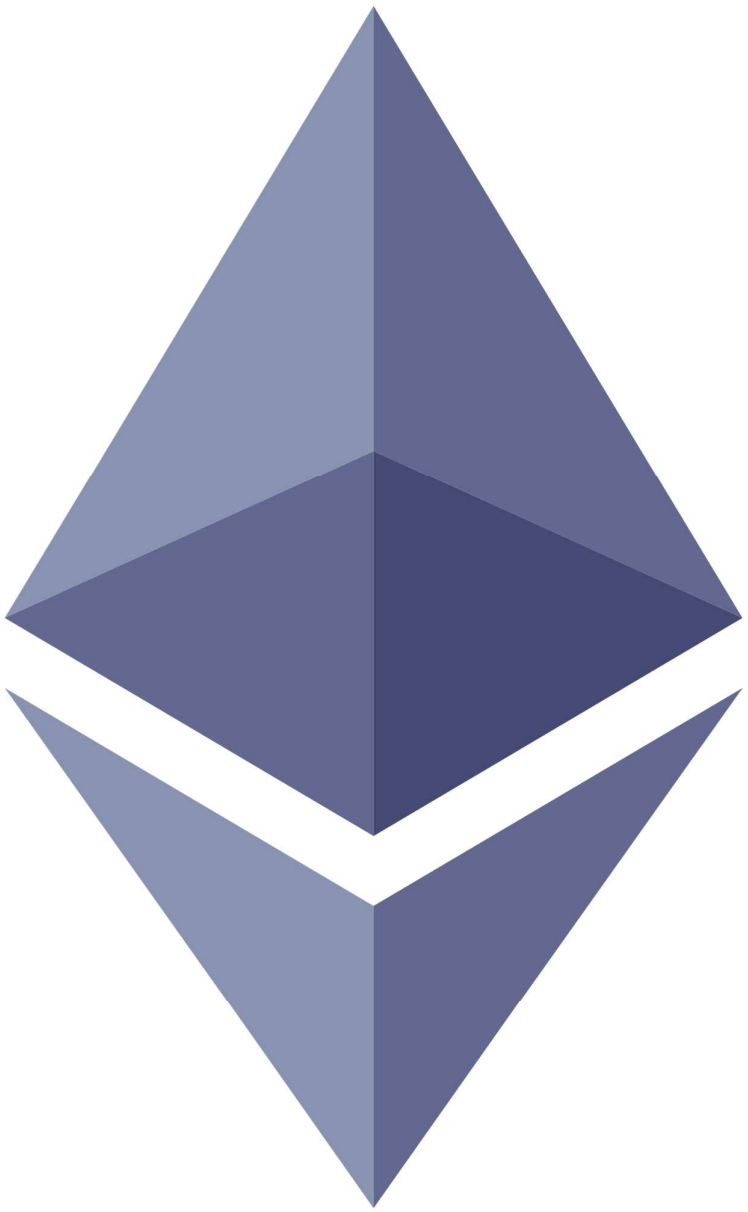
# COURSEWORK



# ETHEREUM ANALYSIS

By Anuja Jyothi Vijayakumar- 190211413

General Steps:

1. The python module is loaded
2. The environment is activated
3. The ethereum datasets are listed using the hadoop list command
4. The appropriate directory is reached using CD (Change directory command)

**Commands:**

```
bash-4.2$ module load python/3.7.7
bash-4.2$ source bdp-anu/bin/activate
(bdp-anu) bash-4.2$ hadoop fs -ls /data/ethereum
        drwxr-xr-x   - hdfs supergroup          0 2019-10-27 20:55 /data/ethereum/blocks
        drwxr-xr-x   - hdfs supergroup          0 2019-10-27 16:10 /data/ethereum/contracts
        drwxr-xr-x   - hdfs supergroup          0 2020-11-02 23:27 /data/ethereum/prices
        -rw-r--r--  10 hdfs supergroup     860400 2019-10-27 17:46 /data/ethereum/scams.json
        drwxr-xr-x   - hdfs supergroup          0 2019-10-28 01:11 /data/ethereum/transactionSmall
        drwxr-xr-x   - hdfs supergroup          0 2019-10-27 20:24 /data/ethereum/transactions
(bdp-anu) bash-4.2$ cd ecs765
(bdp-anu) bash-4.2$ cd Ethereum
```

# PART A. TIME ANALYSIS

## STEPS – For Transactions per month throughout the start and end of Dataset:

1. The python file “PARTA_transactions_per_month.py” is created
2. The input file fields are separated by comma. Hence the split () function with the parameter ‘,’ is used to identify each field and stored in “fields” variable
3. The index 7 and 8 (thus field[7] and field[8]) corresponds to timestamp and transaction count and therefore stored in variables timestmp and txncount for further processing
4. If either of these fields is not having values then the variables are kept empty for that particular instance, thus avoiding unwanted data for this particular question.
5. The mapper yields the month and the year from the epoch timestamp using the strftime() function
6. The reducer then, using the function sum() adds up all the transaction counts of that particular month-year and yields the same.
7. Thus the output of this code gives a month-year format and the sum of the transaction count in that particular month-year format
8. The dataset used as input is “BLOCKS”.
9. The code is executed using the hadoop command stated along with other commands below.
10. The output files are displayed as non-zero size files using the **hadoop fs –ls**
11. The files are merged using the **hadoop fs –getmerge** command
12. The content of the output it copied to an excel sheet and the plot generated thus shows the Transactions per month from August 2015 to May 2019

