Analysis of Ethereum Transactions and Smart Contracts

Milan Babu Supriya

Big Data Science

School of Engineering and Computer Science

Queen Mary University of London

Mile End Road, London E1 4NS, UK

Introduction

The aim of the coursework is analyzing the Ethereum transactions and contracts using Spark. The analysis includes aggregating the required fields from large transaction datasets and finding various anomalies, popular transactions etc. using different spark methods. The data is obtained from the Ethereum-parvulus repository. Data is provided in csv and json formats.

Part A

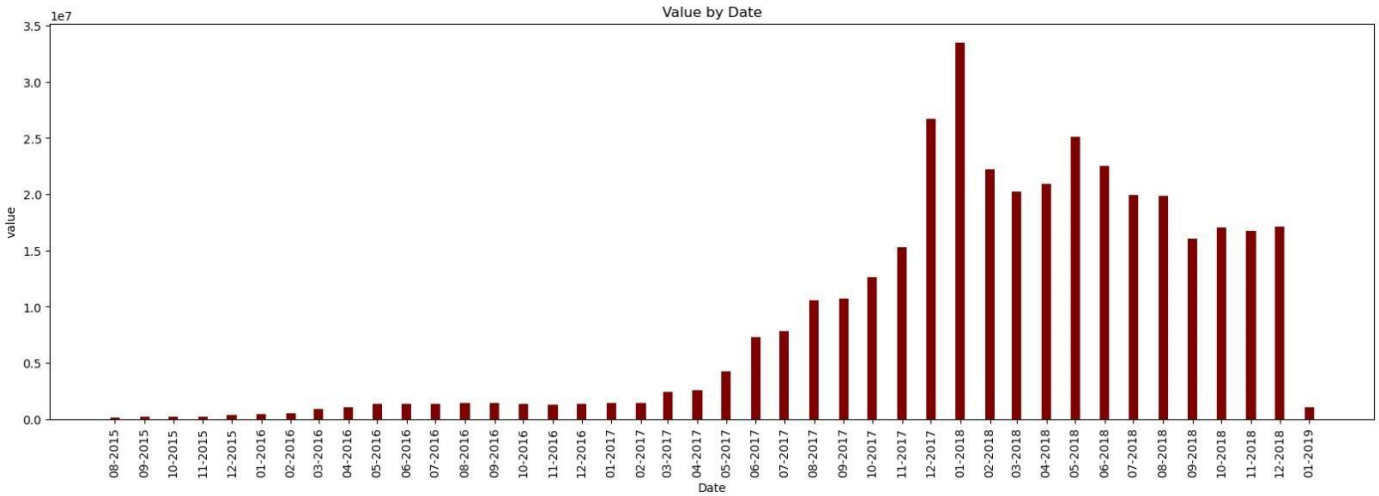
Task is to use the transactions dataset to find the count of total ether value per month and the average count of ether used per month from August 2015 till January 2019. Two bar plots are created using these values to study and understand the variation in value monthly.

Count: The transaction dataset is called from the Ethereum-parvulus repository as text file. Good_line function is used to change the datatype of required columns into float type, here ether value and timestamp columns and filtered the dataset and stored in clean_lines. Using map function timestamp is converted into month and year using strftime. The mapped values are reduced by key using lambda function which adds each value to get the total count per month.

The results are output as a text file and dumped into s3_bucket under count_.txt.

Plot 1

The monthly count of ether is read as a text file(count_.txt) and converted to a data frame using pandas. The data frame is grouped by date and plotted using matplotlib as a bar graph. The graph is shown below:



From the graph we can say that there was a gradual from 2015 to 2017 and fluctuations in value from the beginning of 2018, with a low value in 2019.

