Cara Evangeline Chandra Mohan
190539421

**Big data Processing Coursework : Ethereum**

# Part A:Time Analysis

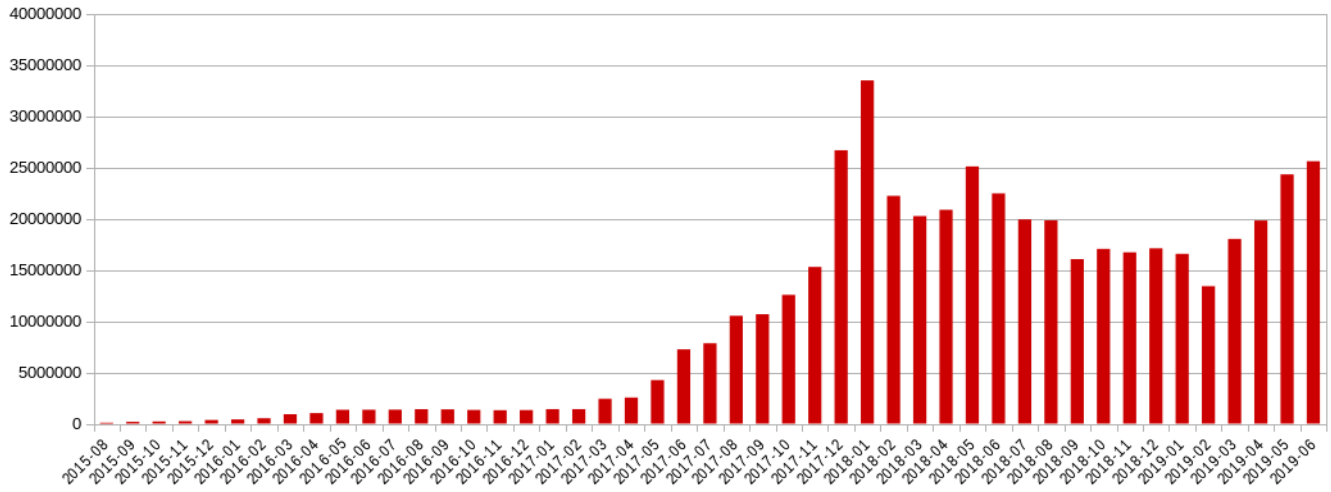
*Application Id :*
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1574975221160_2701/

*Explaination:*

- The Mapper outputs the (key, value) pair and here in this question the key of the mapper program is (month, year) and the value is the number of transactions that occurred in a particular year and month.
- The reducer outputs the total number of transactions that took place in a particular year and month.

*Dataset used:*
From hadoop: data/ethereum/blocks

*Graph:*




*Analysis:*
As seen in the above graph we can observe that in the intial stages of ethereum startup, the number of transactions is very less and gradually it is increasing and again there are few inconsistencies in the usage of ethereum.

Cara Evangeline Chandra Mohan
190539421

# Part B:Top Ten most popular services

## Job 1: Initial Aggregation

*Application Id:*
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1574975221160_7231/

*Dataset used:*
From Hadoop: /data/ethereum/transactions

*Explaination:*

- Mapper outputs the address and value_in_wei
- Reducer outputs the aggregate value_in_wei for the addresses given by the mapper.

## Job 2: Joining transactions/ contracts and filtering

*Application Id:*
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1574975221160_3541/

*Dataset used:*
From Hadoop: /data/ethereum/contracts and the output of previous job (ie. **Job1: Initial Aggregation**)

*Explaination:*

- Here repartition join is performed on the above given dataset.
- Mapper for contracts:
  Outputs (address, block) as (key, value) pair
- Mapper for previous job output:
  Outputs (address, value_in_wei) as (key, value) pair
- Reducer outputs (address, block) as key and value_in_wei as value

## Job 3:Top Ten

*Application Id:*
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1574975221160_3603/

Cara Evangeline Chandra Mohan
190539421

*Dataset Used:*
From Hadoop: Output of previous job (ie. **Job 2: Joining transactions/ contracts and filtering**)

*Explaination:*

- The task here is to find the top 10 contracts which has the highest value_in_wei and hence we need to sort out the records of the previous output in descending order
- Mapper outputs (address and value_in_wei) as value and key is None
- Combiner sorts value_in_wei in descending order using the function 'sorted'
- Reducer outputs the top 10 value in required format.