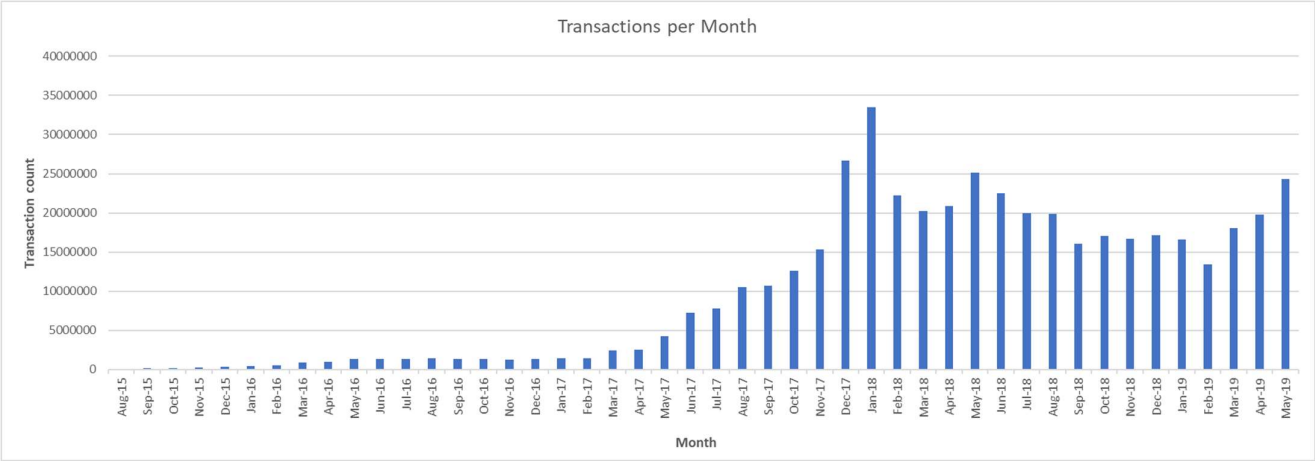
**Commands:**

```
(bdp-anu) bash-4.2$ nano PARTA_transactions_per_month.py
(bdp-anu) bash-4.2$ python PARTA_transactions_per_month.py -r hadoop --output-dir plot1 --no-cat-output
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/blocks
(bdp-anu) bash-4.2$ hadoop fs -ls plot1
        -rw-r--r--   3 ajv31 ECS640U        0 2020-12-12 17:40 plot1/_SUCCESS
        -rw-r--r--   3 ajv31 ECS640U      439 2020-12-12 17:40 plot1/part-00000
        -rw-r--r--   3 ajv31 ECS640U      421 2020-12-12 17:40 plot1/part-00001
(bdp-anu) bash-4.2$ hadoop fs -getmerge plot1 plotA1.txt
```

**From YARN Report:**

| | |
|---|---|
| Job: | job_1607539937312_4956 |
| Application: | application_1607539937312_4956 |
| URL: | http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_4956/ |
| Output directory: | hdfs:///user/ajv31/plot1 |

**Output bar Plot:**




## STEPS – For average Transactions per month throughout the start and end of Dataset

1. The python file "PARTA_avg_transactions.py" is created
2. The input file fields are separated by comma. Hence the split () function with the parameter ',' is used to identify each field and stored in "fields" variable
3. The index 6 and 3 (thus field[6] and field[3]) corresponds to timestamp and value and therefore stored in variables tmpstmp and value for further processing
4. If either of these fields is not having values then the variables are kept empty for that particular instance, thus avoiding unwanted data for this particular question.
5. The mapper yields the month and the year from the epoch timestamp using the strftime() function
6. The combiner yields the month-year and the tuple (transaction count, amount)
7. The reducer yields the average by dividing the amount by transaction count
8. Thus the output of this code gives a month-year format and the average transaction in that particular month-year format
9. The dataset used as input is "TRANSACTIONS".
10. The code is executed using the hadoop command stated along with other commands below.
11. The output files are displayed as non-zero size files using the **hadoop fs –ls**
12. The files are merged using the **hadoop fs –getmerge** command
13. The content of the output it copied to an excel sheet and the plot generated thus shows the average transactions per month from August 2015 to May 2019

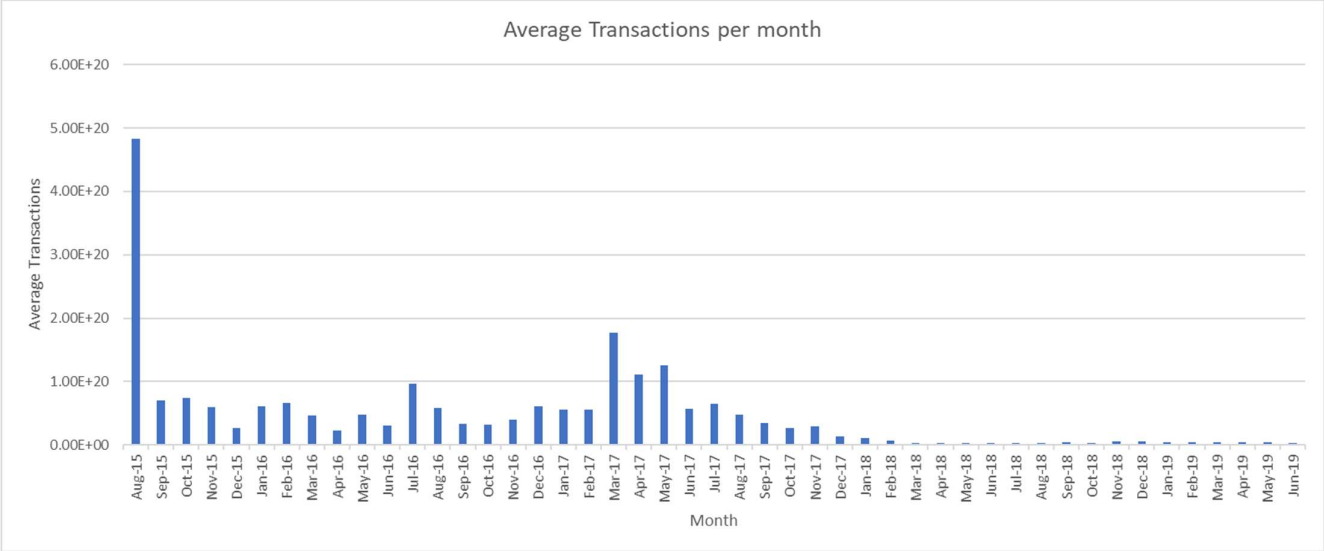
**Commands:**

```
(bdp-anu) bash-4.2$ nano PARTA_avg_transactions.py
(bdp-anu) bash-4.2$ python PARTA_avg_transactions.py -r hadoop --output-dir plot2 --no-cat-output
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions
(bdp-anu) bash-4.2$ hadoop fs -ls plot2
-rw-r--r--   3 ajv31 ECS640U        0 2020-12-12 19:47 plot2/_SUCCESS
-rw-r--r--   3 ajv31 ECS640U      769 2020-12-12 19:47 plot2/part-00000
-rw-r--r--   3 ajv31 ECS640U      740 2020-12-12 19:47 plot2/part-00001
(bdp-anu) bash-4.2$ hadoop fs -getmerge plot2 plotA2.txt
```

