Madeline Tobolewski

DATA470 - Capstone Project Project Update #1

Ethereum vs. Ethereum Classic

Project Background

The intention of this project is to analyze the elements of popular cryptocurrency blockchains to better understand their network structures and operations. The primary focus of this study is the Ethereum blockchain network, while acknowledging the Bitcoin network consistently as a comparable blockchain application in which to compare the network structures. Ethereum and Bitcoin both use the Nakamoto consensus to regulate their blockchains' transaction serialization and are similar architecturally. Primary differences in the two cryptocurrencies include their APIs, abstractions, and wire protocol (Gencer, et al. 2018).

Blockchain technologies have been quickly extending their presence among a variety of industries, not the least of which includes the financial sector and cryptocurrency markets. This rapid expansion and implementation of blockchain has proven its potential to be altered and adopted by any company or organization seeking to more securely transfer, verify, and log data. In the writing, Blockchain Technology: Beyond Bitcoin, Crosby et al. describes blockchain as,

Essentially, a distributed database of records, or public ledger of all transactions or digital events that have been executed and shared among participating parties. Each transaction in the public ledger is verified by consensus of a majority of the participants in the system. Once entered, information can never be erased. The blockchain contains a certain and verifiable record of every single transaction ever made. (Crosby, et al. 2016)

The creation of the cryptocurrency, Bitcoin, in 2009 and the creation of blockchain are synonymous, and in recent years the success of Bitcoin has inspired other companies and individuals to jump onto the blockchain bandwagon. Most of these other projects aim to build onto the foundation of the Bitcoin network itself to leverage the high value placed in the system and the vast amount of computation that goes into the consensus mechanism (Wood n.d.).

The Ethereum, a cryptocurrency introduced in 2015, is a decentralized virtual machine, which can execute programs – called contracts – written in a Turing-complete bytecode language, called Ethereum Virtual Machine (EVM) (Wood n.d.) (Bartoletti, Carta, et al. 2019). Every contract has a permanent storage where to keep data, and a set of functions which can be invoked either by users or by other contracts. Users and contracts can own a cryptocurrency (called ether, or ETH), and send/receive ether to/from users or other contracts (Bartoletti, Carta, et al. 2019). The Ethereum blockchain is similar in a variety of ways to the Bitcoin blockchain, although there are significant differences. The main difference between Ethereum and Bitcoin, in regard to the blockchain architecture, is that Ethereum blocks contain a copy of both the transaction list and the most recent state, while Bitcoin only blocks only contain a copy of the transaction list. Aside from that, two other values, the block number and the difficulty, are also stored in the block (Chinchilla 2019).

The first exploration into the concept of smart contracts was done in the late 1990s, and it showed clear potential to support the algorithmic enforcement of agreements. However, until very recently, in 2015, no specific system was proposed to create or implement such a system. Smart contracts are distributed programs that control the flow of the digital currency, Ether, and allow for expressing a broad spectrum of financial applications. No new concept goes unaffected by negative implications, as the significant semantic complexity increases the risk of programming errors (Grishchenko, Maffei and Schneidewind 2018).

In the research piece, Dissecting Ponzi schemes on Ethereum: Identification, analysis, and impact, by Bartoletti, Carta et al. explain the process in which transactions are initiated and executed, as well as how contracts are tied into the process. The process is broken down as follows:

Users can send transactions to the Ethereum network in order to: (i) create new contracts; (ii) invoke a function of a contract; (iii) transfer ether to contracts or to other users. All the transactions sent by users, called external transactions, are recorded on a public, append-only data structure — the blockchain. Upon receiving an external transaction, a contract can fire some internal transactions, which are not explicitly recorded on the blockchain, but still have effects on the balance of users and of other contracts. (Bartoletti, Carta, et al. 2019)

An understanding of the contract execution code and the heightened possibility for security risks is paramount to analyzing the Ethereum blockchain data. However, the analyzation of smart contracts is difficult for a few reasons. The first being that Ethereum smart contracts are developed in Solidity, a language which features transaction-oriented mechanisms and a number of non-standard semantic behaviors. Additionally, smart contracts are uploaded onto a blockchain in the form of

The Data

We are going to use Google BigQuery's dataset called "ethereum_blockchain" in their big-query-public-datasets. This has 7 tables associated with the ethereum_blockchain. These tables include: blocks, contracts, logs, token_transfers, tokens, traces, transactions. Due to my forgetfulness of how to use Google Cloud Platform, we have not been able to explore the data as much as I would have liked. But this data has a lot of values that will be easily used to analyze and come to conclusions. We plan to obtain the data usign GCP. I have been able to run a few queries on the tables and there is a lot of data here, this will make for accurate conclusions. Since there are 7 different tables, we will attempt to join as many as we can. Though I do not think there is an identifying number for each transaction in each table, we will try to combine as many as possible. Also, most of the tables have a few columns that are hash values for the transactions resulting in character strings that take a long time to query. By remvoing uneccessary columns, we hope to make the data clearner and more understandable for us.

