NYDIG Data Engineering Candidate Project
Ian Fitzgerald
June 6, 2021

Overview

The project was given with two primary objectives:

1) Design a data pipeline to parse raw Bitcoin blockchain data from JSON form and convert it to flat files
2) Create metrics that analyze on-chain activity

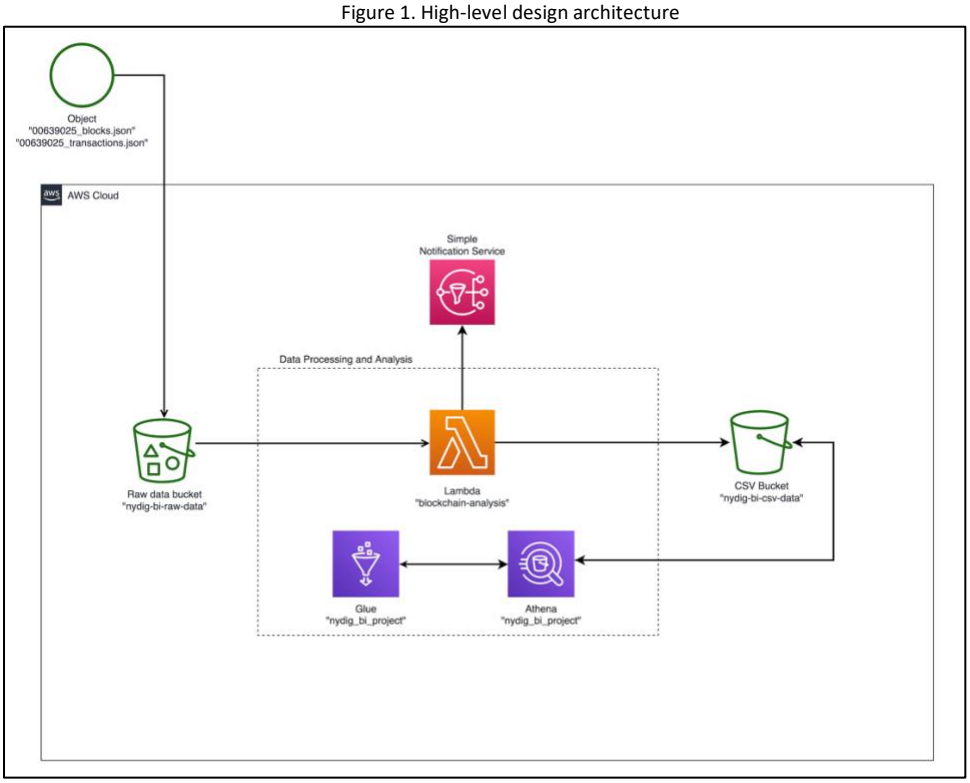
I created a simple and lightweight data pipeline that fulfills both objectives. I will discuss the high level design and the reasoning behind various design decisions.

Design

The design approach for Object 1 can take many different forms, and can easily become quite complicated depending on performance variables. Some factors that were taken into account are: speed, cost, ease-of-use, and ability to scale.

First, the "raw" JSON files are dropped into the S3 bucket named "nydig-bi-raw-data." This triggers the Lambda function "blockchain-analysis" which parses the raw data and converts it to flat pipe-delimited CSV files. It then uploads these files to an output bucket named "nydig-bi-csv-data". Glue and Athena are configured to query the files in this output bucket.
Lastly, the Lambda functions performs a number of calculations on the data to analyze the on-chain activity. If the activity meets certain thresholds, the Lambda triggers a SNS notification.

Figure 1. High-level design architecture



Below we discuss some of the major factors that were taken into consideration for the design.