**From YARN Report:**

| | |
|---|---|
| job: | job_1607539937312_5295 |
| Application: | application_1607539937312_5295 |
| url: | http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_5295/ |
| Output directory: | hdfs:///user/ajv31/plot2 |

**Output bar Plot:**




## Files attached for PART A:

1. Folder: PART A
2. Code: PARTA_avg_transactions.py
3. Code: PARTA_transactions_per_month.py
4. Plot: Part A plotting.xlsx
5. Output File: plotA1.txt
6. Output File: plotA2.txt

# PART B. TOP TEN MOST POPULAR SERVICES

## Steps – For 10 most popular Services

1. The file has two pairs of map reduce jobs. First does the joining as well as the aggregation. Second does the sorting and slicing
2. The datasets used are “TRANSACTIONS” and “CONTRACTS”
3. In the first Map Reduce Task (Step 1),
    a. The **mapper** job uses the address a key to access both the dataset row and to join them.
        i. The length of the record is compared to determine to which dataset the record belong, i.e., 7 for Transactions and 5 for Contracts
        ii. The “value” is taken, from the Transaction dataset, as the value for the key-value pair.
        iii. We are passing 1 for Transaction Dataset and 2 for Contracts Dataset to differentiate the two.
        iv. Also we are passing 1 which will be later used by reducer to the get the count.
    b. The **reducer** then checks for the existence of the differentiating values 1 and 2 from the first mapper. Presence indicate valid record.
        i. The reducer yields the key and the sum of all the values
4. In the second Map Reduce Task (Step 2),
    a. The **mapper** just combines the key and the sum from the previous reducer and yields the pair with “None” as the key.

b. This value is sorted by the **reducer** in descending order using the sorted() function and the parameter reverse = True and by extracting the key, using the lambda expression, from the two values.
    i. The sorted values is then sliced to yield only the top 10 records.

**Commands:**

```
(bdp-anu) bash-4.2$ nano PARTB_top10_popular.py
(bdp-anu) bash-4.2$ python PARTB_top10_popular.py -r hadoop --output-dir etherB --no-cat-output
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/*
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/contracts/*
(bdp-anu) bash-4.2$ hadoop fs -ls etherB
-rw-r--r--   3 ajv31 ECS640U        0  2020-12-13 14:10 etherB/_SUCCESS
-rw-r--r--   3 ajv31 ECS640U      719  2020-12-13 14:10 etherB/part-00000
-rw-r--r--   3 ajv31 ECS640U        0  2020-12-13 14:08 etherB/part-00001
(bdp-anu) bash-4.2$ hadoop fs -getmerge etherB  etherB.txt
```

**From YARN Report:**

```
Job:                job_1607539937312_6792
Application:        application_1607539937312_6792
URL:                http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_6792/
Output directory: hdfs:///user/ajv31/tmp/mrjob/top10_popular.ajv31.20201213.133511.685608/step-output/0000
Job:                job_1607539937312_6861
Application:        application_1607539937312_6861
URL:                http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_6861/
Output directory: hdfs:///user/ajv31/etherB
```

**Output from the text file etherB.txt:**

```
"0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444"    84155100809965865822726776
"0xfa52274dd61e1643d2205169732f29114bc240b3"    45787484483189352986478805
"0x7727e5113d1d161373623e5f49fd568b4f543a9e"    45620624001350712557268573
"0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef"    43170356092262468919298969
"0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8"    27068921582019542499882877
"0xbfc39b6f805a9e40e77291aff27aee3c96915bdd"    21104195138093660050000000
"0xe94b04a0fed112f3664e45adb2b8915693dd5ff3"    15562398956802112254719409
"0xbb9bc244d798123fde783fcc1c72d3bb8c189413"    11983608729202893846818681
"0xabbb6bebfa05aa13e908eaa492bd7a8343760477"    11706457177940895521770404
"0x341e790174e3a4d35b65fdc067b6b5634a61caea"    83790007519177556240575000
```

## Files attached for PART B:

1. Code: PARTB_top10_popular.py
2. Output File: etherB.txt

# PART C. TOP TEN MOST ACTIVE MINERS

## Steps – For 10 most Active Miners

1. The dataset used is “BLOCKS”
2. In the first Map Reduce Task (Step 1),
    a. The **mapper** yields the miner field as the key and the size as value
        i. As we have to aggregate the size in the reducer it is type casted to integer
    b. The **reducer** yields the miner field as the key and the sum of the size as value
3. In the second Map Reduce Task (Step2),
    a. The **mapper** combines the miner and the sum of size (ie, is the key-value from reducer) and yields it along with key as “None”

b. This value is sorted by the **reducer** in descending order using the sorted() function and the parameter reverse = True and by extracting the key, using the lambda expression, from the two values.
    i. The sorted values is then sliced to yield only the top 10 records.

```
(bdp-anu) bash-4.2$ nano PARTC_top10_miners.py
(bdp-anu) bash-4.2$ python PARTC_top10_miners.py -r hadoop --output-dir etherC --no-cat-output
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/blocks
(bdp-anu) bash-4.2$ hadoop fs -ls etherC
-rw-r--r--   3 ajv31 ECS640U        0          2020-12-13 14:31 etherC/_SUCCESS
-rw-r--r--   3 ajv31 ECS640U      563          2020-12-13 14:31 etherC/part-00000
-rw-r--r--   3 ajv31 ECS640U        0          2020-12-13 14:31 etherC/part-00001
(bdp-anu) bash-4.2$ hadoop fs -getmerge etherC  etherC.txt
```