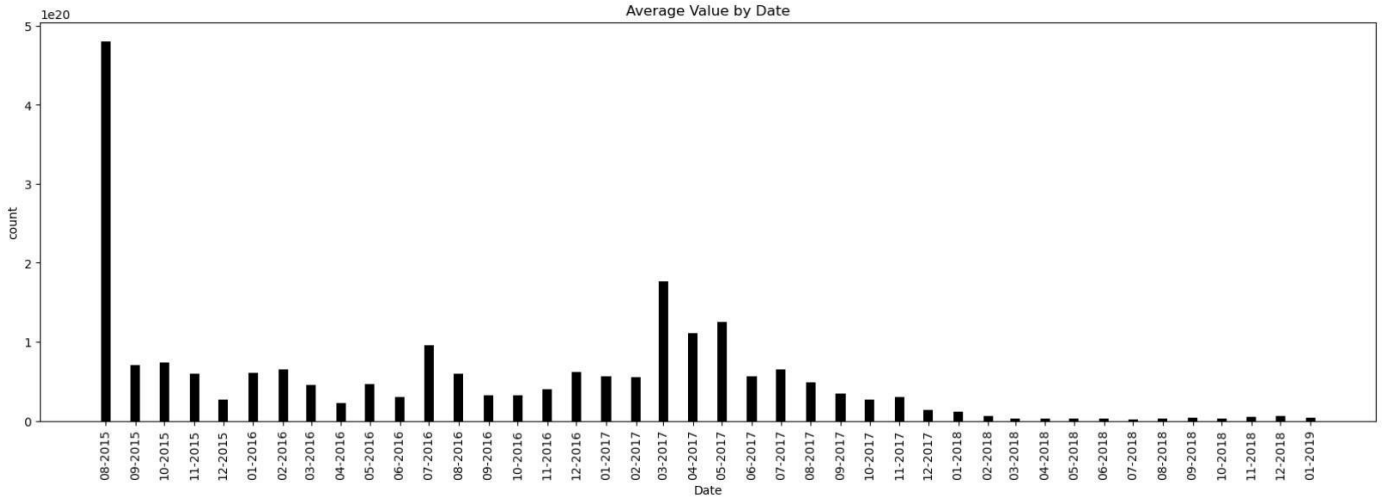
Average Count: To get the average of the values, the data is mapped with timestamp and ether value. Timestamp is changed into month and year using strftime method of time function and

reduced by key to get the total count of value. The obtained values are joined with the monthly data from task 1 and mapped with lambda function where each ether value is divided by total count to get the average count per month.

The results are output as a text file and dumped into s3_bucket under avg_.txt.

Plot 2

The avg count of ether is read as a text file(avg_.txt) and converted to data frame using pandas. The data frame is grouped by date and plotted using matplotlib as a bar graph. The graph is shown below:



From the graph we can see sudden decrease in value after 2015 and noticeable increase till 2018 and the values are dropped to minimum from second month of 2018 to beginning of 2019.

Part B

The task is to find the top 10 smart contracts by the total Ether received.

Both transactions and contract datasets are called from Ethereum-parvulus repositoy and separate good_lines functions are used to filter the data into the required format. The to_address field and the ether value is mapped from transactions and joined with the address field in contracts dataset. The joined data is mapped to get the ether value and address and reduced by key to get the total count of contracts using lambda function which sum each value. From these values using takeOrdered method for 10 with key is defined as a lambda function (lambda x:x[1]) which gives the highest values, the top 10 contracts.

The required contracts are dumped into bucket as txt file(contractTop10.txt).

Given below are the top 10 contracts.

ADDRESS	ETHEREUM VALUE
0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444	8.415536369994054e+25
0x7727e5113d1d161373623e5f49fd568b4f543a9e	4.562712851291586e+25
0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef	4.255298913641433e+25
0xbfc39b6f805a9e40e77291aff27aee3c96915bdd	2.1104195138094623e+25
0xe94b04a0fed112f3664e45adb2b8915693dd5ff3	1.5543077635266931e+25
0xabbb6bebfa05aa13e908eaa492bd7a8343760477	1.0719485945629484e+25
0x341e790174e3a4d35b65fdc067b6b5634a61caea	8.379000751917755e+24
0x58ae42a38d6b33a1e31492b60465fa80da595755	2.902709187105732e+24
0xc7c7f6660102e9a1fee1390df5c76ea5a5572ed3	1.2380861145200434e+24
0xe28e72fcf78647adce1f1252f240bbfaebd63bcc	1.172426432515822e+24

Part C

The task is to obtain the top 10 miners using the blocks dataset.

