Queen Mary
University of London

ANALYSIS OF ETHEREUM TRANSACTIONS AND SMART CONTRACTS
[ECS 765]

Done By
Athithya Balasubramani [EC211224]

Queen Mary University of London

Part A

1)Create a bar plot showing the number of transactions occurring every month between the start and end of the dataset.

Job id:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_4982/

Input file => hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions

Procedure

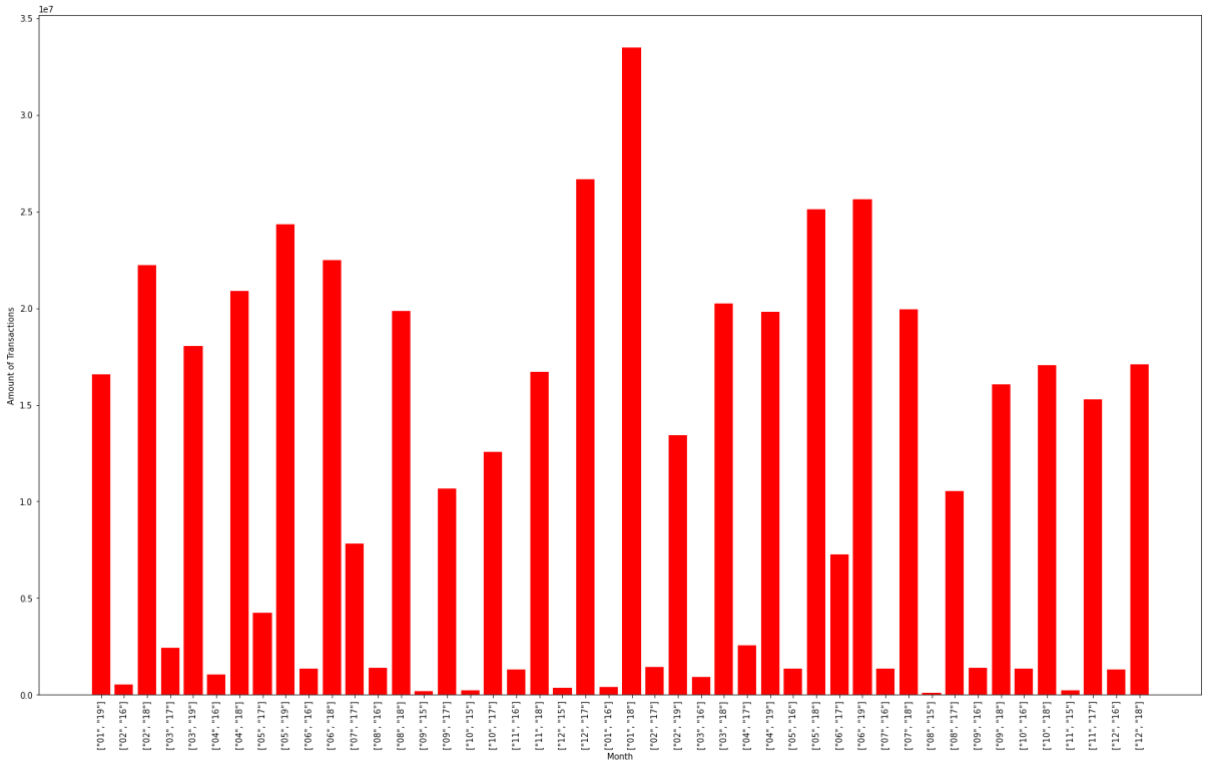
- For this question I used the map reduce technique where I started with importing the necessary library such as mrjob , time to convert the data in epoch time format into regular time format with month and years of the transactions.
- I performed the map reduce program where the epoch time converted to month and year is given to the mapper and counting to be performed in the reducer.
- As a result of the map reduce program I got the output text file "total_transactions" which will be present in the Part_A folder along with map reduce source code "transactions.py"
- I decided to make the bar graph with the python and I created Google Colab notebook where I started by importing the necessary libraries such as matplotlib, pandas, numpy and made a bar graph with "total_transactions.txt" as input with months in. the X-axis and amount of transactions in the Y-axis.

