Student name: Chandramohan Natarajan
Student ID: 190617866
Module: ECS765P

Part A: Time Analysis

Job ID: #http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1575381276332_1565/

Plot:

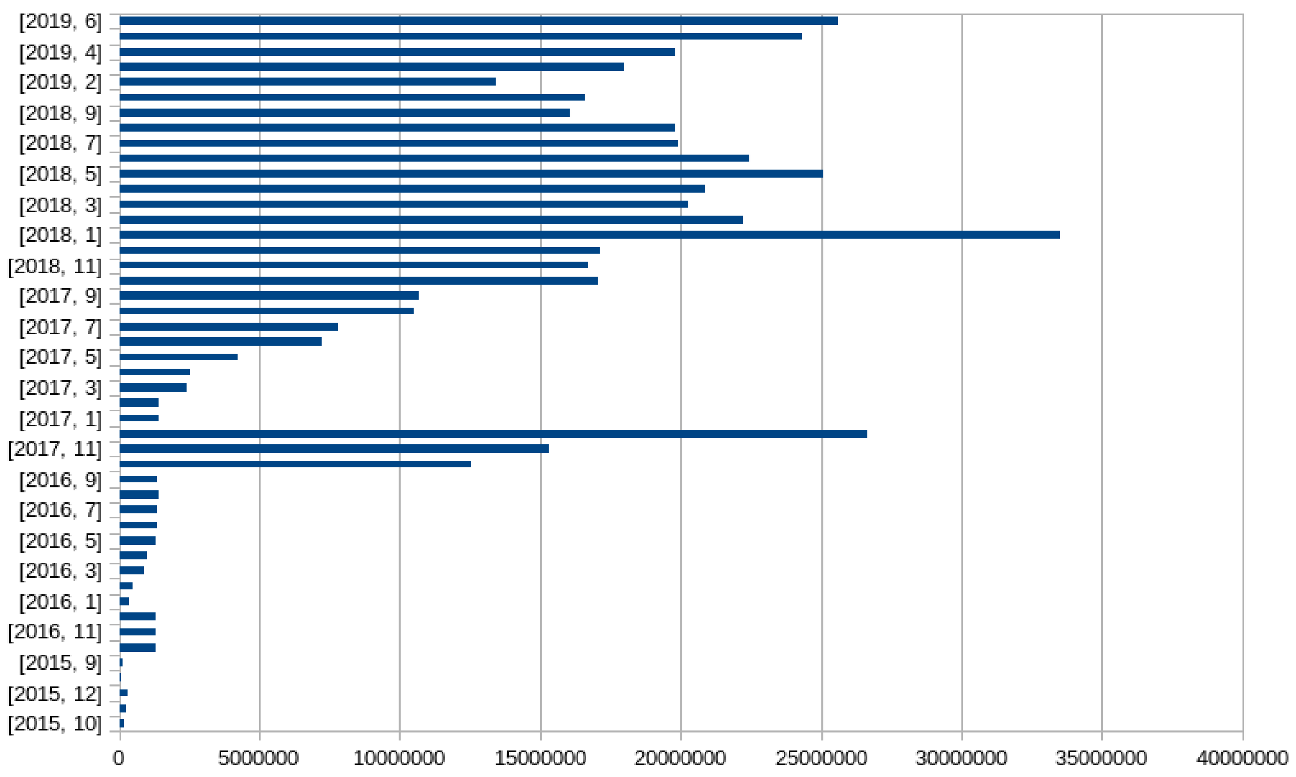


Table used: Transactions

Columns required to execute Part One : block_timestamp

Task : To create a bar plot showing the number of transactions occurring every month between 14-02-2016 till 30-06-2019.

Process flow:

Mapper:

1) Every row is made into a line with space as delimiter

block_number | from_address | to_address | value | gas | gas_price | block_timestamp

2) values in block_timestamp is broken into year with "time" function

3) Key value pair is generated for every year/month

Map input records=486522454

Map output records=486521365

<u>Combiner</u>:

All key value are summed up before giving that output to reducer

Combine input records=486521365

Combine output records=1530

<u>Reducer:</u>

All key value pairs are aggregated and output is stored in user folder of hdfs

Reduce input records=1530

Reduce output reports = 47

Part B: Top Ten Most Popular Services - Initial Aggregation

<u>Job id</u> : http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1575381276332_1621/

<u>Table used</u>:

Transactions

<u>Column required to execute Job 1</u>: to_address and value

Task:

<u>Process flow</u>

<u>Mapper:</u>

1) Every row is made into a line with space as delimiter

block_number | from_address | to_address | value | gas | gas_price | block_timestamp

2) The file is converted to line with delimiter space. The respective data from columns address and value is is fed as input to mapper and key value pair is generated.

Map input records=486522454

Map output records=486521365

<u>Combiner:</u>

Shuffle and sort option further aggregate the count in combiner.

Combine input records=639004185

Combine output records=244400090

Reducer:

Finally all aggregation upon key value pair happens in reducer.