The BigQuery dataset is loaded every day. I do not believe that it is real time, since there is no data for 9/16/2019 as of 9:00am. I will look more into when the data is uploaded to GCP.

Goals of Analysis

1. We can look at price and value trends of an Ether (the monetary value associated with Ethereum) and see if there are weekdays or times of the year where the Ether has historically been worth more? Has it been on a constant increase? Or is there a patters?
2. We can look at the transaction counts for certain blocks on the blockchain. Are some used more during the day / at night? Are they more popular on the weekend or on week days?
3. Are there any stand-out transactions? For reasons such as: volume, gas amount, token amount, etc.

Tools we plan to use

We plan to use a lot of graphs and visualizations. This data does not make sense if you were to just stare at the tables, because you can only see a snap shot of what the data is trying to tell you. If we convert the data to easy-to-read visualizations, we can easily draw priliminary conclusions about the data and display those findings to the class.

We also plan on implementing some use of machine learning, whether it is to predict the highest trafficed times of Ethereum transaction, or something else. We do not know yet.

Products

We will produce graphs and other easy to read visualiations. Since most of the data we have uses keys and unknown characters to someone with no knowledge of the data set, we will convert all ambigous column names to "English" in order to allow for conclusions to be drawn.

Project Methodology

This project will use Google's Colaboratory to run Python3 Jupyter Notebooks. To utilize the publicly available datasets from Google BigQuery, I will use a sequence of SQL queries. The goal of this is to query data from the crypto_ethereum and crypto_ethereum_classic datasets to compare Ethereum with Ethereum Classic.

I also plan on using plotly to plot my findings and add visuals to my project. Given the similarities in the structure of the datasets, it should be easy to compare the two cryptocurrencies together. However, because the datasets are so large most of the analysis will keep each of the two datasets separate.

In [ ]:
```
import numpy as np
import pandas as pd
import os
from google.cloud import bigquery
!pip install plotly
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

In [ ]:
```
import plotly
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.graph_objs as go
plotly.offline.init_notebook_mode(connected=True)
```

In [ ]:
```
project_id = 'practical-lodge-253418'
client = bigquery.Client(project=project_id)
```

In [ ]:
```
blocks_ETC = client.query('''
WITH mined_block AS (
  SELECT miner, DATE(timestamp)
  FROM `bigquery-public-data.crypto_ethereum_classic.blocks`
  WHERE DATE(timestamp) > DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH)
  ORDER BY miner ASC)
SELECT miner, COUNT(miner) AS total_block_reward
FROM mined_block
GROUP BY miner
ORDER BY total_block_reward ASC
''')
```

In [ ]:
```
blocks_ETC = client.query('''
WITH mined_block AS (
  SELECT miner, DATE(timestamp)
  FROM `bigquery-public-data.crypto_ethereum_classic.blocks`
  WHERE DATE(timestamp) > DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH)
  ORDER BY miner ASC)
SELECT miner, COUNT(miner) AS total_block_reward
FROM mined_block
GROUP BY miner
ORDER BY total_block_reward ASC
''')
```

In [ ]:
```
rows = list(iterator)
# Transform the rows into a nice pandas dataframe
top_miners = pd.DataFrame(data=[list(x.values()) for x in rows], columns=list(rows
[0].keys()))
# Look at the first 10 headlines
top_miners.head(10)
```

In [ ]:
```
```

In [ ]:
```
blocks_ETH = client.query('''
  SELECT DATE(timestamp) as date, number, miner, difficulty, total_difficulty, siz
e, gas_limit, gas_used, transaction_count
  FROM `bigquery-public-data.crypto_ethereum.blocks`
  WHERE DATE(timestamp) >= "2016-01-01" AND DATE(timestamp) < "2019-10-31" ''').to
_dataframe()
```

In [ ]:

In [ ]: