Devika Chandnani and Ryan McLarty

Professor Ying Mao, CISC 5950

29th April 2022

Big Data Computing: Lab 2

In Lab2, we will be trying to implement K-means using any Spark related library package such as SparkML or SparkMLLib. The following is a summary of each requested task:

- Part 1 requires us to use the data file named shot_logs.csv to classify each players records into 4 comfortable zones, consider the hit-rate and identify which zone is best for the four given players – James Harden, Chris Paul, Stephen Curry and Lebron James.
- Part 2 requires us to use the NY Parking Violation data and figure out if a Black Vehicle that is parked illegally at the street code 34510, 10030, 34050 (street codes), what is the probability that it will get a parking ticket.
- Part 3 requires us to explore different levels of parallelism, specifically 2,3,4,5 when exploring the question: When are tickets most likely to be issued?

PART 1

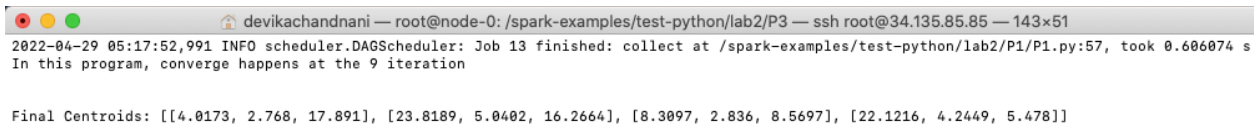
P1.py

Here, we implement Kmeans using an RDD based iteration, like the question specifies, we transform the previous RDD into a new one for the next iteration. We begin by:

- Initializing the Spark session
- Loading the CSV into spark as a dataframe
- We implement the K-Means algorithm and initialize 4 centroids
- The algorithm calculates the distances to the four centroids and assigns each to the closest cluster just like any other K-Means algorithm
- We then calculate a new centroid and stop the iteration when the old centroid is the same as the new one, this is the converging point.

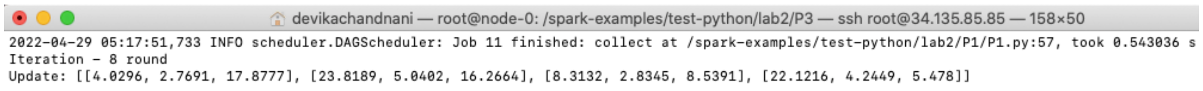
Output:

Running the command " bash test.sh" in the directory named P1 returns the output centroid:



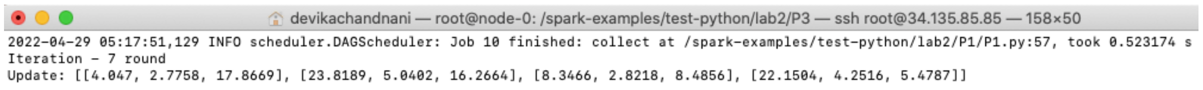
```
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 143×51
2022-04-29 05:17:52,991 INFO scheduler.DAGScheduler: Job 13 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 0.606074 s
In this program, converge happens at the 9 iteration


Final Centroids: [[4.0173, 2.768, 17.891], [23.8189, 5.0402, 16.2664], [8.3097, 2.836, 8.5697], [22.1216, 4.2449, 5.478]]
```

Below are the iterations:



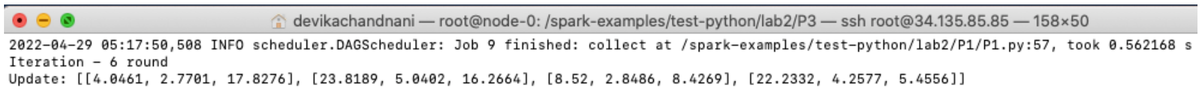
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 158×50

```
2022-04-29 05:17:51,733 INFO scheduler.DAGScheduler: Job 11 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 0.543036 s
Iteration - 8 round
Update: [[4.0296, 2.7691, 17.8777], [23.8189, 5.0402, 16.2664], [8.3132, 2.8345, 8.5391], [22.1216, 4.2449, 5.478]]
```



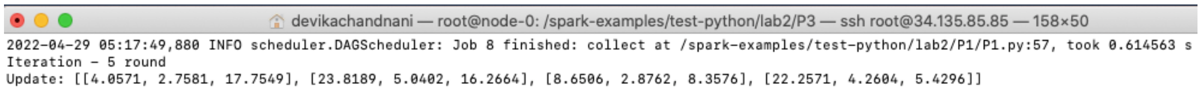
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 158×50

```
2022-04-29 05:17:51,129 INFO scheduler.DAGScheduler: Job 10 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 0.523174 s
Iteration - 7 round
Update: [[4.047, 2.7758, 17.8669], [23.8189, 5.0402, 16.2664], [8.3466, 2.8218, 8.4856], [22.1504, 4.2516, 5.4787]]
```



