**CIS6930 – BlockChain**

**HW -2: Exploring Bitcoin Transactions**

Subham Agrawal | UFID – 79497379

**Github**: <https://github.com/subhambgh/blockchain_spark>

**Dataset:** Original dataset from <https://senseable2015-6.mit.edu/bitcoin/>

**Technology Stack**

Apache Spark v2.3.2 in Scala v2.11.12

Hadoop v3.0

Java v8

on AWS Educate Instance using EMR, S3 Modules

**System Configuration**

**Master Node**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EMR Instance Type | vCPU | ECU | Memory (GiB)\*\* | Instance Storage (GB)\*\* | Instance Count |
| m5.xlarge | 4 | 16 | 16 GiB | 96 GB | 1 |

**Worker Nodes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EMR Instance Type | vCPU | ECU | Memory (GiB)\*\* | Instance Storage (GB)\*\* | Instance Count |
| m5.xlarge | 4 | 16 | 16 GiB | 96 GB | 7 |

**Task Node (spot instance)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| EMR Instance Type | vCPU | ECU | Memory (GiB)\*\* | Instance Storage (GB)\*\* | Instance Count |
| m5.xlarge | 4 | 16 | 16 GiB | 96 GB | 1 |

\*\* data specified above corresponds to available resources before spark and Hadoop installations

\* HDFS was used with default replication factor i.e., 3

**Main Classes**

1. [blockchain\_spark](https://github.com/subhambgh/blockchain_spark)/[src](https://github.com/subhambgh/blockchain_spark/tree/master/src)/[main](https://github.com/subhambgh/blockchain_spark/tree/master/src/main)/[scala](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala)/[com](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala/com)/[blockchain](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala/com/blockchain)/**app**/**Part1\_1.scala** – Used for Part1. Q1 – Q4
2. [blockchain\_spark](https://github.com/subhambgh/blockchain_spark)/[src](https://github.com/subhambgh/blockchain_spark/tree/master/src)/[main](https://github.com/subhambgh/blockchain_spark/tree/master/src/main)/[scala](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala)/[com](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala/com)/[blockchain](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala/com/blockchain)/**app**/**Part1\_2.scala** - Used for Part1. Q5 – Q8
3. [blockchain\_spark](https://github.com/subhambgh/blockchain_spark)/[src](https://github.com/subhambgh/blockchain_spark/tree/master/src)/[main](https://github.com/subhambgh/blockchain_spark/tree/master/src/main)/[java](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/java)/[com](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/java/com)/[blockchain](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/java/com/blockchain)/**app**/**PreProcessinginHDFS.java** – Used to preprocess txin.dat and txout.dat
4. [blockchain\_spark](https://github.com/subhambgh/blockchain_spark)/[src](https://github.com/subhambgh/blockchain_spark/tree/master/src)/[main](https://github.com/subhambgh/blockchain_spark/tree/master/src/main)/[scala](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala)/[com](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala/com)/[blockchain](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala/com/blockchain)/**app**/**PreProcForPart2.scala** – Draws Graph and calculates the connected component analysis
5. [blockchain\_spark](https://github.com/subhambgh/blockchain_spark)/[src](https://github.com/subhambgh/blockchain_spark/tree/master/src)/[main](https://github.com/subhambgh/blockchain_spark/tree/master/src/main)/[scala](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala)/[com](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala/com)/[blockchain](https://github.com/subhambgh/blockchain_spark/tree/master/src/main/scala/com/blockchain)/**app**/**Part2.scala** – Used for Part2

**Compiled Using**: sbt assembly

Note: Can also be verified on a small dataset using the test configurations (files locations can be specified in resources/config-Local.properties file)

Also, note that all the logic for part1 and part2 are specified as comments in the main classes above.

**Idea behind Part II (the most interesting one)**

Let say, we have the following data

txin.dat txout.dat

|  |  |
| --- | --- |
| addID | txID |
| A | tx1 |
| B | tx1 |
| B | tx2 |
| C | tx2 |
| D | tx2 |

|  |  |
| --- | --- |
| addID | txID |
| Z | tx1 |

**Step1: Joint Control**

1. Firstly, draw a vertex for each address ID in addresses.dat.
2. Add an edge between address if they belong to the same transaction and store the tx information.

