NFT analytics application with value prediction

Eshaan Bhardwaj[1] ,Gyanesh Samanta[1] , Dr. C Fancy[2]

SRM IST, Kattankulathur, Chennai, India

eb7127@srmist.edu.in, gs9064@srmist.edu.in, fancyc@srmist.edu.in

Abstract: The popularity and recognition of blockchain technology along with dealing with crypto assets has grown multifariously in recent years. A direct consequence of this is also seen in the NFT marketplace mushrooming as witnessed in recent years. The aim of this paper is to implement ETL pipelines to transform data on the EVM nodes into a centralised database for analytical modelling. This research involves perform analytical modeling on NFT trades to understand and recommend potential uprising high-value NFTs to buyers. This paper encompasses concepts and procedures of extracting data from EVM nodes, transformation of raw data by identifying patterns in hex values, visualising data using SparkSQL, analytical modeling based on statistics and integration with a cross-platform flutter application. As the end product, we aim to deliver a customized NFT marketplace packaged as a mobile app that allows analysis of various trends that can be filtered by category, creator, value, etc.

Keywords— Web3, Data engineering, Data analytics, Blockchain, Ethereum, ETL, Dune

I. Introduction

Data is of much importance in the current day and age as it enables one to understand performance, visualize metrics and identify key parameters driving businesses. Data alone however is not sufficient. Fundamentally, data is discrete units of raw material that carries no meaning when viewed individually. To transform data into collective groups such that it carries logical meaning, one needs to skillfully perform multiple operations onto it and draw corresponding inferences. This process of classification, filtering, and grouping data resulting in discovery of meaningful patterns constitutes data analytics. Over time, numerous technologies and tools have emerged (such as Tableau, Splunk, PowerBI, Apache Spark, etc) that help data engineers and data scientists perform data analytics to drive business growth. This data is conventionally taken from a central database where all the information of the particular business is stored, and then filtered for processing. This paper however talks about performing data analytics for data on the blockchain which is stored in the form of absolute-distributed nodes. This paper not only elaborates on the inferences drawn from blockchain data and dashboards produced, but also outlines the necessary Extract-Transform-Load (ETL) steps that were followed to filter, classify, and finally visualize the scattered data. The visualization tool used for this research was Dune Analytics which uses spark sql as they query language, and the end goal achieved was to perform analytics on Non Fungible Tokens(NFTs) on the Ethereum chain.

II. Terminologies, Technologies and algorithms used

A. *Dune Analytics*

Dune Analytics is a community-owned tool for blockchain analytics. It allows users to examine NFT collections, blockchain data, DeFi protocol, and much more. It is a web based dashboarding-platform that allows users to search through public blockchain data and aggregate this data into informative dashboards. The data transformed is stored in a SparkSQL format. Similar functionalities are provided by other softwares like Chainanalysis, MythX and Coinpath too, but we choose Dune due to it's integration with 32+ chains. This research harnesses the power and utilities of Dune Analytics to make the dashboards on the transformed data.

B. *Non Fungible Token(NFT)*

Non-Fungible Token is essentially a unique digital asset that cannot b, substituted, copied, or subdivided and is stored on the blockchain. NFTs are trustworthy, easily transferable, and enforce strict ownership rights over assets. They are not interchangable and each NFT represents unique assets owned by a specific person.[1] Each NFT contains distinguishable information making them distinct and easily verifiable. NFTs employ blockchain technology under the hood.[1] This research focuses on analyzing, visualizing, and inferring only NFT specific data from the ethereum blockchain. We perform analytical modelling and show trends and deviations on currently existing NFTs, NFT collections, creators, and owners.

C. *Ethereum Virtual Machine (EVM)*

Ethereum Virtual Machines constitutes of numerous nodes that contribute to carrying out blockchain transactions. EVM reads and executes bytecode. It process blocks sequentially and updates it's state after running each transaction.[2] This research is performed for data of the chains that run on the ethereum virtual machine, like Ethereum, Polygon, Binance, Avalanche, etcetera.