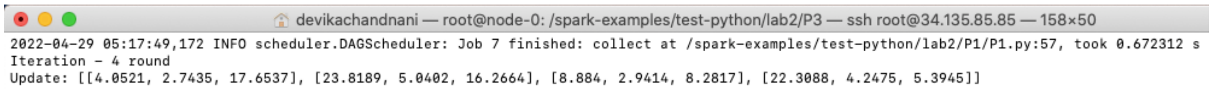
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 158×50

```
2022-04-29 05:17:50,508 INFO scheduler.DAGScheduler: Job 9 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 0.562168 s
Iteration - 6 round
Update: [[4.0461, 2.7701, 17.8276], [23.8189, 5.0402, 16.2664], [8.52, 2.8486, 8.4269], [22.2332, 4.2577, 5.4556]]
```



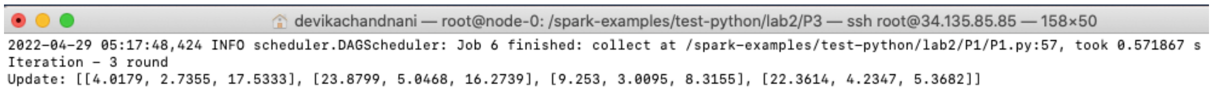
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 158×50

```
2022-04-29 05:17:49,880 INFO scheduler.DAGScheduler: Job 8 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 0.614563 s
Iteration - 5 round
Update: [[4.0571, 2.7581, 17.7549], [23.8189, 5.0402, 16.2664], [8.6506, 2.8762, 8.3576], [22.2571, 4.2604, 5.4296]]
```



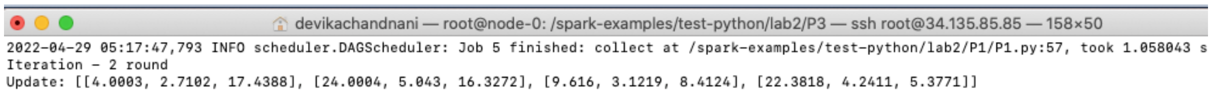
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 158×50

```
2022-04-29 05:17:49,172 INFO scheduler.DAGScheduler: Job 7 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 0.672312 s
Iteration - 4 round
Update: [[4.0521, 2.7435, 17.6537], [23.8189, 5.0402, 16.2664], [8.884, 2.9414, 8.2817], [22.3088, 4.2475, 5.3945]]
```



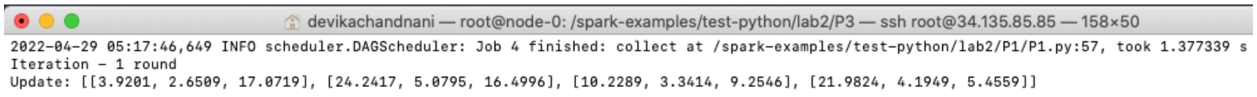
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 158×50

```
2022-04-29 05:17:48,424 INFO scheduler.DAGScheduler: Job 6 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 0.571867 s
Iteration - 3 round
Update: [[4.0179, 2.7355, 17.5333], [23.8799, 5.0468, 16.2739], [9.253, 3.0095, 8.3155], [22.3614, 4.2347, 5.3682]]
```



devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 158×50

```
2022-04-29 05:17:47,793 INFO scheduler.DAGScheduler: Job 5 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 1.058043 s
Iteration - 2 round
Update: [[4.0003, 2.7102, 17.4388], [24.0004, 5.043, 16.3272], [9.616, 3.1219, 8.4124], [22.3818, 4.2411, 5.3771]]
```



devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 158×50

```
2022-04-29 05:17:46,649 INFO scheduler.DAGScheduler: Job 4 finished: collect at /spark-examples/test-python/lab2/P1/P1.py:57, took 1.377339 s
Iteration - 1 round
Update: [[3.9201, 2.6509, 17.0719], [24.2417, 5.0795, 16.4996], [10.2289, 3.3414, 9.2546], [21.9824, 4.1949, 5.4559]]
```

PART 2

P2.py

Here, we utilize K-means again implemented with PySpark. For this question, we follow the theory that the probability of the given black car getting a ticket is equal to the total number of issued tickets in the cluster divided by the average number of illegally parked cars times total number of street codes in that cluster.