**From YARN Report:**

| | |
|---|---|
| Job: | job_1607539937312_6906 |
| Application: | application_1607539937312_6906 |
| URL: | http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_6906/ |
| Output directory: | hdfs:///user/ajv31/tmp/mrjob/PARTC_top10_miners.ajv31.20201213.143019.234803/step-output/0000 |
| Job: | job_1607539937312_6911 |
| Application: | application_1607539937312_6911 |
| URL: | http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_6911/ |
| Output directory: | hdfs:///user/ajv31/etherC |

**Output, from the text file etherC.txt:**

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

## Files attached for PART B:

1. Code: PARTC_top10_miners.py
2. Output file: etherC.txt

# PART D. DATA EXPLORATION - Scam Analysis

## STEPS – For Lucrative Scams

1. Two datasets, “TRANSACTIONS” and “SCAMS.JSON” are joined for this task
2. In the first Map Reduce Task (Step 1),
    a. In the **mapper** job,
        i. The address is taken as the key for both datasets
        ii. The value(ie., fields[3]) is taken as value from the TRANSACTIONS dataset
        iii. The category of scam is taken as value from SCAMS.JSON
        iv. The records from the dataset is differentiated by the 0 and 1 added along with the value ,i.e., (value, 0) for Transaction dataset and (value, 1) for SCAMS.JSON
    b. In the **reducer** job, if the differentiating values 0 and 1 are present then it is considered as valid record.

   i. All the valid records are then aggregated
   ii. The reducer yields the aggregated value along with **category as key**
3. In the second Map Reduce Task (Step 2),
   a. The mapper yields the key value pair as it is from the previous job
   b. The reducer groups and sums the values as per the key(the category)

**Commands:**

```
(bdp-anu) bash-4.2$ python PARTD_scamsjob1.py -r hadoop --output-dir scamsD1 --no-cat-output
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/*
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/scams.json
```

```
(bdp-anu) bash-4.2$ hadoop fs -ls scamsD1
-rw-r--r--   3 ajv31 ECS640U        0 2020-12-13 16:37 scamsD1/_SUCCESS
-rw-r--r--   3 ajv31 ECS640U       33 2020-12-13 16:37 scamsD1/part-00000
-rw-r--r--   3 ajv31 ECS640U       74 2020-12-13 16:37 scamsD1/part-00001
```

```
(bdp-anu) bash-4.2$ hadoop fs -getmerge scamsD1  scamsD1.txt
```

**From YARN Report:**

| | |
|---|---|
| Job: | job_1607539937312_7191 |
| Application: | application_1607539937312_7191 |
| URL: | http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_7191/ |
| Output directory: | hdfs:///user/ajv31/tmp/mrjob/PARTD_scamsjob1.ajv31.20201213.160454.938125/step-output/0000 |
| Job: | job_1607539937312_7302 |
| Application: | application_1607539937312_7302 |
| URL: | http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_7302/ |
| Output directory: | hdfs:///user/ajv31/scamsD1 |

**Output, from the text file scamD1.txt:**

```
"Scamming"      3.833616286244431e+22
"Fake ICO"      1.35645756688963e+21
"Phishing"      2.699937579408742e+22
"Scam"  0
```

**The about output shows that the "Scamming" has the highest aggregate value and hence it's the most lucrative scam.**

## STEPS – Scams over Time

1. Two datasets, "TRANSACTIONS" and "SCAMS.JSON" are joined for this task
2. In the first Map Reduce Task (Step 1),
   a. In the **mapper** job,
      i. The address is taken as the key for both datasets
      ii. The value(ie., fields[3]) is taken as value from the TRANSACTION dataset
      iii. The epoch time formatted and month and year is retrieved.
      iv. The category of scam is taken as value from SCAMS.JSON
      v. The records from the dataset is differentiated by the 0 and 1 added along with the value ,i.e., (value, 0) for Transaction dataset and (value, 1) for SCAMS.JSON
   b. In the **reducer** job, if the differentiating values 0 and 1 are present then it is considered as valid record.
      i. All the valid records are then aggregated
      ii. The reducer yields the aggregated value along with **date and category as key**
3. In the second Map Reduce Task (Step 2),
   a. The mapper yields the key value pair as it is from the previous job.
   b. The reducer groups and sums the values as per the key (date, category)
   c. Thus the total values of a category on a specific month and year is obtained.
4. The output is stored in a csv file, sorted by ascending date and plotted using matplotlib

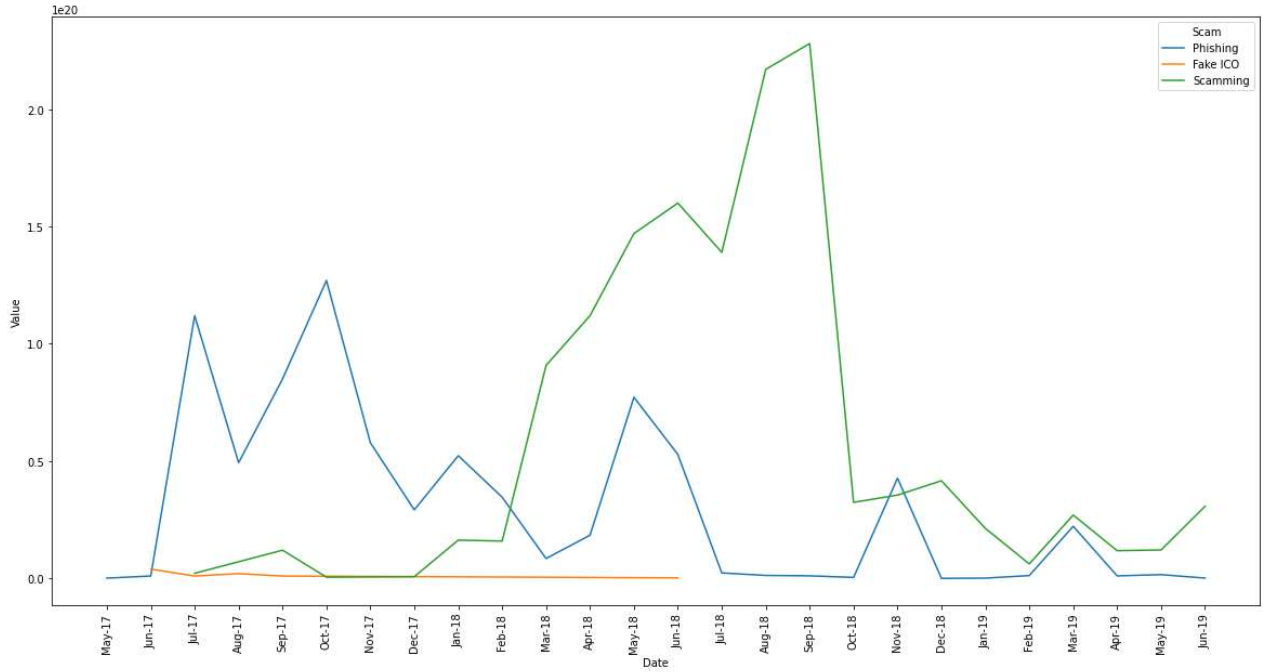
**Commands:**

```
(bdp-anu) bash-4.2$ python PARTD_scamsjob2.py -r hadoop --output-dir scamsD2 --no-cat-output
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/*
hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/scams.json
(bdp-anu) bash-4.2$ hadoop fs -ls scamsD2
-rw-r--r--   3 ajv31 ECS640U       0  2020-12-13 18:48 scamsD2/_SUCCESS
-rw-r--r--   3 ajv31 ECS640U    1055  2020-12-13 18:48 scamsD2/part-00000
-rw-r--r--   3 ajv31 ECS640U    1043  2020-12-13 18:48 scamsD2/part-00001
(bdp-anu) bash-4.2$ hadoop fs -getmerge scamsD2  scamsD2.txt
```

**From YARN Report:**

```
Job:              job_1607539937312_7651
Application:      application_1607539937312_7651
URL:              http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_7651/
Output directory: hdfs:///user/ajv31/tmp/mrjob/PARTD_scamsjob2.ajv31.20201213.181438.516987/step-output/0000
Job:              job_1607539937312_7757
Application:      application_1607539937312_7757
URL:              http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_7757/
Output directory: hdfs:///user/ajv31/scamsD2
```

**Output from plotting, using matplotlib and scamsD2_formatted.csv**:




**Fake IO** shows the existence in scams. But looking at the graph it seems that as soon as it was introduced it was effectively made offline as it was close to the value 0 for nearly a year and finally disappeared from the graph.

**Phishing** seems to have few seasonal hikes but other than that it is gradually decreasing and is staying close to 0.

**Scamming** was having a highest effect during the year 2018. Preceding and succeeding year has lesser level of Scamming

## STEPS – Scams over Time and correlation

1. Two datasets, “TRANSACTIONS” and “SCAMS.JSON” are joined for this task
2. In the first Map Reduce Task (Step 1),
    a. In the **mapper** job,
        i. The address is taken as the key for both datasets

ii. The value(ie., fields[3]) is taken as value from the TRANSACTION dataset
iii. The epoch time formatted and month and year is retrieved.
iv. The category of scam is taken as value from SCAMS.JSON
v. The records from the dataset is differentiated by the 0 and 1 added along with the value ,i.e., (value, 0) for Transaction dataset and (value, 1) for SCAMS.JSON

b. In the **reducer** job, if the differentiating values 0 and 1 are present then it is considered as valid record.
i. All the valid records are then aggregated
ii. The reducer yields the aggregated value along with **status and category as key**

3. In the second Map Reduce Task (Step 2),
a. The mapper yields the key value pair as it is from the previous job.
b. The reducer groups and sums the values as per the key(date, category)
c. Thus the total values of a category on a specific status is obtained.

**Commands:**

```
(bdp-anu) ajv31@itl210 ~/ecs765/Ethereum> python PARTD_scamsJ3.py -r hadoop --output-dir scamsD3 --no-cat-output hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/transactions/* hdfs://andromeda.eecs.qmul.ac.uk/data/ethereum/scams.json
(bdp-anu) ajv31@itl210 ~/ecs765/Ethereum> hadoop fs -ls scamsD3
-rw-r--r--  3 ajv31 ECS640U      0  2020-12-14 04:31 scamsD3/_SUCCESS
-rw-r--r--  3 ajv31 ECS640U    165  2020-12-14 04:31 scamsD3/part-00000
-rw-r--r--  3 ajv31 ECS640U     87  2020-12-14 04:31 scamsD3/part-00001
(bdp-anu) ajv31@itl210 ~/ecs765/Ethereum> hadoop fs -getmerge scamsD3 scamsD3.txt
```

**From YARN Report:**

| | |
|---|---|
| Job: | job_1607539937312_9330 |
| Application: | application_1607539937312_9330 |
| URL: | http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_9330/ |
| Output directory: | hdfs:///user/ajv31/tmp/mrjob/PARTD_scamsJ3.ajv31.20201214.035549.127636/step-output/0000 |
| Job: | job_1607539937312_9399 |
| Application: | application_1607539937312_9399 |
| URL: | http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_9399/ |
| Output directory: | hdfs:///user/ajv31/scamsD3 |

**Output, from the text file scamD3.txt:**

```
["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
```

There is high level of **Scamming** in total (88444 active ones). Also in the graph it can be seen that they are not completely offline. It has gradually increased during mid of 2018 and has fallen down towards 2019. The Scamming seems active over time. Even after 24692 has been offline, 88444 of scamming is considerably a high value.

The **Fake ICO** has remained Offline and its presence in the graph is nearly 0, which can be correlated.

**Phishing** has reached nearly 0 towards the year 2019 and this can be seen from the Inactive, Offline and Suspended counts of phishing (22, 7022 and 11 respectively) compared to the active ones (1584)

## Files attached for Scam Analysis:

1. Code: PARTD_scamsJ1.py

2. Output File: scamsD1.txt
3. Code: PARTD_scamsJ2.py
4. Output File: scamsD2.txt
5. Input file for plotting: scamsD2_formatted.csv (derived from scamsD2.txt)
6. Code for plotting: Scam_Plotting.ipynb
7. Code: PARTD_scamsJ3.py
8. Output File: scamsD3.txt

# PART D. DATA EXPLORATION - Machine Learning

## Steps:

1. The pricing information is gathered from https://public.opendatasoft.com/explore/dataset/ethereum/table/ and the dataset is downloaded as csv file
2. The dataset is stored in a dataframe
3. The VectorAssembler() takes two parameters inputCols and outputCols and the respective columns mentioned in these parameters are stored in as vector.
4. assembler.transform() then transform the dataframe object.
5. PriceUSD is to be predicted and stored separately.
6. Using randomSplit() the data is split into 70% training data and 10% testing data.
7. The LinearRegression() is applied and it is fit using fit()
8. The model is evaluated using evaluate()
9. The model is checked for root mean squared error value and the r squared value
10. The r squared value is nearly 70% (0.697)

**Commands:**

```
(bdp-anu) bash-4.2$ hadoop fs -copyFromLocal ./ethereum_formatted.csv ethereum_price.csv
(bdp-anu) bash-4.2$ hadoop fs -ls
          -rw-r--r--   3 ajv31 ECS640U     876931 2020-12-13 22:33 ethereum_price.csv
(bdp-anu) bash-4.2$ module load hadoop/cdh-6.3.0
(bdp-anu) bash-4.2$ spark-submit PARTD_priceforecast.py
```

**Output, from the text file pforecastD3.txt:**

```
123.07177763559044
0.6972296579701295
+--------------------+-----------------+
|            features|       prediction|
+--------------------+-----------------+
|[7.233504128E7,12...|79.56793669941442|
|[7.266255112E7,39...|80.11363104164434|
|[7.282109737E7,55...|80.74002537188903|
|[7.288930112E7,45...|79.09343646033494|
|[7.293093378E7,50...|79.42156276882224|
|[7.294932487E7,51...|79.35892152979477|
|[7.304158503E7,54...|79.01524319015914|
|[7.311936566E7,55...|78.53245974962203|
|[7.319945081E7,54...|77.89262209356832|
|[7.325121034E7,54...|77.50453900425975|
|[7.327682566E7,53...|77.15454438769245|
|[7.332857941E7,50...|76.42245500797185|
|[7.338166081E7,50...|76.04595099028029|
|[7.340964347E7,62...|77.20453528750772|
|[7.343629534E7,61...|76.91336781079667|
|[7.35690455E7,585...|75.54028819437417|
|[7.361877081E7,60...|75.35913385842991|
|[7.3668233E7,5782.0]|74.72421645826876|
|[7.372256816E7,59...|74.51095533846603|
|[7.377450206E7,66...|74.88299320622048|
+--------------------+-----------------+
only showing top 20 rows
```

## Files attached for Machine Learning:

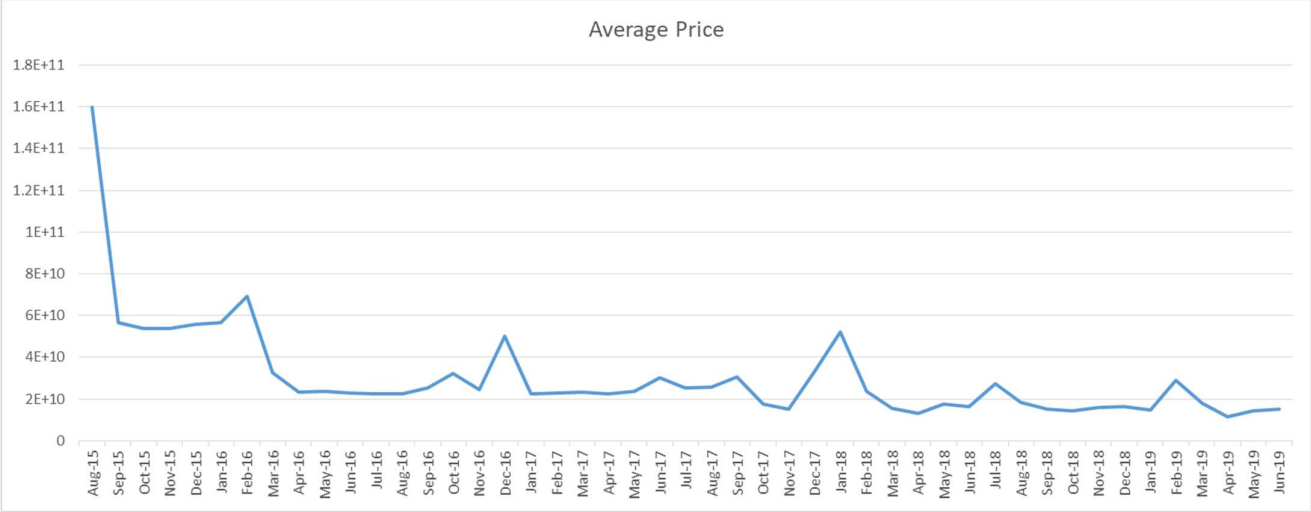
1. Input File: ethereum_formatted.csv
2. Code: PARTD_priceforecast.py
3. Output File: pforecastD3.txt

# PART D. DATA EXPLORATION - ANALYSIS - Gas Guzzlers:

## Steps - Price over time:

1. The TRANSACTION dataset is accessed using the function textFile() by passing file path as parameter
2. The valid records are identified by checking the length of the record, ie 7. The rest are discarded.
3. Using map() the dataset is split as columns and stored in a list based on the parameter which is passed to the split() function. Here the parameter is comma separator
1. Using map() The date and the price are combined along with '1' (1 appended to get the count at the later stage)
4. Using strftime() the month and year are extracted from the epoch timestamp.
5. Using reduceByKey() all the values of prices are added up and the counts are added.
6. Using these "sum of prices" and the "total count", the average price per month for all the years is calculated.
7. URL: http://andromeda.student.eecs.qmul.ac.uk:8088/cluster/app/application_1607539937312_9757
8. The below graph is obtained by storing the output in excel sheet and plotting it.