Source Codes Attached:

1. transactions.py
2. Total_transactions.ipynb

Output file Attached:

1. total_transactions.txt



2) Create a bar plot showing the average value of transactions in each month between the start and end of the dataset.

Job id:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_4990/

Input file => hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions

Procedure

- For this question I used the map reduce technique where I started with importing the necessary library such as mrjob , statistics,time to convert the data in epoch time format into regular time format with month and years of the transactions.
- I performed the map reduce program where the epoch time converted to month and year along with the gas price is given to the mapper and I wrote a program to calculate the average transactions with respect to month and counting to be performed in the reducer.
- As a result of the map reduce program, I got the output text file "average_transactions" which will be present in the Part_A folder along with map reduce source code "average_transactions.py"
- I decided to make the bar graph with the python and I created Google Colab notebook where I started by importing the necessary libraries such as matplotlib,pandas,numpy and made a bar graph with "total_transactions.txt" as input with months in. the X-axis and amount of transactions in the Y-axis.

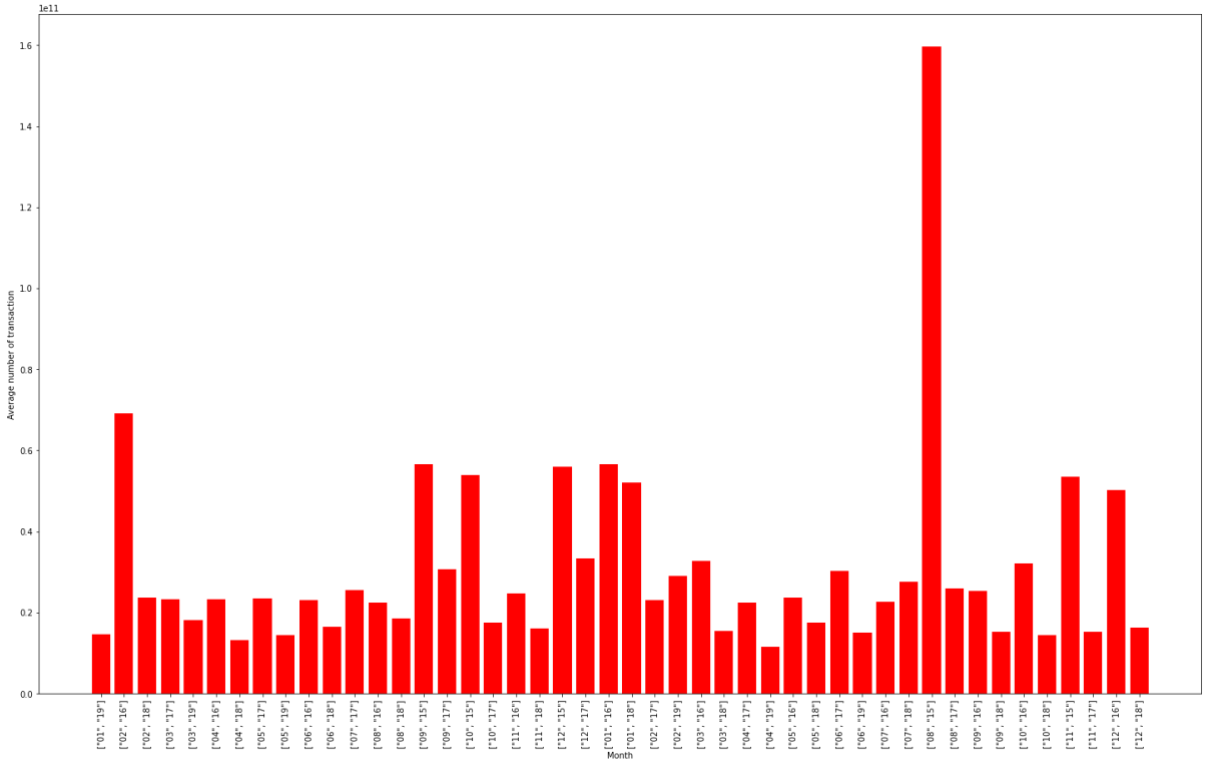
Source Codes Attached:

1.transactions.py

2. Total_transactions.ipynb

Output file Attached:

1. average_transactions.txt



PART B. TOP TEN MOST POPULAR SERVICES

Evaluate the top 10 smart contracts by total Ether received.