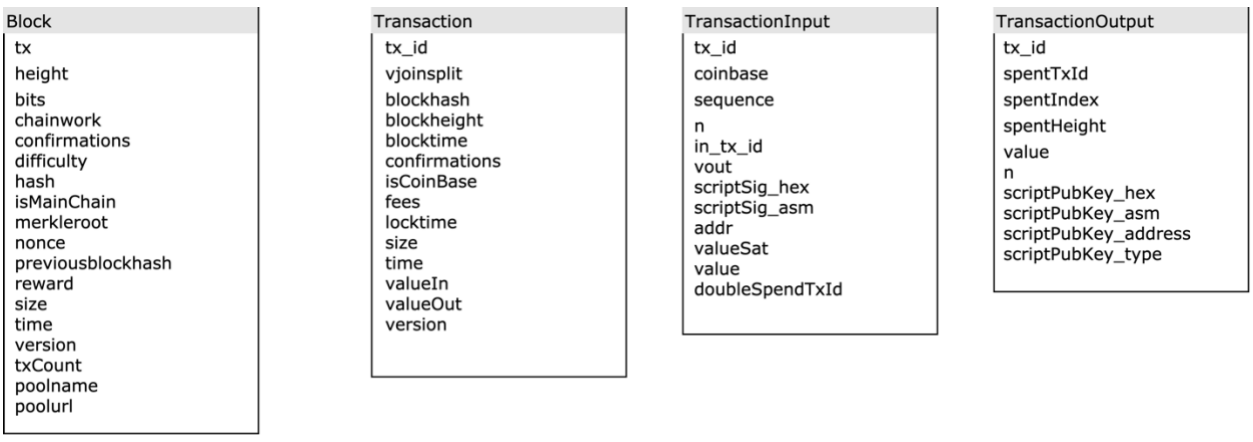
Speed: the project explicitly states to utilize Python code for the data transformation. Therefore absolute speed did not play a large factor in this design approach. If, however, speed were to become a major factor in the future, I would consider abandoning Python and Lambda and instead switch to using Scala or C++ and an always-on EC2.

Cost: using Lambda functions saves on ETL cost and does not require optimizing cloud compute resources. Similar to Lambda, Athena is an on-demand solution and is cost-effective for this small-scale project. If the requirements were to change such that analysts would consistently run large queries on the entire blockchain's history, this would potentially make Athena a less attractive option for cost-saving purposes.

Ease-of-use: for non-technical users, utilizing Athena allows users to explore the data in a familiar relational database environment. Additionally, Athena makes it easy to connect to front-end data visualization tools. We can extend this project to include a Superset front-end for building data dashboards. Further, for the purposes of this exercise, other AWS options such as Redshift were not seriously considered due to the significant amount of time required to prepare and set up the cluster.

Ability to scale: the current Bitcoin blockchain size is over 300GB and, at current pace, is growing by one gigabyte every few days[1]. A single block on the Bitcoin blockchain will remain capped at 1 megabyte unless the Bitcoin network undergoes a significant technical change. For these two reasons, processing a single block with Lambda is a great solution, as Lambda is able to finish parsing a single block's JSON file within a few seconds. Further, Athena has the ability to analyze the aggregate blockchain data (>300GB).

Table definitions:



Metrics

The metrics I created for object 2 are relatively basic. When a transaction file is processed by the Lambda, the metrics are emailed via SNS to anyone who is subscribed to the topic.
The metrics are:

- The total sum of transactions for the current block.
- The total sum of transaction inputs for the current block.
- The total sum of transaction outputs for the current block.

The total value of the Bitcoin in.
The total value of the Bitcoin out.
The total sum of fees collected.

These metrics are useful because they are directly correlated to the volume on the Bitcoin network. For example, we would be able to track the amount of Bitcoin being moved around for a particular block, and the amount of transactions. If there is a low amount of transactions but a high amount of Bitcoin being moved, we might deduce that "whales" are moving a lot of Bitcoin around.

We might also be able to see that the transaction fees are influenced by total transaction volume and transaction size. This would be useful for determining when to move Bitcoin on the network in order to minimize your transaction fees.

Future Improvements

The data pipeline and metrics created for this exercise are extremely useful, however we could create a more sophisticated pipeline with additional features. Some key improvements include:

1) Creating a mechanism to alert when a new block file hasn't landed on the S3 bucket within an expected time frame. We know that the current block time is around 10 minutes. Therefore, if a block file doesn't land on the S3 bucket for 30 minutes, we would send out an alert to the data team.

2) PagerDuty Integration: it would be beneficial for the data team to integrate PagerDuty, as different alerts could generate different levels of alarm. For example, a small risk alert would notify only one engineer, but if an alert is high risk, PagerDuty could immediately escalate it to the leadership team. Additionally, PagerDuty allows for the data team to create a schedule of on-call engineers.

3) Metrics:
    a) More sophisticated time-series statistics. The current metrics ignore time. It would be beneficial to create metrics that take into account time. The new metrics would be able to produce alerts explicitly based on time-series data, such as the number of transactions per hour.
    b) Additional metrics on the aggregate data. The existing metrics analyze the data for each individual block. I would like to introduce new metrics that analyze the entire blockchain for patterns such as:
        a. Common wallet addresses
        b. Recurring periods of high volume/low volume
        c. Large transactions that are outliers to the historic average

4) Data visualization
   I would love to add a Superset or Tableau front-end component for analysts to create their own charts/graphs and visualizations.
5) DevOps release pipeline
   To further productionize this data pipeline, I would create a release pipeline to allow for CI/CD.

References

[1] https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/