**Output Plot:**



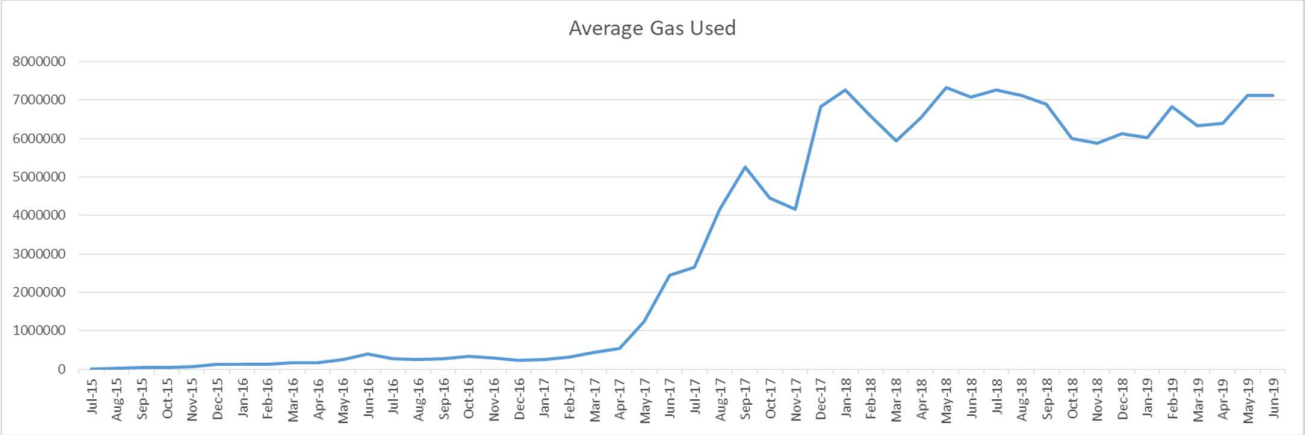

Overall the price seems to be having a decreasing trend. But if we observe the spikes inbetween they appear mostly during the end of the year or the beginning of the year.

## Steps - Gas used over time:

1. The BLOCK dataset is accessed using the function textFile() by passing file path as parameter
2. The valid records are identified by checking the length of the record, i.e. 9. The rest are discarded.
3. Using map() the dataset is split as columns and stored in a list based on the parameter which is passed to the split() function. Here the parameter is comma separator
4. Using map() The date and the gas use are combined along with '1' (1 appended to get the count at the later stage)
5. Using strftime() the month and year are extracted from the epoch timestamp.

6. Using reduceByKey() all the values of gas used are added up and the counts are added.
7. Using these “sum of gas used” and the “total count”, the average gas per month for all the years is calculated.
8. URL: http://andromeda.student.eecs.qmul.ac.uk:8088/cluster/app/application_1607539937312_9807
9. The below graph is obtained by storing the output in excel sheet and plotting it.

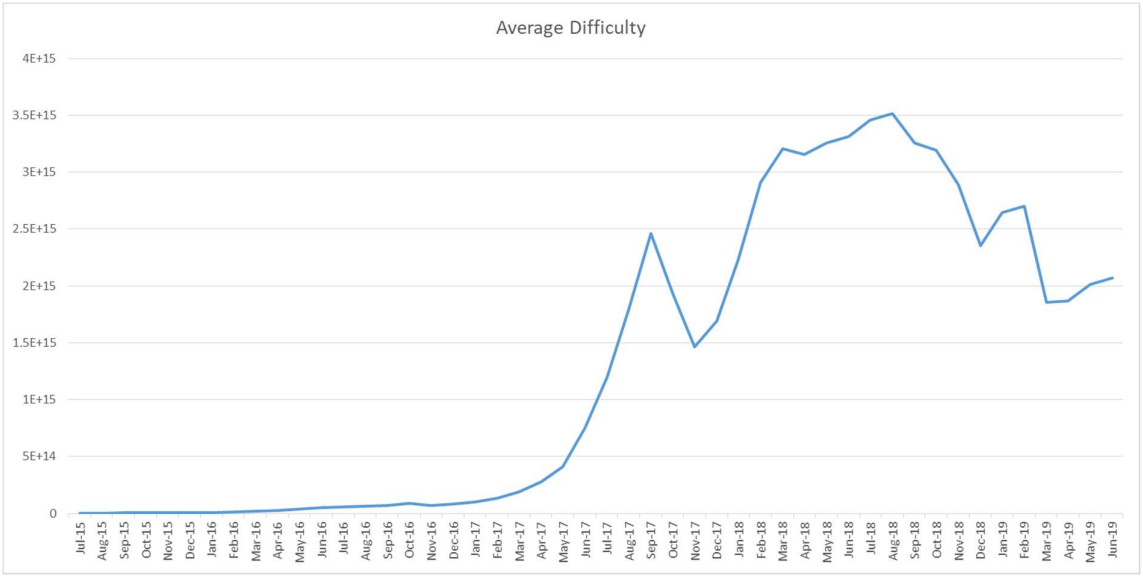
**Output Plot:**




## Steps - Complexity over time:

1. The BLOCK dataset is accessed using the function textFile() by passing file path as parameter
2. The valid records are identified by checking the length of the record, i.e. 9. The rest are discarded.
3. Using map() the dataset is split as columns and stored in a list based on the parameter which is passed to the split() function. Here the parameter is comma separator
4. Using map() the date and the difficulty are combined along with ‘1’ (1 appended to get the count at the later stage)
5. Using strftime() the month and year are extracted from the epoch timestamp.
6. Using reduceByKey() all the values of difficulty are added up and the counts are added.
7. Using these “sum of difficulty” and the “total count”, the average difficulty per month for all the years is calculated.
8. URL: http://andromeda.student.eecs.qmul.ac.uk:8088/cluster/app/application_1607539937312_9812
9. The below graph is obtained by storing the output in excel sheet and plotting it.