Reduce input records=91917270

Reduce output records=53296857

Part B: Top Ten Most Popular Services – Joining transactions/contracts and filtering

Job ID:

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1575381276332_1964/

Table used: Output of Job1 and address in Contracts file

Process flow

Mapper:

The aggregated file which was output of Job 1 and addresses in contracts are joined together and repartition happens.

Address column and address in contracts file are converted to key value pair and given a input to combiner.

Map input records=486522454

Map output records=486521365

Combiner:

Combiner shuffle and sort the key value pair and give input to reducer.

Combine input records=639004185

Combine output records=244400090

Reducer:

As required reducer gives the output desired ,if the address for a given aggregate from Job 1 was not present within contracts this should be filtered out as it is a user address and not a smart contract.

Reduce input records=91917270

Reducer output records = 53296857

<u>**Part B: Top Ten Most Popular Services - Top Ten**</u>

Job ID:

 http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1575381276332_2005

Table used: Output of Job2

<u>**Mapper**</u>:

The aggregated file is used to find the top 10 . Mapper generates key value pairs of address and fed to reducer

Map input records=21398912

Map output records=2243685

<u>**Reducer:**</u>

Reducer further counts upon key value pair and give top 10 values.

Reduce input records=2243685

Reduce output records=10

<u>**Part C – Data Exploration -Scam Analysis**:</u>

<u>**Job ID:**</u>

http://andromeda.student.eecs.qmul.ac.uk/proxy/application_1575381276332_3311/

http://andromeda.student.eecs.qmul.ac.uk/proxy/application_1575381276332_3477/

<u>**Part 1**</u>

Table used: csv file

Let us convert the json file to csv.

Code the program to read the csv files and split the values with spaces.

<u>**Mapper**</u>:

Key value pairs are generated with address and category and later get the output saved in csv format for further processing.

<u>**Part 2**</u>

<u>**Mapper**</u>

After cleaning the csv file , repartition join is done with transaction and output of part 1..

<u>**Reducer**</u>

Reducer aggregates the count and gives the category and addresses

Part 3

Mapper

The output of Part 2 is sorted by first generating key value pair and fed to reducer

Reducer

Reducer aggregates the required output which is top scam.

Part -C Data Exploration - Gas Guzzlers

Job ID http://andromeda.student.eecs.qmul.ac.uk/proxy/application/_1575381276332_3324/

Process

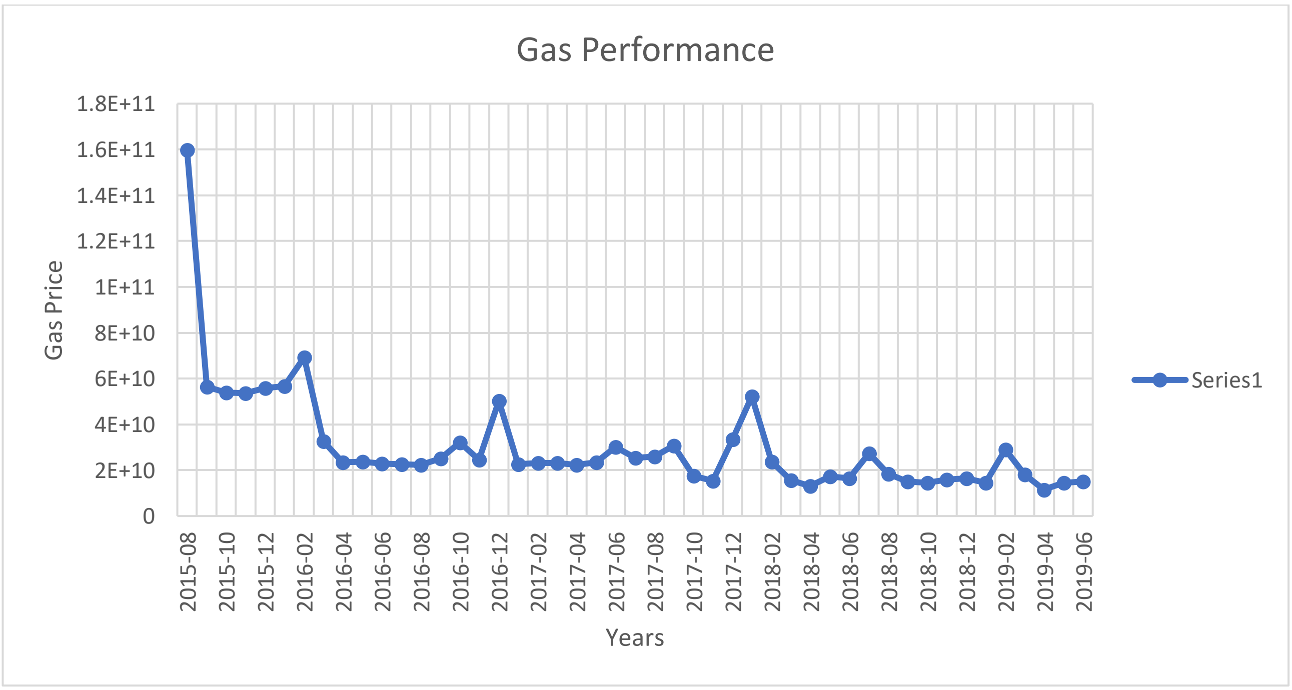
Mapper

The transaction file is fed as input to mapper and the block_times stamp and gas price columns are used.

