# Parallel Programming in Pyspark

David Merchán Cano, Elianne Mora, Camila San José Elizundia

March 15, 2020

The dataset contains records for the yellow and green taxi services of the TLC Taxi Zones. Some of these records include pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts.

To analyze the data, we chose four different studies of interest:

1. The average fare amount (the time-and-distance fare calculated by the meter) for each of the rate codes: standard rate, JFK, Newark, Nassau or Westchester, negotiated fare and group ride.

2. The average trip duration given the reported rate code.

3. The average speed by recorded rate code.

4. The average total amount of payment by the trip distance given specified mileage intervals: i) 0 to 0.98 miles, ii) 0.98 to 1.61, iii) 1.61 to 3 miles and iv) more than 3 miles.

5. The trip distance and average tip amount with the same intervals as previously mentioned.

The study is first carried out in Jupiter notebook, with the Python language and is later translated into Pyspark to be processed in parallel.

1. Fare Amount for each Rate Code

    - The following graphs were made with the Python (Figure 1) and the Pyspark (Figure 2) programs. We can see that the highest fare amount belongs to the people that go to Newark with an average of about 64.31 dollars, while the lowest fare amount belongs to the group ride, followed by the standard rate.

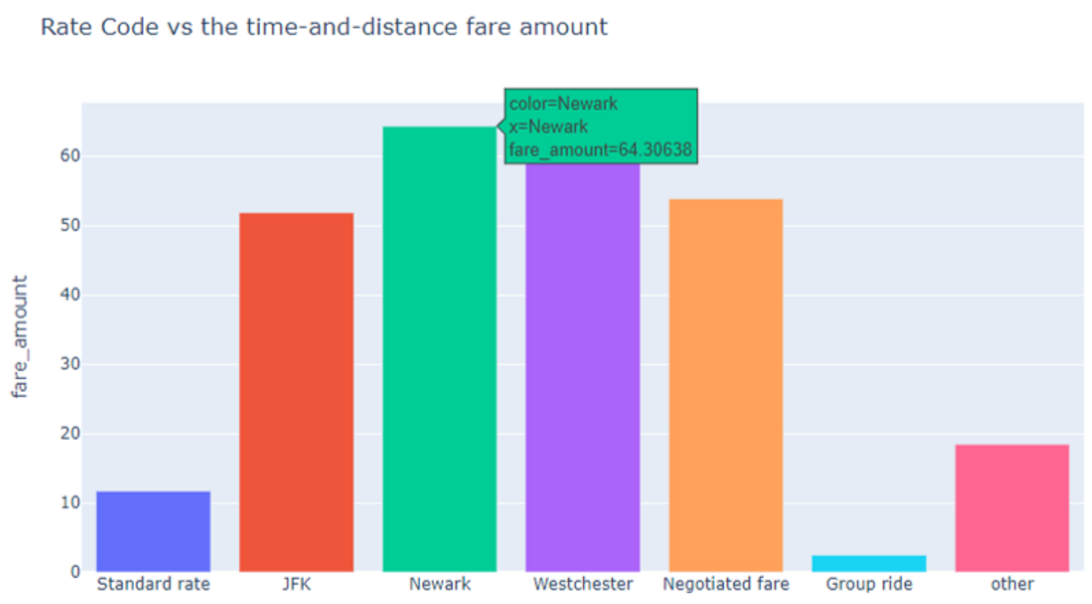
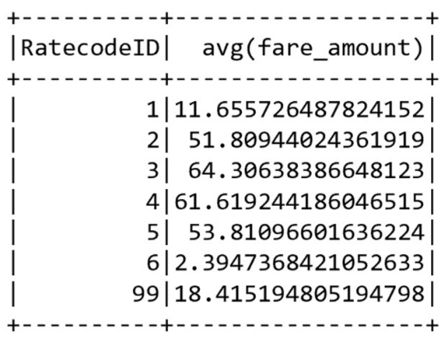


Figure 1: Bar Chart of the Rate Code vs the Fare amount



```
+----------+------------------+
|RatecodeID|  avg(fare_amount)|
+----------+------------------+
|         1|11.655726487824152|
|         2| 51.80944024361919|
|         3| 64.30638386648123|
|         4|61.619244186046515|
|         5| 53.81096601636224|
|         6|2.3947368421052633|
|        99|18.415194805194798|
+----------+------------------+
```

