XBlock-EOS: Extracting and Exploring Blockchain Data From EOSIO

Weilin Zheng[a,b], Zibin Zheng[a,b,*], Hong-Ning Dai[c], Xu Chen[a,b], Peilin Zheng[a,b]

[a]*School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China*
[b]*National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou, China*
[c]*Faculty of Information Technology, Macau University of Science and Technology, Macau*

Abstract

Blockchain-based cryptocurrencies and applications have flourished in blockchain research community. Massive data generated from diverse blockchain systems bring not only huge business values but also technological challenges in data analytics of heterogeneous blockchain data. Different from Bitcoin and Ethereum, EOSIO has richer diversity and a higher volume of blockchain data due to its unique architectural design in resource management, consensus scheme and high throughput. Despite its popularity (e.g., 89,800,000 blocks generated till November 14, 2019 since its launch on June 8, 2018), few studies have been made on data analysis of EOSIO. To fill this gap, we collect and process the up-to-date on-chain data from EOSIO. We name these well-processed EOSIO datasets as XBlock-EOS, which consists of 7 well-processed datasets: 1) Block, Transaction and Action, 2) Internal and External EOS Transfer Action, 3) Contract Information, 4) Contract Invocation, 5) Token Action, 6) Account Creation, 7) Resource Management. It is challenging to process and analyze a high volume of raw EOSIO data and establish the mapping from original raw data to the well-grained datasets since it requires substantial efforts in extracting various types of data as well as sophisticated knowledge on software engineering and data analytics. Meanwhile, we present statistics and exploration on these datasets. Moreover, we also outline the possible research opportunities based on XBlock-EOS.

Keywords: Blockchain, EOSIO, Big Data, Data Acquisition, Data Analysis, Security

1. Introduction

With the growing prosperity of cryptocurrencies like Bitcoin [1], blockchain has attracted extensive attention from both academia and industry in recent years. In particular, blockchain can be used in information management systems so as to reduce the dependence of third parties, to improve the interoperability across different business sectors, and to improve efficiency [2, 3]. Moreover, blockchain can also be used in Internet of Things (IoT) [4, 5, 6], critical infrastructures [7, 8], enterprise operational management [9], anti-fake news [10], vehicle management [11], data management and auditing [12, 13].

Blockchain systems can be roughly categorized into permissionless and permissioned blockchains, corresponding to publicly accessing and limited accessing, respectively [14]. Substantial efforts have been made on the permissionless blockchain systems recently, consequently leading to the proliferation

*This is to indicate the corresponding author.
Email addresses: zhengwlin@mail2.sysu.edu.cn (Weilin Zheng), zhzibin@mail.sysu.edu.cn (Zibin Zheng), hndai@ieee.org (Hong-Ning Dai), chenx397@mail2.sysu.edu.cn (Xu Chen), zhengpl3@mail2.sysu.edu.cn (Peilin Zheng)

of diverse blockchain systems, such as Ethereum [15] and EOSIO [**?**]. In a permissionless blockchain system, each peer interacts with a public ledger, which is traceable, tamper-resistant and censorship-resistant. Compared with the traditional Proof of Work (PoW)-based blockchain systems (such as Bitcoin and Ethereum), which are limited by low throughput, EOSIO attempts to offer high throughput with a novel architectural design and Delegated Proof of Stake (DPoS) consensus.

1.1. Motivation

Blockchain that serves as a middleware in information management systems as well as other systems (e.g., IoT) generates massive blockchain data. Meanwhile, the prosperous development of permissionless blockchain systems has also led to the generation of massive data. For example, the volume of Bitcoin data has reached 268GB on March 19, 2020 according to the statistics of *BlockChair*[1]. Meanwhile, according to *Etherscan*[2], there are more than 16,000,000 smart contracts (including about 230,000 ERC20 token contracts) deployed in Ethereum. Regarding EOSIO, the number of transactions of EOSIO has reached 2.8 billion according to the statistics of *eosflare.io*[3], far exceeding those of Bitcoin and Ethereum. Moreover, the architectural design of EOSIO is significantly different from that of Bitcoin and Ethereum, in aspects of the resource management model and consensus mechanism [16]. EOSIO can essentially provide researchers with more diverse types of data than Bitcoin and Ethereum. However, there are few studies on EOSIO data, thereby motivating the study of this paper.