**Output:**

"0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444 - 841551008099658658227267 76 " null
"0xfa52274dd61e1643d2205169732f29114bc240b3 - 45787484483189352986478805 " null
"0x7727e5113d1d161373623e5f49fd568b4f543a9e - 45620624001350712557268573 " null
"0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef - 43170356092262468919298969 " null
"0x6fc82a5fe25a5cdb58bc74600a40a69c065263f8 - 27068921582019542499882877 " null
"0xbfc39b6f805a9e40e77291aff27aee3c96915bdd - 21104195138093660050000000 " null
"0xe94b04a0fed112f3664e45adb2b8915693dd5ff3 - 15562398956802112254719409 " null
"0xbb9bc244d798123fde783fcc1c72d3bb8c189413 - 11983608729202893846818681 " null
"0xabbb6bebfa05aa13e908eaa492bd7a8343760477 - 11706457177940895521770404 " null
"0x341e790174e3a4d35b65fdc067b6b5634a61caea - 83790007519177556240575 00 " null

# Part C: Data Exploration

## Scams

**Datasets used:**
From Hadoop: /data/ethereum/scams.json and /data/ethereum/transactions

**Task : Lucrative Form of Scam**

1. Convert .json to .csv file
2. Filter the required columns (ie. address, category)
3. Perform Repartition join of scams.csv and transaction
4. Top scams

Cara Evangeline Chandra Mohan
190539421

- *Filtering the required columns:*

*Application Id:*
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1574975221160_7028/

*Explaination:*

- Mapper outputs the category of scam as key and address as value

- *Repartition Join:*

*Application Id:*
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1574975221160_7070/

*Explaination:*

- Mapper outputs (address,category) for the data of the previous filtered output and outputs (address, value_in_wei) for the transaction dataset of ethereum
- Reducer outputs ((address, category), value_in_wei)

- *Top Scams:*

*Application Id:*
http://andromeda.student.eecs.qmul.ac.uk:8088/proxy/application_1574975221160_7178/

*Explaination:*

- Mapper outputs (category, value_in_wei) as (key, value) pair
- Reducer outputs the total value_in_wei for every category of scams given

**Output:**

"Fake_ICO" 5272387075000000000
"Scamming" 1310809509244117714543
"" 100000000000000000
"Phishing" 1120460711004853613600

Cara Evangeline Chandra Mohan
190539421

## Gas Guzzlers

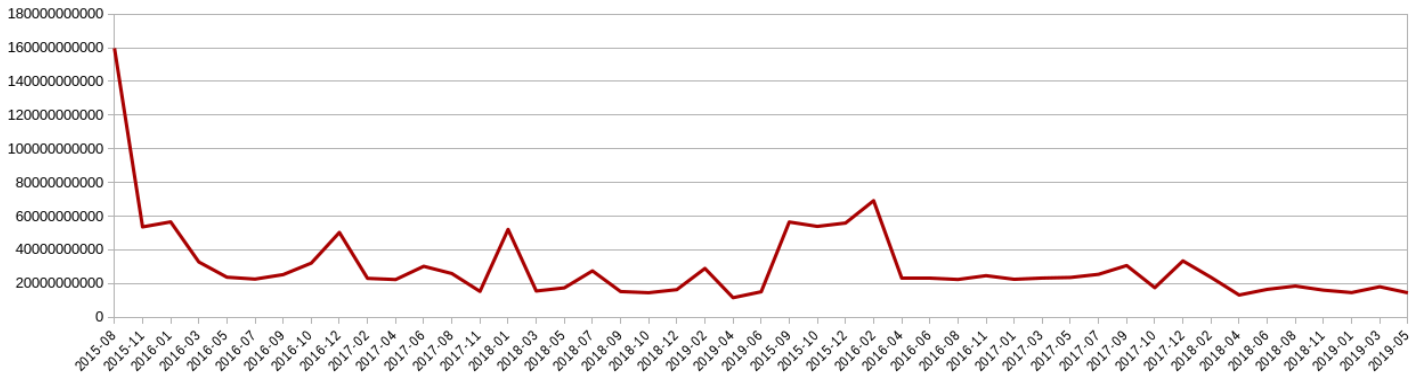
**Task: For any transaction on Ethereum a user must supply gas. How has gas price changed over time?**

*Dataset used:*
From hadoop: /data/ethereum/transactions

*Explaination:*

- Mapper outputs the gas_price for a particular (month, year)
- Reducer outputs the average of gas_price for a particular year and month

*Graph:*




*Analysis:*
Initially the cost of gas is too high but later as the usage of ethereum is becoming more common there is a decline in the rate of gas_price.

## Comparison of Spark and MapReduce

**Result:**

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

Cara Evangeline Chandra Mohan
190539421

- application_1575381276332_2284 5.3 min
- application_1575381276332_2328 6.1 min
- application_1575381276332_2372 5.3 min
- application_1575381276332_2408 4.6 min
- application_1575381276332_2441 5.4 min

Average time taken = 5.34 minutes

Whereas when we run the same task in Map reduce framework the approximate time taken is roughly 1 hour 30 minutes.

Spark Framework is more better than Mapreduce for this task of finding the top 10 contracts. Spark is much faster than Mapreduce framework because spark does In-memory processing and avoids unnecessary I/O operations. Spark uses transformations where number of writes to disk is reduced.