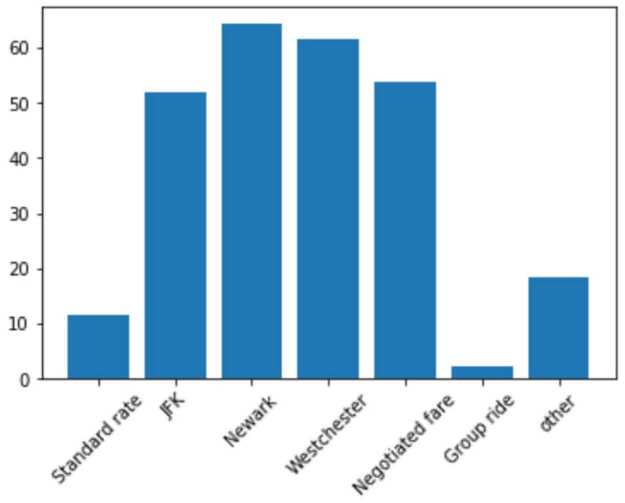


Figure 2: Pyspark Bar Chart of the Rate Code vs the Fare amount

2. Average trip duration by Rate Code

    - We can observe that in Figure 3 and 4 that the highest average trip duration corresponds to trips recorded under the JFK airport destination, while the lowest average trip duration belongs to the group ride rate.

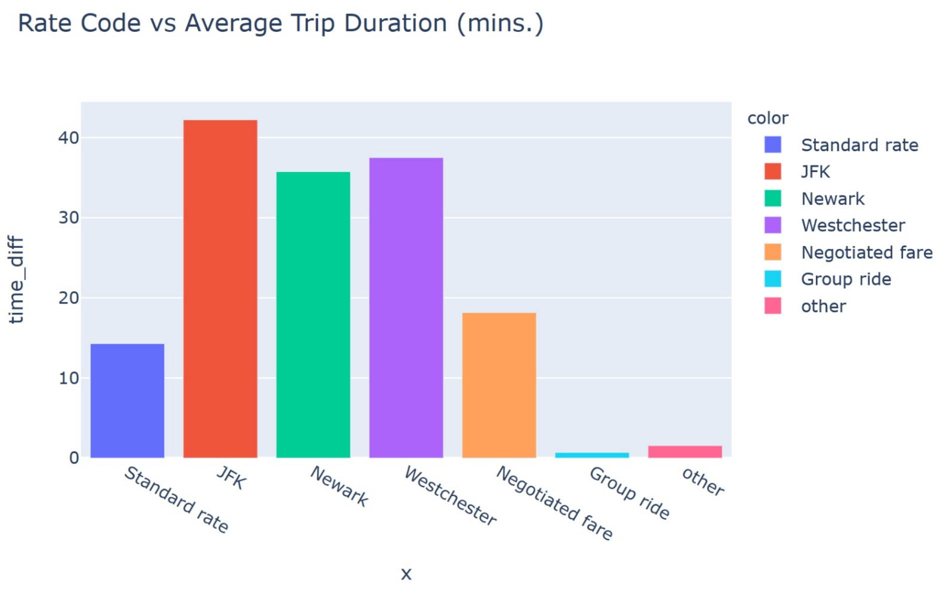
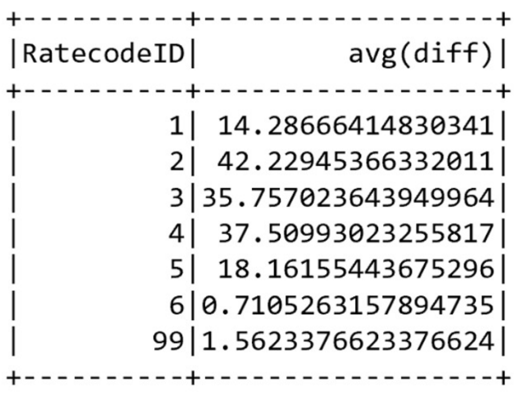


Figure 3: Bar Chart of Rate Code vs the Average Trip duration



```
+----------+------------------+
|RatecodeID|         avg(diff)|
+----------+------------------+
|         1| 14.28666414830341|
|         2| 42.22945366332011|
|         3|35.757023643949964|
|         4| 37.50993023255817|
|         5| 18.16155443675296|
|         6|0.7105263157894735|
|        99|1.5623376623376624|
+----------+------------------+
```