The formula can be represented more clearly like this:

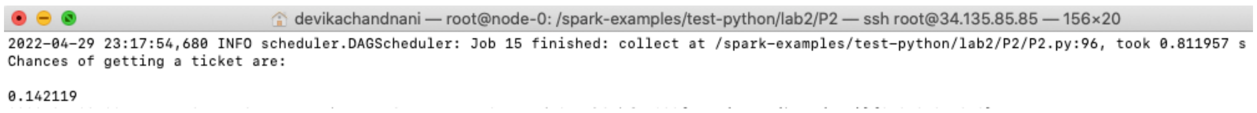
% of getting a ticket = [(total number of tickets in cluster/((average illegally parked cars)*(total number of street codes in the cluster))]

We begin by:

- Initializing the Spark Session
- We then load the dataset in as a Data Frame, we filter it by black vehicles only.
- For all black vehicles, we single out the street code, and then transform this filtered dataset into a data frame that we can continue to work with
- We implement the K-means algorithm as before, so we start by calculating each data point's distance to the initialized centroids
- We assign them to closest centroids, calculate a new centroid using mean and recalculating the distances with the new centroid.
- Next we calculate the probability using the formula explained above

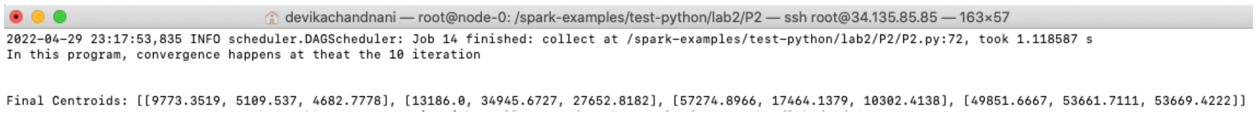
Output:

Running the command " bash test.sh" command in P2 directory returns the following output probability of 0.14



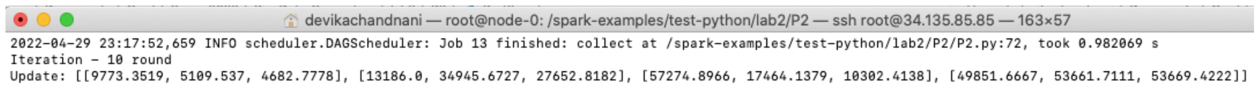
```
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 156×20
2022-04-29 23:17:54,680 INFO scheduler.DAGScheduler: Job 15 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:96, took 0.811957 s
Chances of getting a ticket are:

0.142119
```

The final centroids, with convergence at the 10th iteration, were the following:

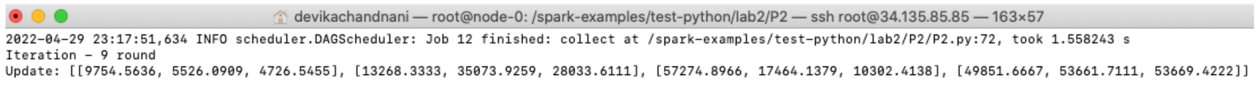


```
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57
2022-04-29 23:17:53,835 INFO scheduler.DAGScheduler: Job 14 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.118587 s
In this program, convergence happens at theat the 10 iteration


Final Centroids: [[9773.3519, 5109.537, 4682.7778], [13186.0, 34945.6727, 27652.8182], [57274.8966, 17464.1379, 10302.4138], [49851.6667, 53661.7111, 53669.4222]]
```

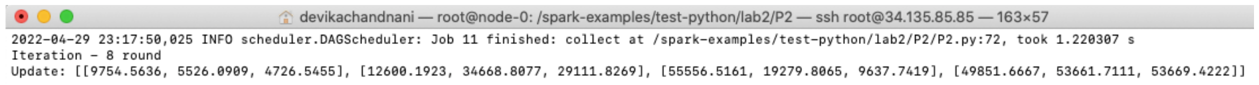
Below are the results from each iteration:


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

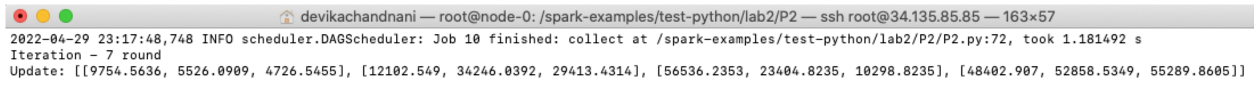
```
2022-04-29 23:17:52,659 INFO scheduler.DAGScheduler: Job 13 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 0.982069 s
Iteration - 10 round
Update: [[9773.3519, 5109.537, 4682.7778], [13186.0, 34945.6727, 27652.8182], [57274.8966, 17464.1379, 10302.4138], [49851.6667, 53661.7111, 53669.4222]]
```