Job id:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_5005/

Input file =>

1. hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions
2. hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/contracts

Procedure

- To find the top ten most popular services in the Ethereum datasets I have opted to go with map reduce method
- In the map reduce program I initially start by importing the mrJob library
- In this program I have used two datasets transactions and contracts where fields = 7 refers to the transactions dataset and fields = 5 refers to the contracts dataset
- In the first map reduce program I have selected the address as the key value from both the datasets where the key value pair will be counted as 1 for the transactions dataset and 2 for the contracts datasets. In the

reducer part 1 will be passed as the value to count. A program to check the values passed from the mapper is implemented if the value is true then the record is valid and it sums all the values and since the key is same the values will be transferred from summed value to value.

- In the 2nd map reduce program the mapper takes the values from the 1st reducer and joins the key and values to find the top 10 contracts[popular services] with no specific key . the lambda function will be used in the descending order for values , then a for loop will be used to yield the top 10 values. The output of the code is stored as "top_ten.txt" in the Part_B folder.

Output

```
"0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444"      8415510080996586582272677 6
"0xfa52274dd61e1643d2205169732f29114bc240b3"      4578748448318935298647880 5
"0x7727e5113d1d161373623e5f49fd568b4f543a9e"      4562062400135071255726857 3
"0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef"      4317035609226246891929896 9
"0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8"      2706892158201954249988287 7
"0xbfc39b6f805a9e40e77291aff27aee3c96915bdd"      2110419513809366005000000 0
"0xe94b04a0fed112f3664e45adb2b8915693dd5ff3"      1556239895680211225471940 9
"0xbb9bc244d798123fde783fcc1c72d3bb8c189413"      1198360872920289384681868 1
"0xabbb6bebfa05aa13e908eaa492bd7a8343760477"      1170645717794089552177040 4
"0x341e790174e3a4d35b65fdc067b6b5634a61caea"      8379000751917755624057500
```

Source Codes Attached: top_ten.py

Output file Attached: top_ten.txt

PART C. TOP TEN MOST ACTIVE MINERS

Job id:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_5062/

Input file => hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/blocks
Procedure

- I have opted to proceed with map reduce program to find the top 10 miners by the size of the blocks
- To begin I have imported MRJob and MRStep library and the fields will be 9 since I am going to take the blocks column 9 as value
- I have planned to use two map reduce programs for this problems. In the 1st map reduce program I will use miner as the key value and the size field as our value by converting it into a integer format and then we have to count in the reducer part and the summed value we obtained from the mapper and yield it which should be same.
- In the 2nd mapper I have opted for the miners as the key and the totalsize as the value and in the reducer portion I have made a program with no key value to sort the values and a for loop will be used to yield the top 10 miners

OUTPUT

```
"0xea674fdde714fd979de3edf0f56aa9716b898ec8"	23989401188
"0x829bd824b016326a401d083b33d092293333a830"	15010222714
"0x5a0b54d5dc17e0aadc383d2db43b0a0d3e029c4c"	13978859941
"0x52bc44d5378309ee2abf1539bf71de1b7d7be3b5"	10998145387
"0xb2930b35844a230f00e51431acae96fe543a0347"	7842595276
"0x2a65aca4d5fc5b5c859090a6c34d164135398226"	3628875680
"0x4bb96091ee9d802ed039c4d1a5f6216f90f81b01"	1221833144
"0xf3b9d2c81f2b24b0fa0acaaa865b7d9ced5fc2fb"	1152472379
"0x1e9939daaad6924ad004c2560e90804164900341"	1080301927
"0x61c808d82a3ac53231750dadc13c777b59310bd9"	692942577
```

Source Codes Attached: miners_top_10.py
Output file Attached: top_ten_miners.txt

Part D : SCAM ANALYSIS

Popular Scams:

Categories of scams

Job id:

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_5539/

Input file =>

- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions
- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/scams.json