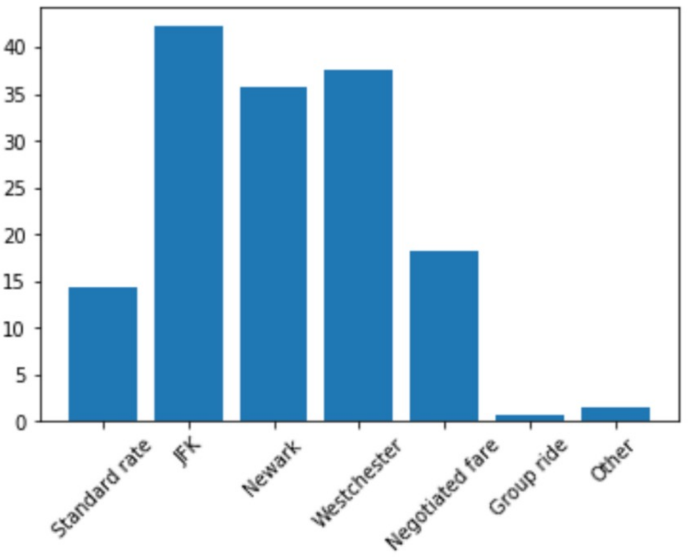


Figure 4: Pyspark Bar Chart of Rate Code vs the Average Trip duration

3. Average speed by Rate Code

    - Our computations of the average speed, in miles per hour, given the reported pick-up and drop-off date/times, along with the trip distance, lead to an abnormal average speed of approximately 520 mph. The high computed speeds solely belong to the group ride rate, as it is the only group that reports long distances (20+ miles) and trip duration of less than a minute. These records may be due to drivers inputting the fare codes and amounts once the group has been delivered to their final destinations (which would take a few seconds), given a pre-established rate according to the number of passengers and destination.

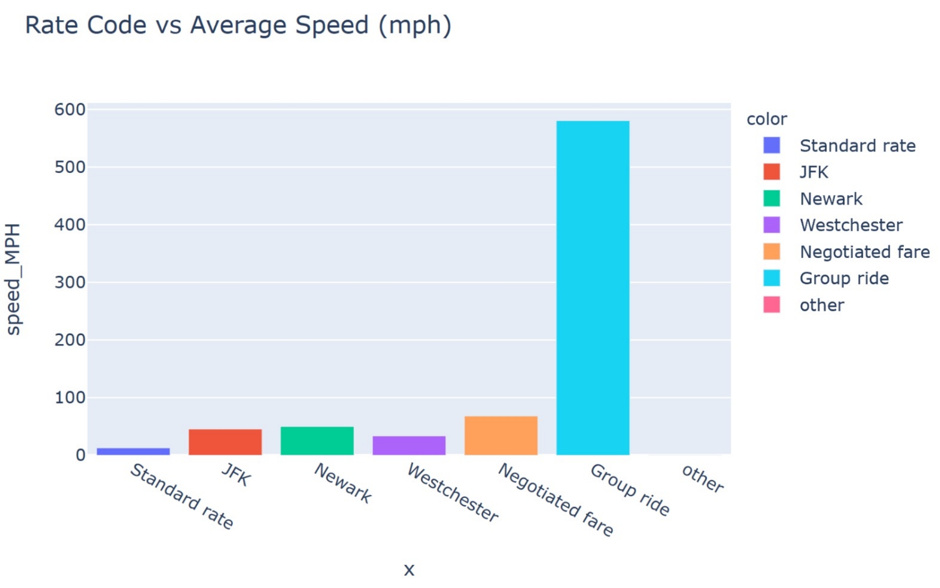
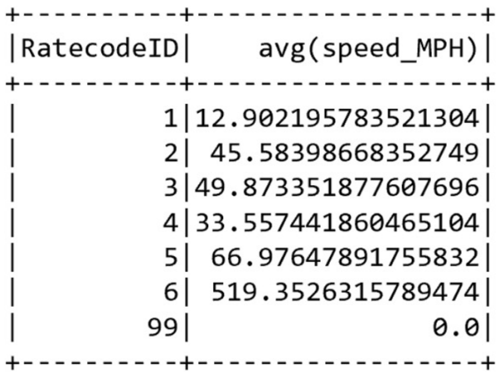


