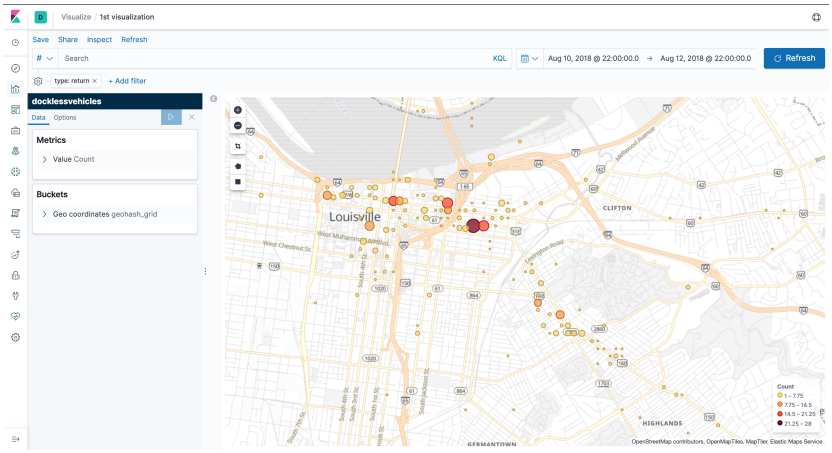

Fig.1: Data related to the first weekend in the dataset

We have investigated the occurrences of pickups and returns during specific hours and days of the week.

6.1 Weekend day v.s. Weekday - clusters

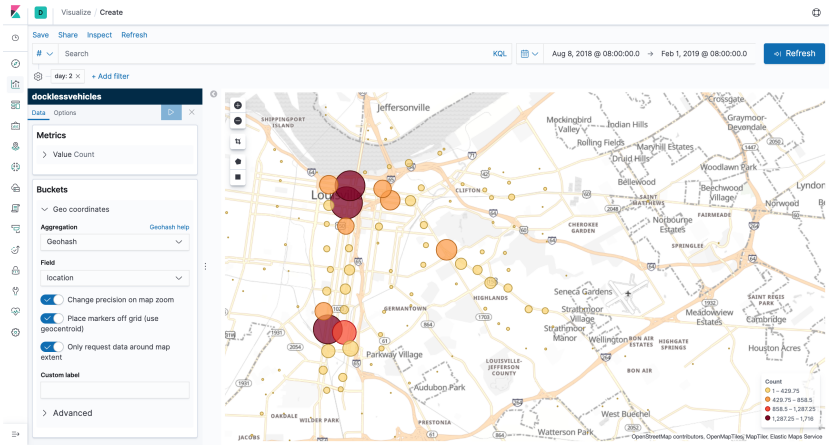

Fig.2: Clusters measured on Monday

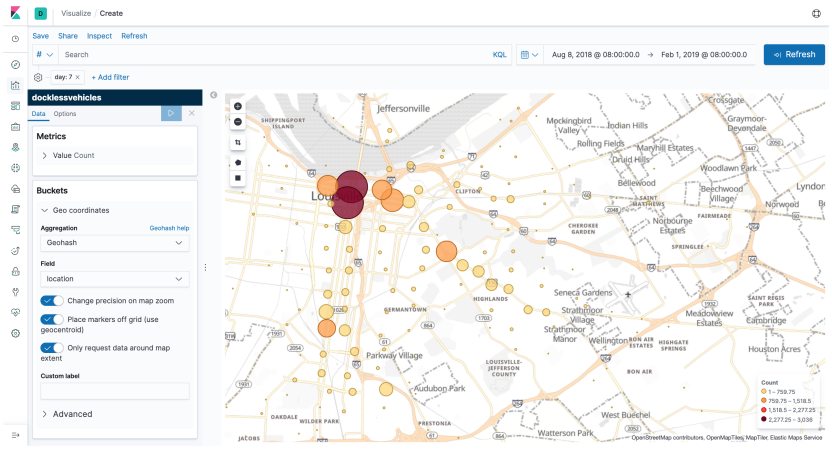

Fig.3: Clusters measured on Saturday

In the figure, it is possible to notice that during weekends (Saturday here), the clusters tend to be more concentrated on the city center. Instead, on weekdays (Monday here) the clusters are also spread over certain zones. For instance, in the cluster on the bottom-left of the picture there is a school.

6.2 Weekend day v.s. Weekday - number of returns

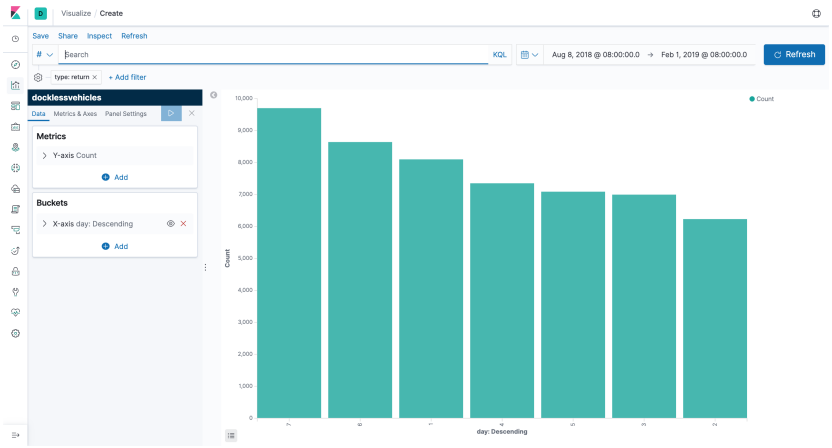

Fig.4: Number of returns measured every week-day

In the figure, it is possible to notice that during weekends, the number of rents visibly increases. In particular, the most busy-day is Saturday, the least is on Monday.
Note that here 1 stands for Sunday, 2 for Monday, and so on until 7 for Saturday.