D. *Opensea*

Opensea is the world's first and largest marketplace for user-owned digital assets with support for multiple blockchians and wide range of and competitive prices for emerging digital item classes. It is a major marketplace for

buying, selling, and exchanging NFTs. It was founded in2018 by Alex Atallah and Devin Finzer who took inspiration from CryptoKitties's success in the market. Opensea not only allows you to buy, sell, and trade NFTs, but also enables you to see it's entire trading history, transfer of ownership over time, and also classifies NFTs on the basis of popularity, sales, owner, genre, etc.[3]

E. ETL

ETL stands for Extract-Transform-Load. ETL is a three phase data integration process that is widely followed and used for combining data from multiple sources and loading it to a single, consistent data container.[4] This research has followed ETL steps throughout. We also setup ETL pipelines to streamline the process of updation of realtime data and even used tools like Grafana and Prometheus to monitor the data and pipeline created. It gives user stats like latency, health of pipeline alerts, etcerta.

III. MODELING APPROACH

The high level overview of the modeling approach of the research is as follows - we targeted the EVM nodes for data, extracted and transformed it using a middleware, identified NFTs from the filtered data, created downstream databases for the same, and finally queried the downstream databases to visualize it onto a dashboard. The diagrammatic representation and intricacies of each step are detailed in the upcoming sections.

A. Relational database management service

The first step of our research naturally involves fetching the EVM chain data. The chain data is openly available to the public for free use. However to get the data in a systematic, ordered format we used a middleware called *Web3.py*. Web3.py is a python library for interacting with ethereum. It enables developers to read block data, interact with smart contracts, and much more.[5] We used Web3.py to convert the raw data which is bytecode into hex data groups needed for this research - Blocks, Logs, and Transactions. A pictorial representation is shown in Figure 3.1.1

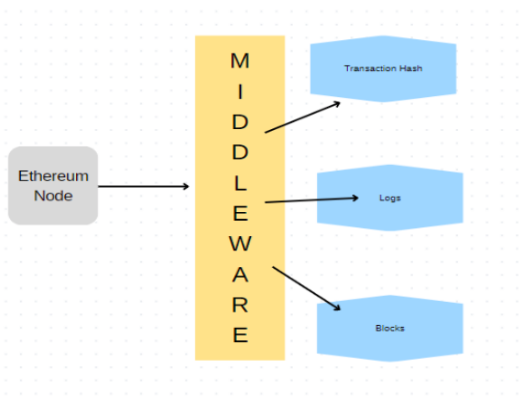


Figure 3.1.1

Blocks represent information of the specific block on the chain. Logs represent the history of transactions on the block and transfer of ownerships over time, and transactions refer to function calls (e.g. minting, transfers,smart contract transactions). We used Amazon Relational Database Service (RDS) to store this grouped data. The RDS we used implemented the PostgreSQL engine of the *db.r6g.xlarge* class. This RDS contains not only Transactions, logs, and block data, but also the raw data of the EVM chain in the form of relational databases.. Some of these databases are shown in the figure 3.1.2 with the Blocks, Logs and transactions of ethereum chain being highlighted.

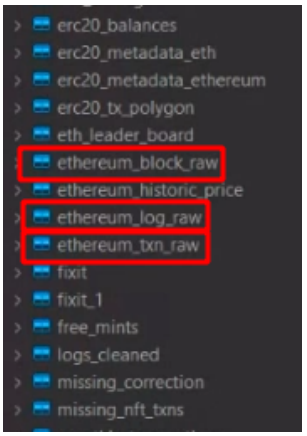


Figure 3.1.2

This data can be queried and retrieved using PostgreSQL. For example, if we execute the following query -

*SELECT * FROM ethereum_block_raw ebr limit 10;*

This lists 10 records from the Blocks table as shown in figure 3.1.3

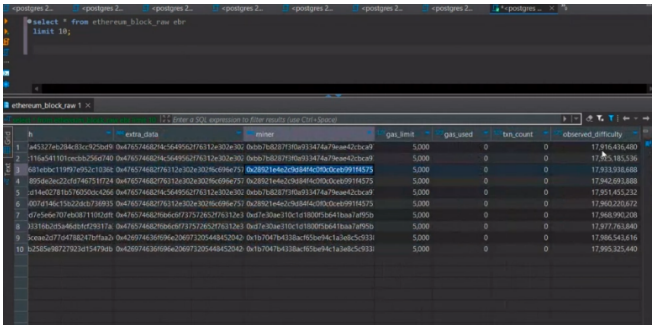


Figure 3.1.3

Similarly, data of Logs and Transactions can be extracted as well.