Figure 5: Bar Chart of Rate Code vs the Average Speed



```
+----------+------------------+
|RatecodeID|    avg(speed_MPH)|
+----------+------------------+
|         1|12.902195783521304|
|         2| 45.58398668352749|
|         3|49.873351877607696|
|         4|33.557441860465104|
|         5| 66.97647891755832|
|         6| 519.3526315789474|
|        99|               0.0|
+----------+------------------+
```

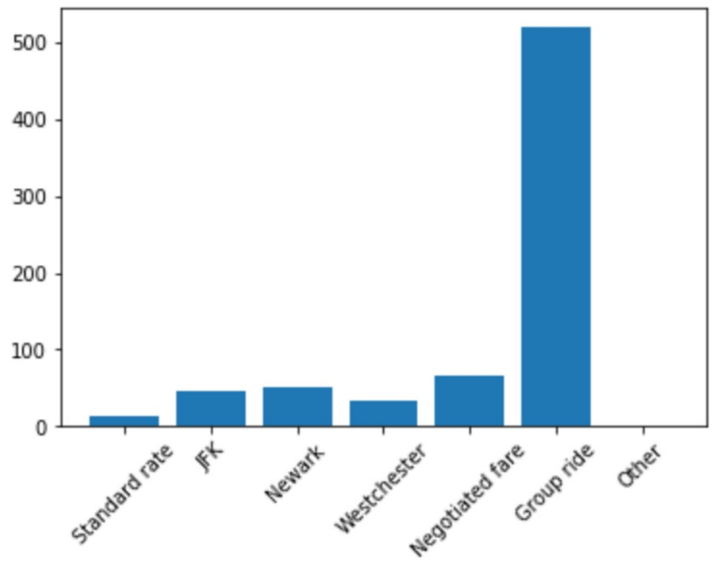
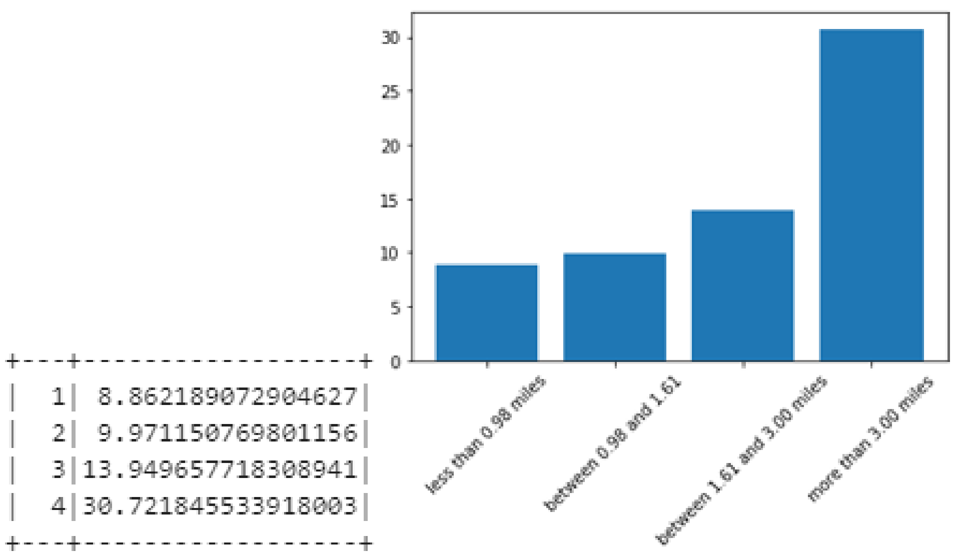


Figure 6: Pyspark Bar Chart of Rate Code vs the Average Speed

4. Average total amount of payment by trip distance

   - In Figure 7 we can see the relation between the total amount by trip distance. The trip distance was grouped in four using the quantiles to classify them. We can conclude that when the distance grows the average total amount that a client pays is higher.



```
+---+------------------+
|  1| 8.862189072904627|
|  2| 9.971150769801156|
|  3|13.949657718308941|
|  4|30.721845533918003|
+---+------------------+
```

Figure 7: Pyspark Bar Chart of the total amount of payment vs the trip distance

5. Average tip amount by trip distance

   - In Figure 9 we can see that the highest amount of tips belong the people that travel greater distance, while the lowest tips come from the lowest distances, which was expected. However, we can see that people that have a distance between 0.98 and 1.61 miles give a similar tip that those with a distance of less than or equal to 0.98 miles.