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

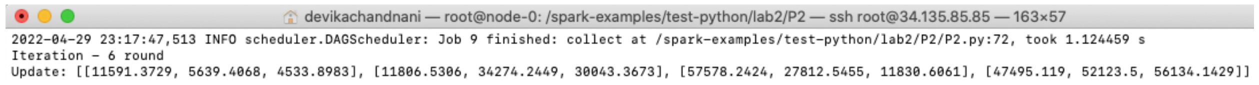
```
2022-04-29 23:17:51,634 INFO scheduler.DAGScheduler: Job 12 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.558243 s
Iteration - 9 round
Update: [[9754.5636, 5526.0909, 4726.5455], [13268.3333, 35073.9259, 28033.6111], [57274.8966, 17464.1379, 10302.4138], [49851.6667, 53661.7111, 53669.4222]]
```


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

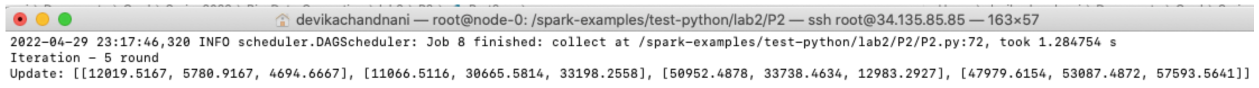
```
2022-04-29 23:17:50,025 INFO scheduler.DAGScheduler: Job 11 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.220307 s
Iteration - 8 round
Update: [[9754.5636, 5526.0909, 4726.5455], [12600.1923, 34668.8077, 29111.8269], [55556.5161, 19279.8065, 9637.7419], [49851.6667, 53661.7111, 53669.4222]]
```


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

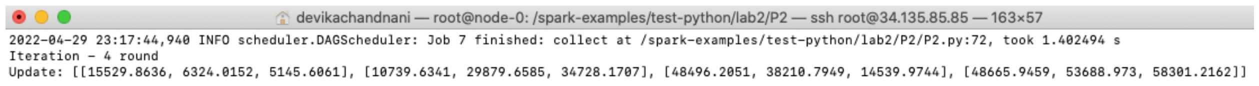
```
2022-04-29 23:17:48,748 INFO scheduler.DAGScheduler: Job 10 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.181492 s
Iteration - 7 round
Update: [[9754.5636, 5526.0909, 4726.5455], [12102.549, 34246.0392, 29413.4314], [56536.2353, 23404.8235, 10298.8235], [48402.907, 52858.5349, 55289.8605]]
```


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

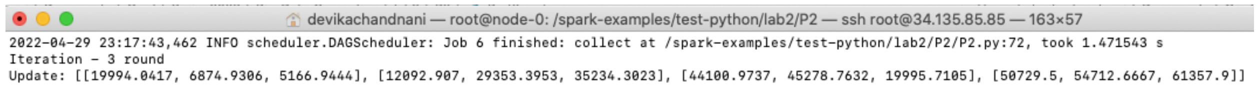
```
2022-04-29 23:17:47,513 INFO scheduler.DAGScheduler: Job 9 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.124459 s
Iteration - 6 round
Update: [[11591.3729, 5639.4068, 4533.8983], [11806.5306, 34274.2449, 30043.3673], [57578.2424, 27812.5455, 11830.6061], [47495.119, 52123.5, 56134.1429]]
```


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

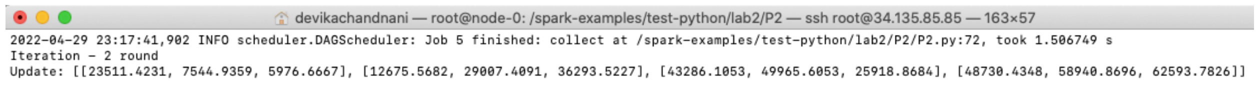
```
2022-04-29 23:17:46,320 INFO scheduler.DAGScheduler: Job 8 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.284754 s
Iteration - 5 round
Update: [[12019.5167, 5780.9167, 4694.6667], [11066.5116, 30665.5814, 33198.2558], [50952.4878, 33738.4634, 12983.2927], [47979.6154, 53087.4872, 57593.5641]]
```


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

```
2022-04-29 23:17:44,940 INFO scheduler.DAGScheduler: Job 7 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.402494 s
Iteration - 4 round
Update: [[15529.8636, 6324.0152, 5145.6061], [10739.6341, 29879.6585, 34728.1707], [48496.2051, 38210.7949, 14539.9744], [48665.9459, 53688.973, 58301.2162]]
```