B. Downstream databases

Using Web3.py middleware we obtained the blocks, logs and transactions data and store it on AmazonRDS for querying. However, this huge amount of raw data comprises data of numerous tokens like ENS names, ERC1155, ERC20, ERC721 etcetera while this research is solely focused on NFT transactions. Hence, querying AmazonRDS repeatedly for only NFT transactions would be very slow since it makes comparisons for each entry for a huge number of records. Therefore, we created downstream databases. Downstream databases contain most frequently accessed/queried data.[6] Downstream databases can be made for individual tokens like one for ERC20 tokens, one for ENS names, one for ERC721 tokens, etc. For this

research we made one for NFT transactions as well. The diagram is shown in Figure 3.2.1

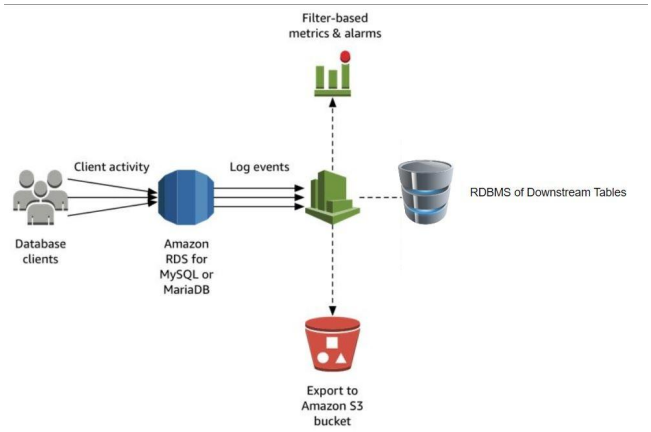


Figure 3.2.1

Overall, downstream databases are created for faster access of data that is needed frequently, and a better understanding of the parameters required to be studied. This downstream table is created by running SQL queries on the raw data and partitioning it. A part of the query executed to make a NFT transactions downstream is shown in Figure 3.2.2

```
CREATE TABLE public.nft_txn (
  block_height int8 NULL,
  log_offset int4 NULL,
  block_signed_at int8 NULL,
  tx_hash text NULL,
  tx_offset int4 NULL,
  collection_address text NULL,
  sender text NULL,
  receiver text NULL,
  token_id text NULL,
  initiator text NULL,
  value numeric NULL,
  price text NULL
)PARTITION BY RANGE (block_signed_at);

CREATE TABLE nft_txn_p2019 PARTITION OF nft_txn
FOR VALUES FROM (1546300800) TO (1577836800);
CREATE TABLE nft_txn_p2020 PARTITION OF nft_txn
FOR VALUES FROM (1577836800) TO (1609459200);
CREATE TABLE nft_txn_p2021 PARTITION OF nft_txn
FOR VALUES FROM (1609459200) TO (1640995200);
CREATE TABLE nft_txn_p2022 PARTITION OF nft_txn
FOR VALUES FROM (1640995200) TO (1672531200);
1653936710

insert into nft_txn
select * from nft_transactions ;
```

Figure 3.2.2

As seen from the query above, we first create a new table(called *nft_txn*) with the parameters we need to study as column titles, create partitions, and finally insert data into that table.

C. *ETL Pipelines*

The raw data was first extracted from the EVM chain using web3.py and split up into blocks, logs and transactions. Transform step corresponds to the creation of downstream databases and insertion of selective raw data into them. Finally, the transformed data from downstream databases is loaded onto the Dune dashboard for visualization. We went for ETL instead of ELT as we're converting the bytecode stored in EVM Nodes into hex values through pattern matching, which in turn would result a slow ELT process as the data won't be readily available.[7]

This alone is not sufficient though because the raw data from web3.py and downstream databases created by the above approach are static. They need to be updated with realtime data. For this purpose we setup and deployed ETL pipelines to keep the raw data and downstream databases up to date with live transactions happening in the real world. Creating this pipeline eliminates the need to run the insert query repeatedly and streamlines the updation process.