Procedure

- In this part we are going to find the popular scams and so I have preferred to go with MapReduce program were 2 datasets will be taken
- The program begins by importing the necessary libraries MRJob, MRStep and Json . in the program fields equal to 7 refers to the transactions dataset and add to the json file.
- In the map reduce program from transactions dataset we use the address as the key and values as itself and indicate them with 0 and from the scams dataset we take address got through the loop and indicate them with the value 1.
- In the reducer part the program check the values where they are from(transactions or scams) and send them to the next mapper key value pairs are taken and categorized and passed to the reducer then the total vales are counted as per categories

Output

```
"Scamming"      3.833616286244434e+22
"Fake ICO"      1.3564575668896297e+21
"Phishing"      2.6999375794087423e+22
"Scam"  0
```

Source Codes Attached: 1_scams.py
Output file Attached: 1_scams_in_ethereum.txt

Scam's Status with categories

Job id:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_5539/

Input file =>

- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions
- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/scams.json

Procedure

- In this program we import the same required libraries as before program and check the value of the fields to check whether we are using the transactions dataset
- In the mapper address will be taken as the key value and then indicate them with the value 1
- In the reducer part also we will take address as the key which we got through the loop and we include categories and status of the scam and represent them by value 2 for differentiating purpose.
- The program will check the values and fed them into 2nd mapper and in the 2nd mapper it just passes the received values to the reducer and then the reducer performs the sum operation and gives the output with respect to the status and categories of the scam.

Output

```
["Active", "Scamming"]    88444
["Inactive", "Phishing"]  22
["Offline", "Fake ICO"]   121
["Offline", "Phishing"]   7022
["Offline", "Scam"]       0
["Suspended", "Phishing"] 11
["Active", "Phishing"]    1584
["Offline", "Scamming"]   24692
["Suspended", "Scamming"] 56
```