**Step2: Serial Control**

1. Find all the single o/p transactions
2. Now, for a single o/p transaction with txID say tx1, find the edge with same transaction ID as tx1
3. Connect the o/p address with one of the vertices belonging to that edge as shown below

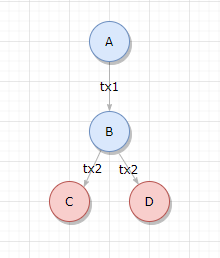
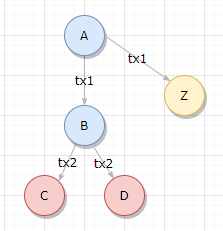
****

Fig 1.Joint Control Fig 2.Serial Control

**Step 3: Calculate the connected component analysis on the formed graph using BFS/DFS**

All the connected component will belong to a single user.

**Part 1: Transactions analysis**

1. **What is the number of transactions and addresses in the dataset**?

Number of transactions: 298325122 and address: 370269747

1. **What is the Bitcoin address that is holding the greatest amount of bitcoins? How much is that exactly? Note that the address here must be a valid Bitcoin address string. To answer this, you need to calculate the balance of each address. The balance here is the total amount of bitcoins in the UTXOs of an address.**

Address ID: 211452559 has greatest amount of bitcoin: 16983763426833

1. **What is the average balance per address?**

avg balance/address: 4551543.822266252

1. **What is the average number of input and output transactions per address? What is the average number of transactions per address (including both inputs and outputs)? An output transaction of an address is the transaction that is originated from that address. Likewise, an input transaction of an address is the transaction that sends bitcoins to that address.**

Avg i/p transactions per address: 1.627057559741709

& Avg o/p transactions/address: 2.170451681541241

1. **What is the transaction that has the greatest number of inputs? How many inputs exactly? Show the hash of that transaction. If there are multiple transactions that have the same greatest number of inputs, show all of them.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **txID** | **blockID** | **n\_ip** | **n\_op** | **hash** |
| 78373103 | 367899 | 20000 | 1 | f2e197a6d8d088b13afd0f99d4027da36a9413b9f3d7730ba5278132ebc950a7 |
| 78371005 | 367897 | 20000 | 1 | 8dabbf51f78c1e7286866af1de403118c5ddbe57ca93b54859245916d2bf1063 |
| 78377474 | 367906 | 20000 | 1 | dd6067e71c04cb62f8e5aa52ecc99b01ffcd551a52727d046a2fabb14eb39b4d |
| 78376719 | 367904 | 20000 | 1 | 740ac533882221099e7202bbdafbb99ec589c6e74fd2fe7ca1274b46ea4f0a96 |
| 78357607 | 367877 | 20000 | 1 | 52539a56b1eb890504b775171923430f0355eb836a57134ba598170a2f8980c1 |
| 78382093 | 367911 | 20000 | 1 | 5f4d2593c859833db2e2d25c672a46e98f7f8564b991af9642a8b37e88af62bc |
| 78361636 | 367885 | 20000 | 1 | 30b3b19b4d14fae79b5d55516e93f7399e7eccd87403b8dc048ea4f49130595a |
| 78364751 | 367891 | 20000 | 1 | c9fe64681c9a12795586a3ae7c5e94b585032f67847c7f9c42e1b979a1e2959b |
| 78361909 | 367886 | 20000 | 1 | cf1032c2213e6faea04f1813aa6890e7f588bb378cb98e7425aec83c11d4457c |

1. **What is the average transaction value? Transaction value is the sum of all outputs’ value.**

average transaction value :1530115769.389244664416830440448123

1. **How many coin base transactions are there in the dataset?**

Number of coinbase transactions: 508241

1. **What is the average number of transactions per block?**

Average number of transactions per block: 586.976

**Part 2: Address de-anonymization**

**Joint control**: assume that all input addresses of a transaction are controlled by the same user.

**Serial control**: assume that the output address of a transaction with only a single output is controlled by the same user owning the input addresses.

1. **How many users are there in the dataset?**

numUsers: 155543894

1. **Answer questions 2, 3, and 4 in part 1 by replacing "address" with "user". Note that each user is identified by the addresses that are owned by him/her. Thus, in answering question 2 (i.e., the user who is holding the greatest amount of bitcoins), you need to list all the user’s addresses.**

2. User 0 has greatest amount of bitcoin= 332101450015043

3. avg balance/user: 1.0833194001679288E7

4. Avg i/p transactions/user : 1.9144925868964036

& Avg o/p transactions/address : 3.4236879912495954

1. **Give the hash of the transaction sending the greatest number of bitcoins to the user who is holding the greatest balance.**