**Output Plot:**



## Steps – Correlation with PART B:

1. Part B output contains the top ten most popular services.
2. So in order to find the correlation between Gas and popular services we need to obtain the corresponding gas used
3. Transactions, Blocks and the Output of Part B are accessed using the function textFile() by passing file path as parameter
10. The valid records are identified by checking the length of the record,
    a. 9 for block
    b. 7 for transaction
    c. 2 Part B
    d. The rest are discarded.
4. Using map(), extracted the block number and gas used as key and value from Block data
5. Aggregate all values for each key using reduceByKey() function
6. Using map(), extracted the block number and address as key and value from Transaction data
7. The block number is used as the joining key for both datasets.
8. Using map(), to distinguish the partB "topten" is appended
9. Using leftOuterJoin() it is then joined with the output of Part B to get distinguish the common records between the two datasets.
10. All records containing "topten" is filtered out and thus obtain the gas used for top 10 most popular services.
11. URL: http://andromeda.student.eecs.qmul.ac.uk:8088/cluster/app/application_1607539937312_9815

**Top 10 Services from PART B:**

```
"0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444"    84155100809965865822726776
"0xfa52274dd61e1643d2205169732f29114bc240b3"    45787484483189352986478805
"0x7727e5113d1d161373623e5f49fd568b4f543a9e"    45620624001350712557268573
"0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef"    43170356092262468919298969
"0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8"    27068921582019542499882877
"0xbfc39b6f805a9e40e77291aff27aee3c96915bdd"    21104195138093660050000000
"0xe94b04a0fed112f3664e45adb2b8915693dd5ff3"    15562398956802112254719409
"0xbb9bc244d798123fde783fcc1c72d3bb8c189413"    11983608729202893846818681
"0xabbb6bebfa05aa13e908eaa492bd7a8343760477"    11706457177940895521770404
"0x341e790174e3a4d35b65fdc067b6b5634a61caea"    83790007519177556240575oo
```

**Output - Gas used by top 10 services:**

```
(u'0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444', 1089070875674)
(u'0xfa52274dd61e1643d2205169732f29114bc240b3', 8421094348549)
(u'0x7727e5113d1d161373623e5f49fd568b4f543a9e', 2618517255200)
(u'0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef', 5944027679759)
(u'0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8', 1760344386753)
(u'0xbfc39b6f805a9e40e77291aff27aee3c96915bdd', 331425681779)
(u'0xe94b04a0fed112f3664e45adb2b8915693dd5ff3', 8323721708803)
(u'0xbb9bc244d798123fde783fcc1c72d3bb8c189413', 171534308083)
(u'0xabbb6bebfa05aa13e908eaa492bd7a8343760477', 3940702747102)
(u'0x341e790174e3a4d35b65fdc067b6b5634a61caea', 42561328)
```

## Files attached for Gas Guzzlers:

1. Code: PARTD_gasJ1.py
2. Output file: Guzzler1.txt
3. Code: PARTD_gasJ2.py
4. Output file: Guzzler2.txt
5. Code: PARTD_gasJ3.py
6. Output file: Guzzler3.txt
7. Code: PARTD_gasJ4.py

8. Output file: Guzzler4.txt

# PART D. DATA EXPLORATION - ANALYSIS - Comparative Evaluation:

## Steps:

1. This is the spark implementation of Part B – "10 most popular Services"
2. "Transactions" and "Contracts" are the two input datasets used. These input files are accessed via the function textFile() and specifying the path of the datasets in parenthesis
3. Function created to remove any malformed lines by checking the length of the record. Record having length 7 is considered valid record for Transactions
4. Function created to remove any malformed lines by checking the length of the record. Record having length 5 is considered valid record for Contracts
5. Function created to remove "none" values in the dataset.
6. Using map() transformation function the address and values are mapped as key and value respectively, for the transaction dataset. Valid record check is done along with this.
7. The reduceByKey() perform the reduce operation where we aggregate all the values of the same key.
8. Map the address as the key and pass a string "contract" as the value from the contracts dataset. Valid record check is done along with this.
9. Left outer join is performed using the "address" key. The join is done on the previous reduce on the transactions dataset.
10. Records having none values are removed.
11. We then sort the data in descending order and take the first 10 values using takeOrdered() function.
12. Write the output to the file

**Commands:**

```
ajv31@itl210 ~/ecs765/Ethereum> spark-submit PARTD_evaluation.py
```

## Comparison:

**SPARK:**

**Run 1:** http://andromeda.student.eecs.qmul.ac.uk:8088/cluster/app/application_1607539937312_9611 **Elapsed**: 6mins, 45sec

**Rerun 1:** http://andromeda.student.eecs.qmul.ac.uk:8088/cluster/app/application_1607539937312_9623 **Elapsed**: 4mins, 52sec

**Rerun 2:** http://andromeda.student.eecs.qmul.ac.uk:8088/cluster/app/application_1607539937312_9642 **Elapsed**: 5mins, 50sec

**Rerun 3:** http://andromeda.student.eecs.qmul.ac.uk:8088/cluster/app/application_1607539937312_9700 **Elapsed**: 6mins, 18sec

**MrJob:**

**Run 1: Step1:** http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_6792/ **Elapsed**: 31mins, 38sec

**Run 1: Step2:** http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_6861/ **Elapsed**: 2mins, 53sec

**Rerun 1: Step1:** http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_9863/ **Elapsed**: 36mins, 59sec

**Rerun 1: Step2:** http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_9907/ **Elapsed**: 2mins, 50sec

**Rerun 2: Step1:** http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_9912/ **Elapsed**: 23mins, 30sec

**Rerun 2: Step2:** http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_9930/ **Elapsed**: 2mins, 42sec

**Rerun 3: Step1:** http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_9950/ **Elapsed**: 25mins, 53sec

**Rerun 3: Step2:** http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1607539937312_9957/ **Elapsed**: 2mins, 41sec

From the above information it can be seen the spark execution is faster than that of Map Reduce job. There is a large gap between SPARK and MrJob with respect to the time taken.

The Map Reduce job store intermediate operations in the Disk, while the in Spark it happens only once towards the end of the job. So the time spend is reduced considerably.

## Files attached for Comparative Evaluation:

1. Code: PARTD_evaluation.py
2. Output File: evaluateD1.txt