Source Codes Attached: Scams7.py
Output file Attached: Popular_Scams1.txt

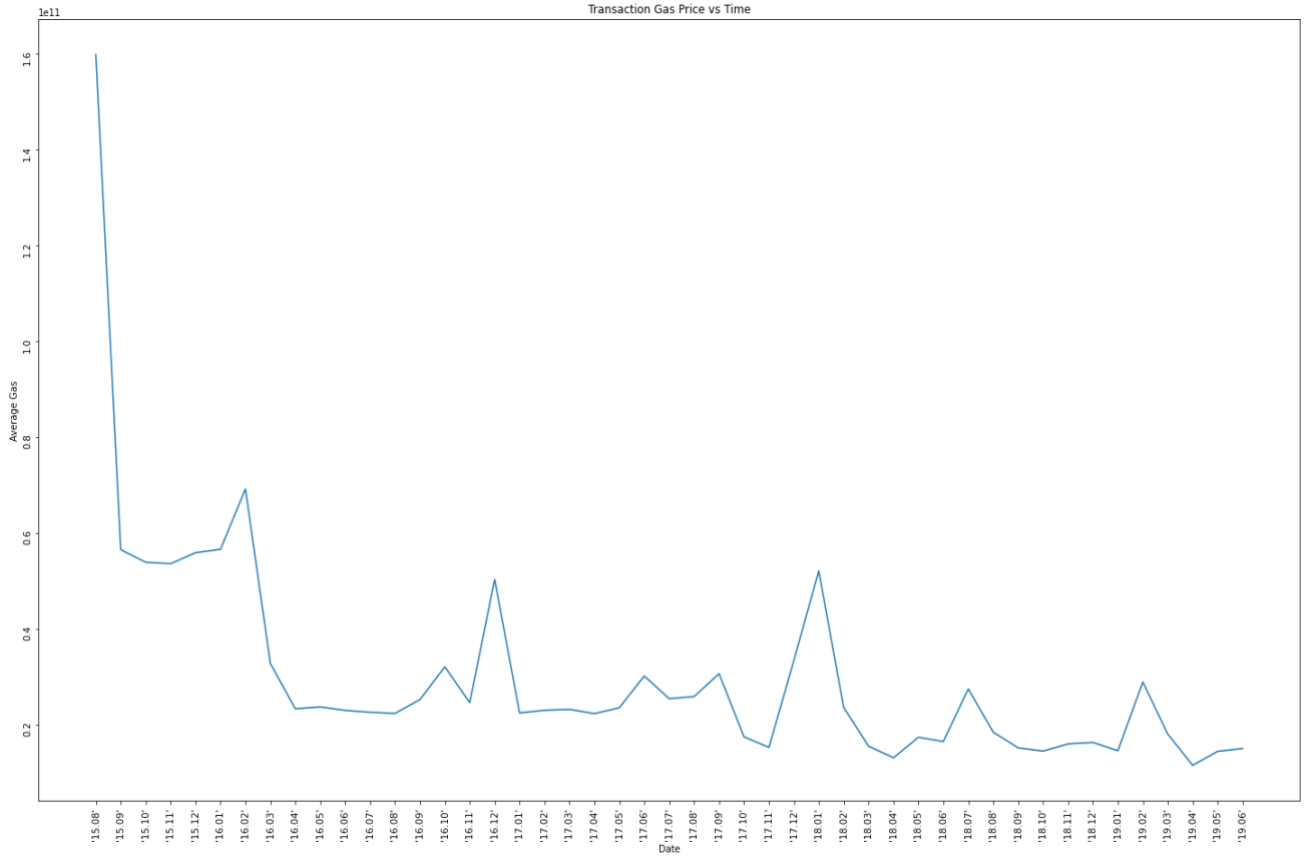
Gas Guzzlers

Job id:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7908
Input file =>

- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions
- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/contracts
- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/blocks

Procedure
I have opted to go with Spark method to solve this problem
Initially we import pyspark and time , I have planned to use 3 different datasets in this program so I have declared one function for each dataset and the dataset can be verified with the field values . Then the dataset are imported .
In the stage 1 the average gas value will be calculated for each month
In the stage 2 a output text file “aver_gas” is obtained and that file is also attached

In the stage 3 using matplotlib I have made a graph with Transaction gas price vs time and the source code is attached in notebook format("average_gas.ipynb")



Source Codes Attached:

- Average_gas.ipynb
- gas.py

Output file Attached: Aver_gas.txt

Comparative Evaluation:

Reimplement Part B in Spark (if your original was MRJob, or vice versa). How does it run in comparison? Keep in mind that to get representative results you will have to run the job multiple times, and report median/average results. Can you explain the reason for these results? What framework seems more appropriate for this task?

Input file =>

- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions
- hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/contracts

Procedure

- In this question we have implemented the part B in PySpark method
- We use two datasets
- first we import the pyspark module since we use 2 dataset I have declared a separate function for each dataset
- with address as key and value as value we perform mapping function with the help of lambda in spark to retrieve all the contract and transaction details and the top 10 contracts will be found out

Output

```
0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444:84155100809965865822726776
0xfa52274dd61e1643d2205169732f29114bc240b3:45787484483189352986478805
0x7727e5113d1d161373623e5f49fd568b4f543a9e:45620624001350712557268573
0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef:43170356092262468919298969
0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8:27068921582019542499882877
0xbfc39b6f805a9e40e77291aff27aee3c96915bdd:21104195138093660050000000
0xe94b04a0fed112f3664e45adb2b8915693dd5ff3:15562398956802112254719409
0xbb9bc244d798123fde783fcc1c72d3bb8c189413:11983608729202893846818681
0xabbb6bebfa05aa13e908eaa492bd7a8343760477:11706457177940895521770404
0x341e790174e3a4d35b65fdc067b6b5634a61caea:837900075191775562405750O
```

Source Codes Attached: spark_comp.py
Output file Attached: comparative analysis.txt

Spark Jobs:

1st time

Job ID :

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7528

Time Taken : 5mins

2nd time

Job ID:

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7650

Time Taken : 3mins

3rd time

Job ID:

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7662

Time Taken : 4mins

Average Time Taken: 3mins

Median Time Taken:4 mins

Hadoop Jobs:

1st time

Job ID:

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_5005/

Time Taken :37 mins

2nd time

Job ID:

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1648683650522_7745/

Time Taken : 30mins

3rd time

Job ID:

http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1649894236110_4077/

Time Taken :45mins

Average Time Taken: 37.33 mins

Median Time Taken: 37 mins

Looking at the average time we can find out that spark is much faster than map reduce and spark seems to be appropriate framework for this task as it takes less than 5 minutes to complete the task where as map reduce takes over 30 mins to complete this task