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

```
2022-04-29 23:17:43,462 INFO scheduler.DAGScheduler: Job 6 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.471543 s
Iteration - 3 round
Update: [[19994.0417, 6874.9306, 5166.9444], [12092.907, 29353.3953, 35234.3023], [44100.9737, 45278.7632, 19995.7105], [50729.5, 54712.6667, 61357.9]]
```


devikachandnani — root@node-0: /spark-examples/test-python/lab2/P2 — ssh root@34.135.85.85 — 163×57

```
2022-04-29 23:17:41,902 INFO scheduler.DAGScheduler: Job 5 finished: collect at /spark-examples/test-python/lab2/P2/P2.py:72, took 1.506749 s
Iteration - 2 round
Update: [[23511.4231, 7544.9359, 5976.6667], [12675.5682, 29007.4091, 36293.5227], [43286.1053, 49965.6053, 25918.8684], [48730.4348, 58940.8696, 62593.7826]]
```

PART 3

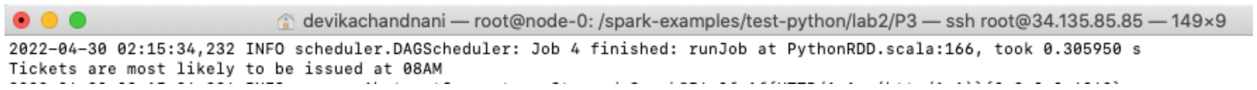
P3.py

Here, we implement a spark program to single out the most frequent time at which tickets are issues, but we check how long the program takes to run with different levels of parelellism. We begin by:

- Initializing the Spark session
- For this question we will simply single out the most frequent time of day in 00 AM/PM hours, the output results in 8AM, which is the same answer we received using Hadoop Map/Reduce Framework for this question in our Project.
- We configure parallelism in the test.sh file using the command:
  "—conf spark.default.parallelism=2"
  "—conf spark.default.parallelism=3"
  "—conf spark.default.parallelism=4"
  "—conf spark.default.parallelism=5"
- We update the test.sh file 4 times, once with each level of parallelism to test the speed

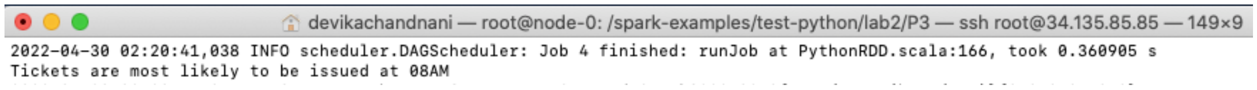
Output:

At parallelism = 2, the output was:



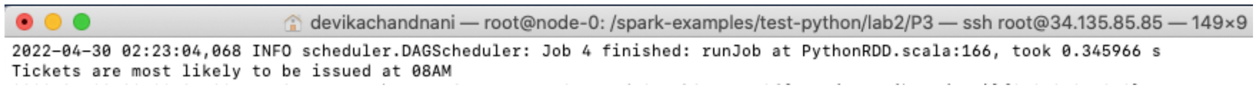
```
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 149×9
2022-04-30 02:15:34,232 INFO scheduler.DAGScheduler: Job 4 finished: runJob at PythonRDD.scala:166, took 0.305950 s
Tickets are most likely to be issued at 08AM
```

At parallelism = 3, the output was:



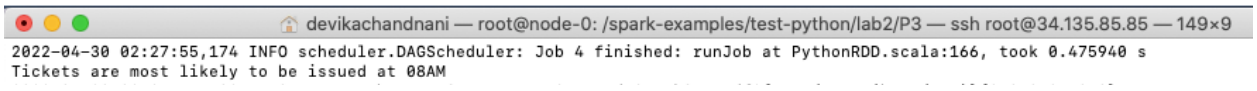
```
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 149×9
2022-04-30 02:20:41,038 INFO scheduler.DAGScheduler: Job 4 finished: runJob at PythonRDD.scala:166, took 0.360905 s
Tickets are most likely to be issued at 08AM
```

At parallelism = 4, the output was:



```
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 149×9
2022-04-30 02:23:04,068 INFO scheduler.DAGScheduler: Job 4 finished: runJob at PythonRDD.scala:166, took 0.345966 s
Tickets are most likely to be issued at 08AM
```

At parallelism = 5, the output was:



```
devikachandnani — root@node-0: /spark-examples/test-python/lab2/P3 — ssh root@34.135.85.85 — 149×9
2022-04-30 02:27:55,174 INFO scheduler.DAGScheduler: Job 4 finished: runJob at PythonRDD.scala:166, took 0.475940 s
Tickets are most likely to be issued at 08AM
```