The blocks dataset is called from the repository, good_line filters the dataset with the size column as float. The miner and the size field is mapped by splitting using commas and reduced by key to get the total count of miners with lambda function(a,b:a+b) and stored in minerList variable. Using takeOredered(10) method top 10 miners are obtained and stored as top10_min.

The result is stored in bucket as a text file(minerTop.txt).

Given below are the top 10 miners.

ADDRESS	TOP MINING
0xea674fdde714fd979de3edf0f56aa9716b898ec8	17453393724.0
0x829bd824b016326a401d083b33d092293333a830	12310472526.0
0x5a0b54d5dc17e0aadc383d2db43b0a0d3e029c4c	8825710065.0
0x52bc44d5378309ee2abf1539bf71de1b7d7be3b5	8451574409.0
0xb2930b35844a230f00e51431acae96fe543a0347	6614130661.0
0x2a65aca4d5fc5b5c859090a6c34d164135398226	3173096011.0
0xf3b9d2c81f2b24b0fa0acaaa865b7d9ced5fc2fb	1152847020.0
0x4bb96091ee9d802ed039c4d1a5f6216f90f81b01	1134151226.0
0x1e9939daaad6924ad004c2560e90804164900341	1080436358.0
0x61c808d82a3ac53231750dadc13c777b59310bd9	692942577.0

PART D

Popular Scams

There exist many types of scams in Ethereum transactions, here the first task is to find out the ID of most lucrative scam, most lucrative category (type of scam) and plot a graph analyzing the change of ether received over time.

The scam dataset is available in Ethereum-parvulus repository in json format. The file is called from the repository and loaded as json. The required fields such as unique ID, addresses, status and category are mapped as key value pairs using flatMap function.

Transactions dataset is also called from repository and good line function defined is called to get the wanted fields such as addresses, ether value and timestamp. These values are mapped and joined with scams data using the join method(joinedrdd).

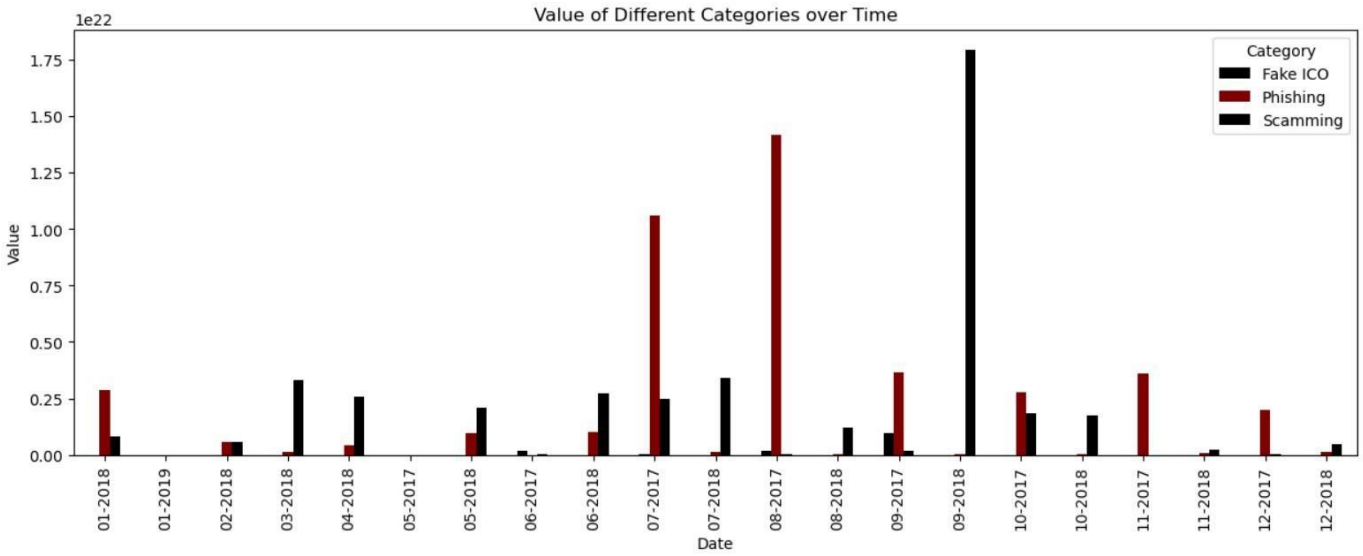
LucrativeID: The joined rdd is mapped to get the addresses used for scams and ether value, these values are reduced by key to get the total count of ether and take the top value to get most lucrative ID.