Fork the Chain:

Job id:
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1649894236110_5654/

Input file =>
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions

Procedure:
In this problem we are going to use the map reduce method where the days are the keys and value of the gas price will be the value and it will be counted as 1. It yields the result according to the month and year mentioned

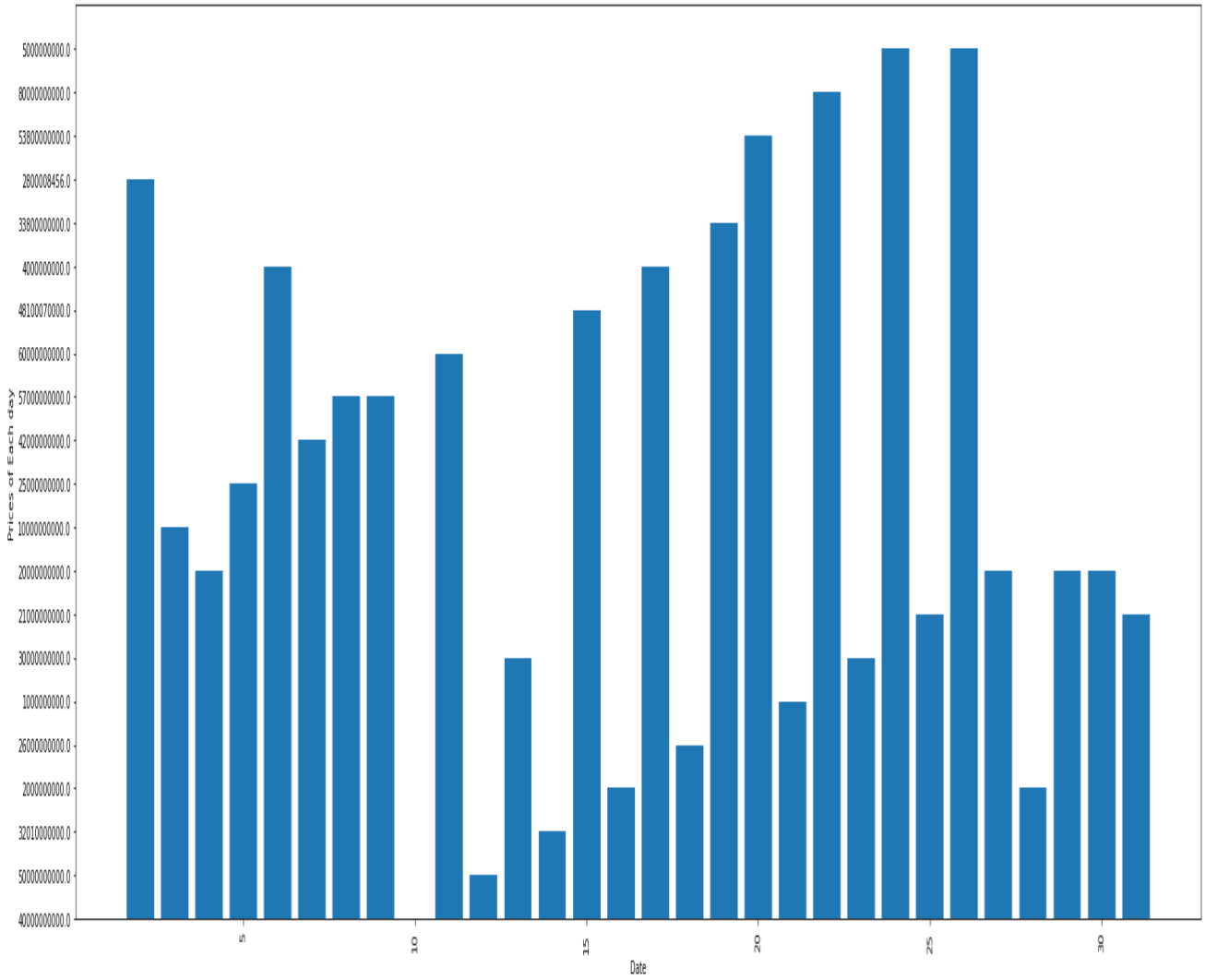
Output:

```
1       [622720, 50000000000.0]
10      [681677, 40000000000.0]
12      [872340, 50000000000.0]
14      [942559, 32010000000.0]
16      [899857, 2000000000.0]
18      [984021, 26000000000.0]
21      [1072665, 1000000000.0]
23      [918881, 30000000000.0]
25      [752361, 21000000000.0]
27      [912752, 20000000000.0]
29      [1090191, 20000000000.0]
3       [693808, 10000000000.0]
30      [1136659, 20000000000.0]
5       [693090, 25000000000.0]
7       [659068, 42000000000.0]
9       [707223, 57000000000.0]
11      [828899, 60000000000.0]
13      [909631, 30000000000.0]
15      [904346, 48100070000.0]
17      [876574, 4000000000.0]
19      [1092234, 33800000000.0]
2       [627785, 2800008456.0]
20      [986104, 53800000000.0]
22      [1094028, 80000000000.0]
24      [758234, 5000000000.0]
26      [813510, 5000000000.0]
28      [949503, 2000000000.0]
31      [946981, 21000000000.0]
4       [701834, 20000000000.0]
6       [791746, 4000000000.0]
8       [766411, 57000000000.0]
```