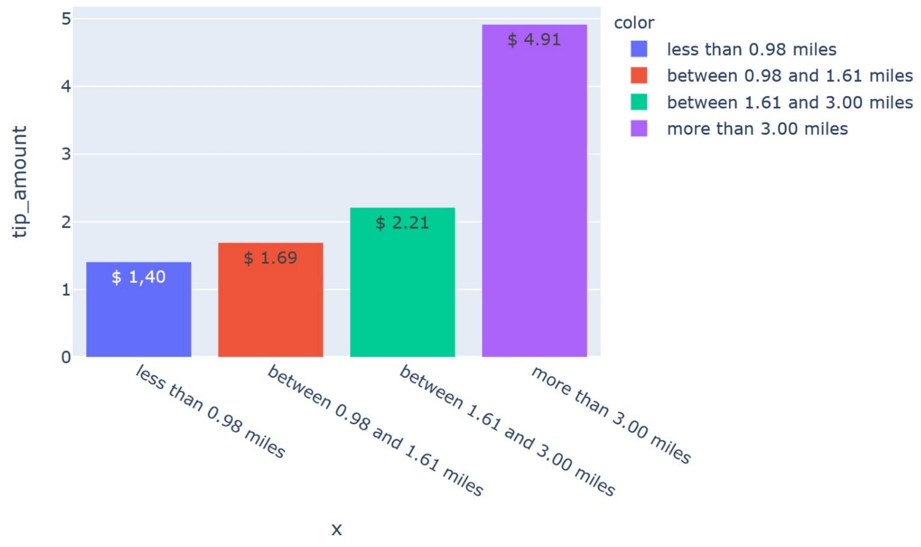


Figure 8: Bar Chart of the Average Speed vs the Tip Amount

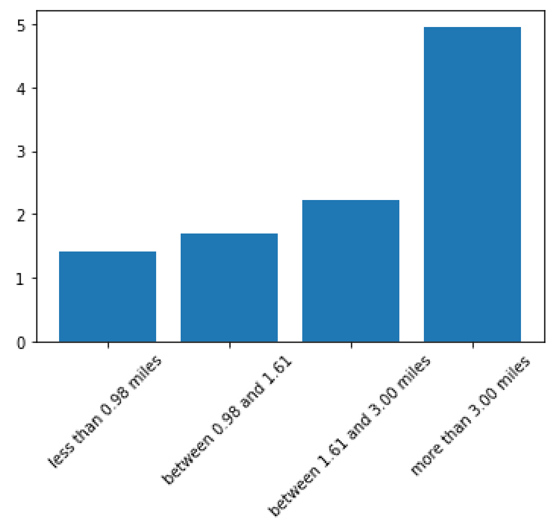


Figure 9: Pyspark Bar Chart of the Average Speed vs the Tip Amount

For our PySpark parallel version, we connect to a Spark cluster and create RDDs, accumulators and broadcast variables on that cluster. Here we set it up to use local nodes - the argument locals[*] means to use the local machine as the cluster, using as many worker threads as there are cores.

The execution time of the Python serial program, inclusive of all data transformations and variable additions and excluding plots, was 0:00:29.428590 seconds, while the parallel program's was 0:00:55.251974 - the latter took longer. However, if we obtain the time it took the serial program to compute our data analysis portion only (computation of averages given specifications) the serial program took 0:00:01.337366 seconds, while the parallel program was able to make these computations in 0:00:00.241810 seconds. This means that while PySpark is slower in the data processing of transformations and variable additions, it is significantly faster than Python in computations of our analysis. This conclusion is supported by the amount of data each program processes. The function sys.getsizeof() returns the size of the object (our dataframe) in bytes. The table below contains all data summarized:

Data Size, Processing Time and Speed by Process

| | **Serial Program (Python)** | **Parallel Program(PySpark** |
|---|---|---|
| Data Manipulation & Analysis (1) | 0:00:29.428590 | 0:00:55.251974 |
| Data Analysis (2) | 0:00:01.337366 | 0:00:00.241810 |
| Plots (3) | 0:00:01.332779 | 0:03:30.099743 |
| Data Processed in Kb | 407,908.304 | 0.056 |
| Processing Speed of section (1) | 13,861.23 KB/s | 0.001013 KB/s |
| Processing Speed of section (2) | 305,008.72 KB/s | 0.231587 KB/s |