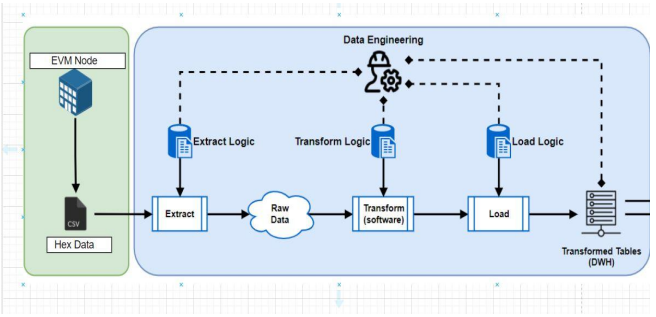


Figure 3.3.1

In addition to this, some more tool like Grafana and Prometheus were also used to help monitor the data and pipeline created. It gives user stats like latency, health of pipeline alerts, etcerta.

D. *Dune dashboard*

After having the raw data from EVM chain extracted, transformed we ultimately require it to be statistically visualized it in a systematic way. The analytics tool used for this research was Dune Analytics.

The downstream databases we created earlier had all the data corresponding to NFTs - block height, log offset, block signing time, transaction hash, transaction offset, collection addres, sender, receiver, token id, initiator, value, price, crypto used, listed price, mint price, NFT type, quantity, and the primary interacting contract.. We then harnessed dune to visualize all that data in the form of pie charts bar graphs, histograms,etc. Dune has a built-in query editor (which works with SparkSQL) that enables us to fetch and therefore visualize the data. For example, sales volume was represented pictorially in Figure 3.4.1 and the SQL query for the same is depicted in Figure 3.4.2

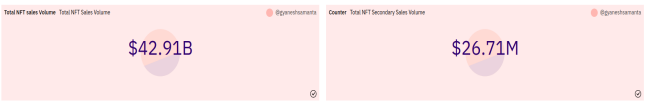


Figure 3.4.1

```
WITH to_exclude AS (
    SELECT COUNT(*) AS exclude_txs
    FROM nft.trades
    WHERE tx_hash IN (
        SELECT '\x92488a00dfa0746c300c66a716e6cc11ba9c0f9d40d8c58e792cc7fcebf432d0' AS txs
            UNION
        SELECT call_tx_hash AS txs
        FROM opensea."WyvernExchange_call_atomicMatch_"
        WHERE (addrs[4] = '\x5b3256965e7c3cf26e11fcaf296dfc8807c01073'
              OR addrs[11] = '\x5b3256965e7c3cf26e11fcaf296dfc8807c01073')
        AND call_success
        AND call_block_time > '2022-01-21'
        GROUP BY 1
        HAVING count(DISTINCT CASE WHEN addrs[7] = '\x0000000000000000000000000000000000000000'
            THEN '\xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2' ELSE addrs[7]
            END) > 1)
    )

    SELECT ((COUNT(*)-(SELECT exclude_txs FROM to_exclude))/1e6 AS total_txs
    FROM nft.trades
```

Figure 3.4.2

Similarly, visualizations were drawn for NFT ranked by volume, transaction count per platform, NFT volume in dollar, etc. and put together to form a dashboard. The respective visualizations are shown through Figure 3.4.3 to Figure 3.4.6

