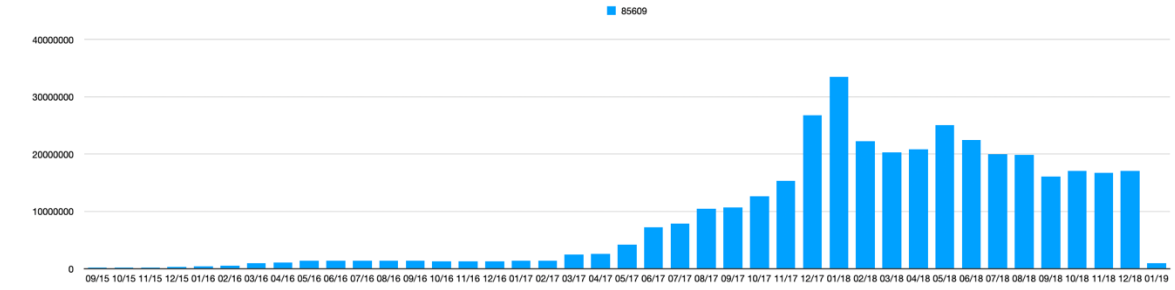
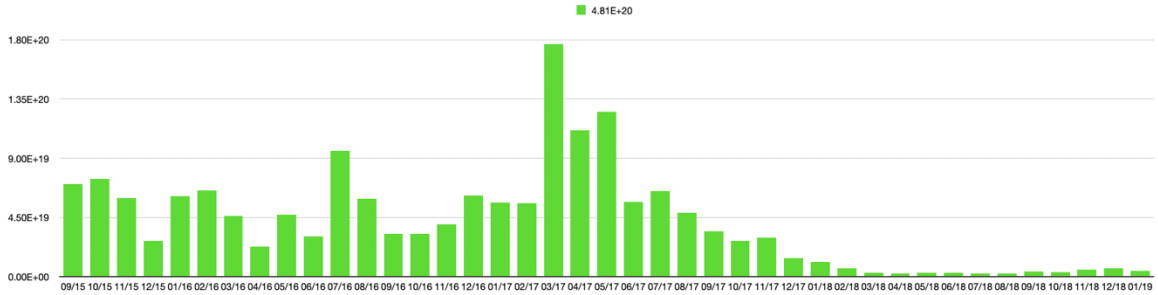
Part A. Time Analysis

I used spark to do this job where I found total monthly transactions and average value of monthly transactions from the transactions dataset. I used RDD called lines to get the dataset from HDFS. I used good_line() function on the RDD to filter the database and get rid of unusable entries. Then I mapped the data using map() function to get month and year from timestamp table and get value of transaction from value table. After that I used reduceByKey() function to add all values and number of values and then I mapped again using map() function and calculated the monthly average value and number of transactions and printed the result.

Number of transactions has been increasing slowly from September 2015. In 2017 the numbers grew rapidly until January 2018 which had the highest number of transactions. The numbers fluctuated throughout 2018 and there was a sudden drop in January 2019.



The average value of monthly transactions has been fluctuating from the beginning until July 2017. Average value has been most high in March 2017. After July 2017 average value has been decreasing.



Part B Top Ten Most Popular Services

Like part a, at first I used good_line() function and filtered transactions RDD which I named t_lines to get rid of malformed lines. Contracts RDD is named c_lines. I also filtered both RDDs to get rid of first rows that consist of field names. I used map() function to map the RDD to get to_addesss(field[6]) and value(field[7]) from the transactions dataset. Then I mapped contracts RDD to get the address which is the first field of the dataset. Then I used the join() function to join contracts with transaction and mapped the result again using map() function. Then I used reducedByKey() function to sum the results. And finally I used takeOrdered() function on results to get top10 values from the entries and then I printed the addresses and the values. Below are the top10 addresses and transaction values.

0x341e790174e3a4d35b65fdc067b6b5634a61caea;8.335689e+24
0xbfc39b6f805a9e40e77291aff27aee3c96915bdd;1e+24
0x52965f9bd9d0f2bbea9b5a9c155a455d0e58fe25;9.257321843822266e+23
0x209c4784ab1e8183cf58ca33cb740efbf3fc18ef;8.4999999939168e+23
0xbfc39b6f805a9e40e77291aff27aee3c96915bdd;7.9999999913194e+23
0xd5c64535f370fe00c5c73b8a42e4943dff4806b7;7.75333e+23
0xbfc39b6f805a9e40e77291aff27aee3c96915bdd;7.037132491261e+23
0xab7c74abc0c4d48d1bdad5dcb26153fc8780f83e;7e+23
0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444;6.726846124823286e+23
0xaa1a6e3e6ef20068f7f8d8c835d2d22fd5116444;6.724243542875985e+23

Part C. Top Ten Most Active Miners

I imported the RDD blocks.csv with .rdd to instantiate lines. Then I mapped the RDD with map() function and got miner addresses(field[9]) and block size(field[12]). Then I used reduceByKey() function and summed the results and finally takeOrdered() function to get top10 miners. Below is list of top10 miners and size of the blocks mined.

('0xea674fdde714fd979de3edf0f56aa9716b898ec8', 17453393724),
('0x829bd824b016326a401d083b33d092293333a830', 12310472526),
('0x5a0b54d5dc17e0aadc383d2db43b0a0d3e029c4c', 8825710065),
('0x52bc44d5378309ee2abf1539bf71de1b7d7be3b5', 8451574409),
('0xb2930b35844a230f00e51431acae96fe543a0347', 6614130661),
('0x2a65aca4d5fc5b5c859090a6c34d164135398226', 3173096011),
('0xf3b9d2c81f2b24b0fa0acaaa865b7d9ced5fc2fb', 1152847020),
('0x4bb96091ee9d802ed039c4d1a5f6216f90f81b01', 1134151226),
('0x1e9939daaad6924ad004c2560e90804164900341', 1080436358),
('0x61c808d82a3ac53231750dadc13c777b59310bd9', 692942577)