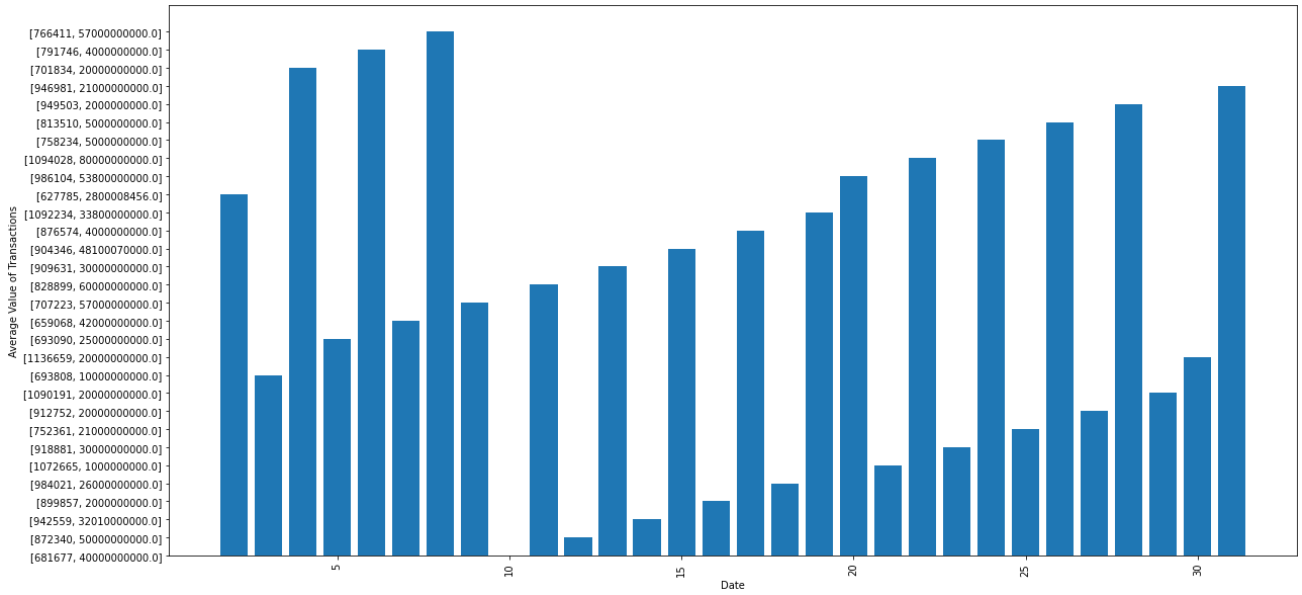
Plotting :

The plotting will be made in the google Colab notebook where we can notice the change in gas price

Gas Prices of each day in the month of December 2017



Average value of transaction of each day in the month of December 2017



Source Codes Attached:

- fork_ethereum.py
- forks-2.ipynb

Output file Attached:

- forks_in_ethereum.txt