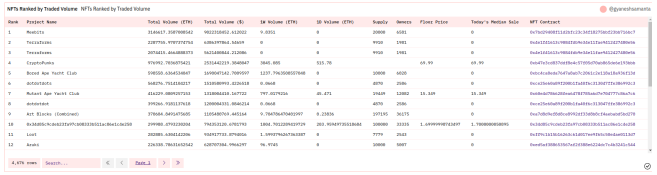


Figure 3.4.1

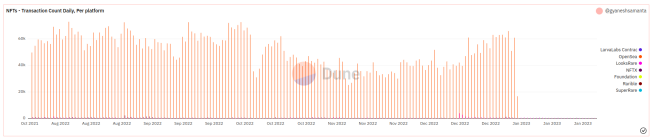


Figure 3.4.1

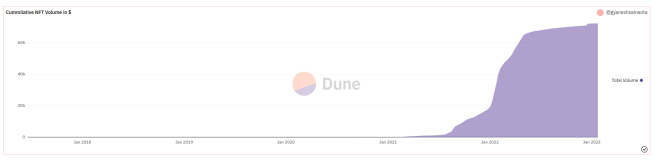


Figure 3.4.1

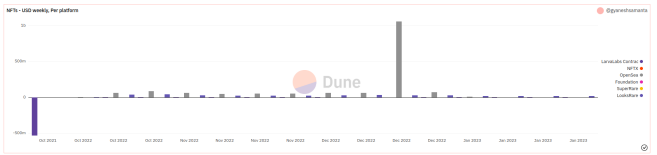
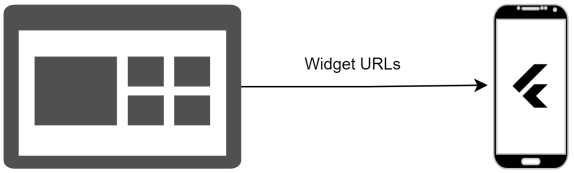


Figure 3.4.1

E. Flutter integration

The final step is to package the live visualizations produced by dune into a mobile app. The visualizations generated by dune are by default available for a web interface. Hence, in order to access the dashboard one needs a url pointing to it. Since the focus of this research was delivering a mobile app, we used Flutter to achieve this. Flutter is a cross-platform UI-toolkit using for making mobile applications. Since it is cross-platform, the apps made using flutter run on both - android as well as iOS devices.[8]

As stated, Dune generates dashboards which can be viewed via a URL. Hence, to make this available on a mobile app, we first generate a unique URLs for each widget on the dashboard, and then send those URLs to the app which are placed in a *WebView*. A Webview widget is a flutter plugin that is used to display web pages on android and iOS devices. Each of these widgets is shown inside a WebView placed on different screens and the desired result is achieved.



IV. CONCLUSION

In this research we extracted raw blockchain data, processed and filtered it, setup ETL pipelines, made a mobile-friendly dashboard displaying analytics, and finally drew logical inferences from it. The resulting product was a mobile app that gave you real time updates and analytics of numerous NFT trends. Main pain points we faced while carrying out the research was setting up streamlined pipelines and modifying and running SQL queries on the abundant raw data to extract and load useful information. Tests were carriedd out everyday with realtime data being updated, and the mobile app was tested on different platforms and different screen sizes to make it more robust across all devices. Finally, a high level diagram of the entire research is shown in Figure 5.1

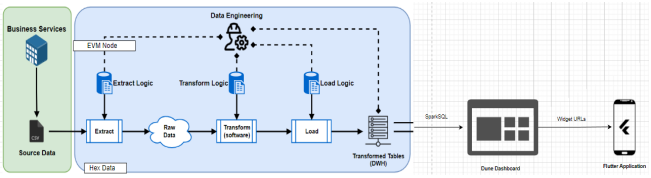


Figure 5.1

V. REFERENCES

[1] Russell, Francis. (2022). NFTs and Value. M/C Journal. 25. 10.5204/mcj.2863

[2] Ma, Fuchen & Ren, Meng & Fu, Ying & Wang, Mingzhe & Li, Huizhong & Song, Houbing & Jiang, Yu. (2021). Security reinforcement for Ethereum virtual machine. Information Processing & Management. 58. 102565. 10.1016/j.ipm.2021.102565.

[3] Efendioğlu, İbrahim. (2022). The rise of the NFT market: The effect of social media interaction and the need for uniqueness on NFT purchase intention.

[4] Singh, Manish. (2022). Extraction Transformation and Loading (ETL) of Data Using ETL Tools. International Journal for Research in Applied Science and Engineering Technology. 10. 4415-4420. 10.22214/ijraset.2022.44939. .

[5] Buldas, Ahto & Draheim, Dirk & Gault, Mike & Saarepera, Märt. (2022). Towards a Foundation of Web3. 10.13140/RG.2.2.18614.98883.

[6] Yulianto, Ardhian. (2020). Extract transform load (ETL) process in distributed database academic data warehouse. APTIKOM Journal on Computer Science and Information Technologies. 4. 61-68. 10.34306/csit.v4i2.92.

[7] Prajapati, Mr & Mary, Mrs. (2022). Study of ETL Process and Its Testing Techniques. International Journal for Research in Applied Science and Engineering Technology. 10. 871-877. 10.22214/ijraset.2022.43931.

[8] Hussain, Hina & Khan, Kamran & Farooqui, Faiza & Ali, Qasim & Siddiqui, Isma. (2021). Comparative Study of Android Native and Flutter App Development.