LucrativeCategory: The joined rdd is mapped to get the categories of scams and ether value, these values are reduced by key to get the total count of ether, that is the type of scam which has received most ether value. And take the top value to get most lucrativeCategory.

changeWithTime: The joinedrdd is mapped to aggregate the ether value and scam categories per month on the available year period. This is reduced by key to find the total count of ether value per month. The graph is plotted with the outputs obtained.

Plotting: The output is read into a python file using pandas and created a dataframe and grouped by date and category. The dataframe is plotted using matplotlib bar plot.

All the three results are dumped to the s3_bucket in txt file format. The graph below displays the change in ether value over time.



Data Overhead

The task is to find out how much space can be saved if unwanted or not strictly necessary information is removed from blocks dataset. The blocks dataset contains many information (in columns) such as logs_bloom, sha3_uncles, transactions_root, state_root, receipts_root which can be removed.

Blocks dataset is called from repository and good_line is called on the dataset to filter the data. The filtered data is mapped to get the length of each value in the specified columns and subtracted from two. This is done because each value is in hex strings with each character after first two requires four bits. And the values are multiplied by 4. These values are reduced by key using lambda function which gives the total count.

The result is stored in s3_bucket as a txt format under dataoverhead.txt.

The obtained result is [[1, 21504003072]].

Gas Guzzlers

The task is to find out how has gas price changed over time and provide a graph showing how gas prices have changed over time. Also, a graph showing how gas used for contract transactions has changed over time and identify most popular contracts use more or less than the average gas used.

Average Gas Price over time: The transactions dataset is called from Ethereum-parvulus repository and dataset is filtered using transaction_line function.
The filtered data is mapped with timestamp and gas price; timestamp is changed into month and year format. This is reduced by key using lambda function which sums the gas price to get total count per month.

The result is dumped into s3_bucket as avg_gasprice.txt.

The graph is given below:

The avg_gasprice is read as a text file and changed into a dataframe using pandas. The datatype of gasprice is changed into float and groupby date. The graph is plotted using matplolib line graph.

Average Gas Used: The contracts dataset is called from repository and filtered with contracts_line function. The filtered data is mapped to get the contracts address column and joined with the transactionRDD1 created on task 1. The joined rdd is mapped with date as key and gas from contracts as value. This is reduced by key to get the total count. The average gas used is obtained by mapping the reduced rdd using lambda function which has date as first value and each gas is divided by total value. The average gas used is sorted by key giving the condition ascending =True.

The result is dumped into s3_bucket as avg_gasused.txt.

Change of gas overtime: The top 10 contracts obtained from Part B is uploaded into bucket and called as textfile. This file is mapped to get the top contract address. The joinRDD which contains the joined rdd of transactions and contracts data is mapped to get the address as key and time, gas as value. These values are joined with the top 10 contracts and reduced by key to get the total count and stored in a variable contractWithGas. contractWithGas is mapped with lambda function giving the date and the average gas count.

The result is dumped into s3_bucket as contractWithGas.txt.

The graph is shown below:

Plotting: The output contractWithGas.txt and avg_gasused.txt was read and changed into a data frame using pandas. Date was changed from object to datetime data type and value as float. Then grouped by date. The graph was plotted using the plot method of matplotlib library. From the graph we can visualize the change of gas compared to the top contracts.