6.3 Clusters in the Morning v.s. Evening

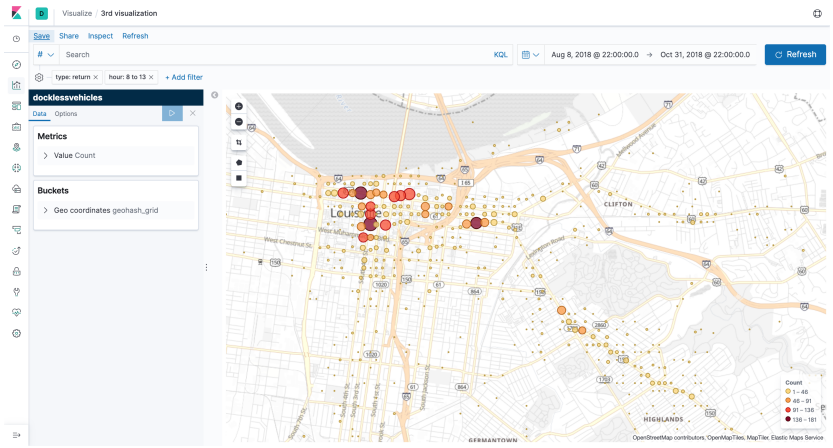

Fig.5: Returns in the morning (8.00 - 13.00)

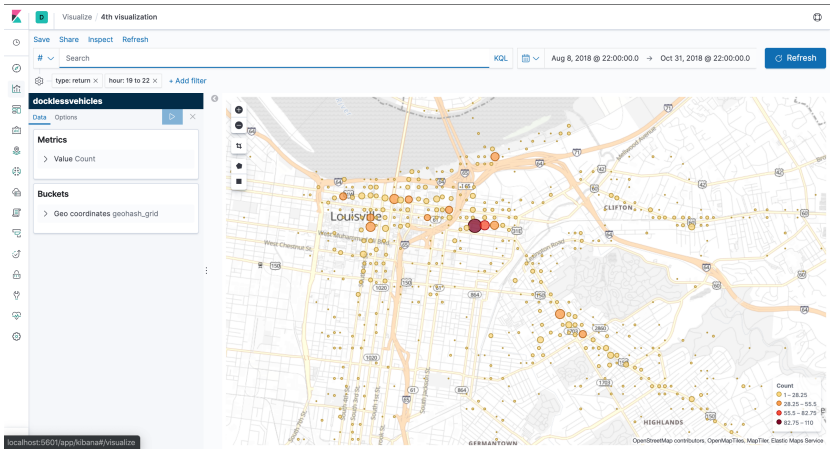

Fig.6: Returns in the evening (19.00 - 22.00)

In the figure, it is possible to notice that during the morning, the number of returns are mainly concentrated in the city centre. Instead, during the evening the returns are spread over the city, meaning that people are probably using the bikes to get back home. As consequence, the clusters are less defined and spread over the map.

6.4 Most common duration

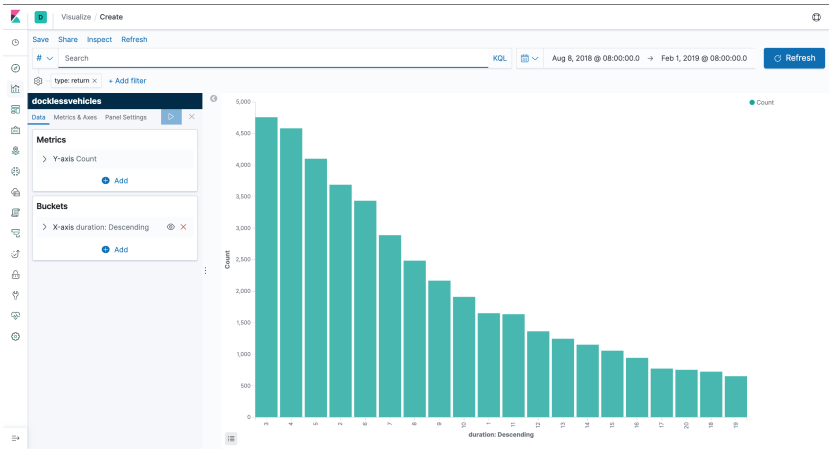

Fig.7: The most common duration if of a few minutes (3, 4, 5) for the specified period

6.5 Most common hours of use

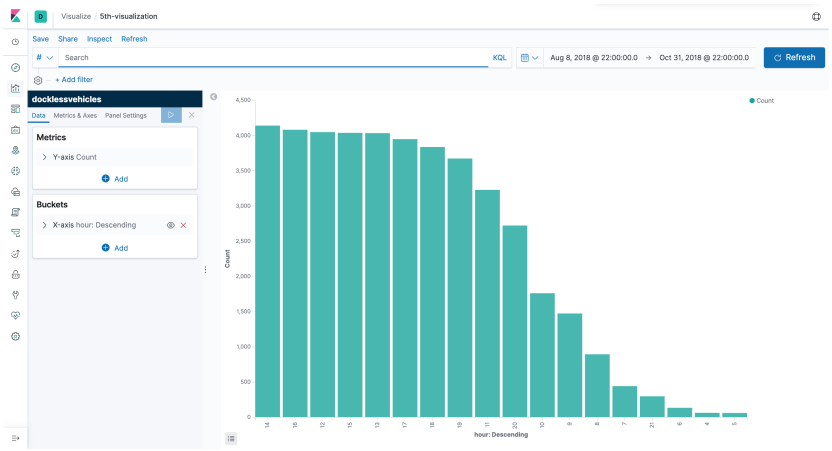

Fig.8: The most common hour of use is 14 for the specified period