The key value pairs are generated for gas and days.

Reducer

The key value pairs are summed up to get the desired output of gas transactions for ethereum.



Part C – Data Exploration_- Comparative evaluation

Spark

Speed

Engineered from the bottom-up for performance, Spark can be 100x faster than Hadoop for large scale data processing by exploiting in memory computing and other optimizations. Spark is also fast when data is stored on disk, and currently holds the world record for large-scale on-disk sorting.

Ease of Use

Spark has easy-to-use APIs for operating on large datasets. This includes a collection of over 100 operators for transforming data and familiar data frame APIs for manipulating semi-structured data.

Part B is reimplemented in spark and representative results has been accounted for executing multiple times. Average execution time has been noted.

The time taken for execution of the task is as follows

1) Transactions for every month in a year for given data set – 3 minutes 55 seconds

2)Aggregating the to_address and value of ethereum – 54 minutes

3) Repartition joins to create a table with aggregated output of job 1 and address part of contracts – 8 minutes

4) To find top 10 among filtered address – 1 minute 12 seconds

Totally the amount taken to execute this job is 1 hour 8 minutes.

- ➢ Whereas the time taken by spark to compute and get desired output is given below.

application_1575381276332_2027

- Time : 6.4 minutes

application_1575381276332_2076

- Time : 3.8 minutes

application_1575381276332_2114

- Time : 3.5 minutes

application_1575381276332_2128

- Time : 3.5 minutes

application_1575381276332_2166

- Time: 3.3 minutes

application_1575381276332_2320

- Time : 6.1 minutes

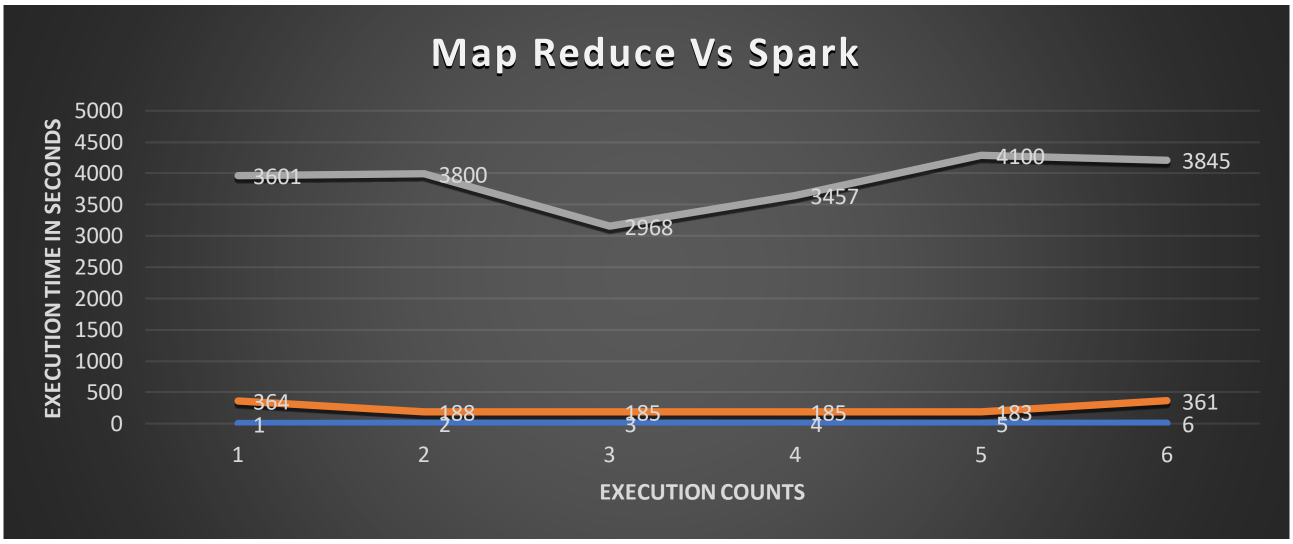
On an average, just 4 minutes is being taken to compute and output the desired results.

As learnt, it is quite evident that Spark is multiple times faster than Map reduce.

<u>Map Reduce</u>

Total time taken by Map Reduce to perform the task can be seen below.

Part B-J1 + Part B_J2 + Part B_J3 = 54.05 + 8.55 + 1.12 = 64 minutes 12 seconds



Map Reduce is less fast because of I/O disk latency while spark is 10x time faster in disk which is quite evident from chart. Map reduce depends on external scheduler but spark got its own scheduler.