The massive data on blockchain systems has brought huge business values and great opportunities to researchers due to openness, decentralization, and tamper-resistance [17, 18]. Data analysis of massive blockchain data can extract useful information and identify system (or enterprise) bottlenecks, consequently making right decisions for enterprises. In the past, because of security, privacy, and ownership concerns, the real business trading data is usually not opened to researchers; this closure of business data severely hampers related research efforts. In contrast, the data on permissionless blockchain systems are all publicly available to anyone. Meanwhile, the blockchain data can be accessed almost everywhere through the interconnected peer-to-peer (p2p) network. In addition to the trading (transaction) data, many blockchain systems, like Ethereum and EOSIO, also contain both smart contracts and cryptocurrencies (tokens) data. Big data analysis on massive blockchain data cannot only bring huge business values but also promote the development of blockchain. For example, blockchain data analysis can be used for price speculation detection, transaction fraud detection, and smart contract vulnerability detection, consequently improving the security and supervision of blockchains.

At present, the existing studies mainly focus on the data analysis of Bitcoin and Ethereum [19, 20, 21, 22, 23] while few studies concentrate on EOSIO data analysis. Different from Bitcoin and Ethereum, EOSIO has richer and more diversity of blockchain data mainly due to its unique architecture design in resource management and DPoS consensus scheme. The massive volume of heterogeneous EOSIO data not only brings opportunities but also challenges in data analysis. It is challenging to analyze EOSIO data due to the following difficulties. **(1) Difficulty in data synchronization.** Although the main network

[1] `https://blockchair.com/bitcoin`
[2] `https://etherscan.io/`
[3] `https://eosflare.io/`

(a.k.a. *mainnet*) of EOSIO has been launched online for less than two years, EOSIO has generated massive volume data mainly due to the high transaction throughput (i.e., a block is generated every 0.5 seconds) thanks to its highly-efficient architectural design. It will take a long time for a newly-joined peer to fully download (or synchronize) the entire EOSIO blockchain data. For example, it takes more than one month and over 500GB storage space to fully synchronize only *entire block data* at a newly-joined peer. Furthermore, it will take longer and require more storage space to collect other EOSIO data such as the traces and receipts of the transactions. The high requirements (in computing, networking and storage) for data synchronization hinder the efficient analysis of EOSIO data. **(2) Absence of general data extraction tools for EOSIO.** Although several blockchain websites provide partial (incomplete) EOSIO data, their data extraction tools are generally closed source. Developers cannot design and build their own data extraction tools based on closed source web tools. In addition, these websites generally provide users (even for paid users) with only limited HTTP/HTTPS interfaces to obtain partial EOSIO data. Data acquisition through these websites is slow and incomplete, which seriously impedes the progress of conducting research on EOSIO blockchain. **(3) Absence of comprehensive data exploration tools for EOSIO.** Although there are a number of studies on data analysis of Bitcoin and Ethereum, including contract security analysis [24, 25], resource management analysis [26, 27], there are few studies on EOSIO. As far as we know, only two most recent studies [28, 29] attempted to analyze EOSIO data, while they only analyzed partial EOSIO data (e.g., characterizing the activities in EOSIO [28] and detecting fake-transfer vulnerabilities of smart contracts of EOSIO [29]). To the best of our knowledge, there is no work on comprehensive analysis of entire EOSIO data from various data types. **(4) Difficulties in data extraction and data processing.** EOSIO contains massive heterogeneous data with various types and different data structures (e.g., structural, non-structural data as well as byte code). Moreover, EOSIO has a much higher volume of blockchain data than other representative blockchains (such as Bitcoin and Ethereum). The massive volume of EOSIO data also brings challenges in data processing.

1.2. Contributions

To address the above challenges, we introduce a blockchain data analytics framework namely eXplore Blockchain EOS (XBlock-EOS) to extract and explore EOSIO data. In particular, we collect raw data consisting of 89,800,000 blocks of EOSIO data from June 8th, 2018 (i.e., the launching date of the EOSIO *mainnet*) to November 14th, 2019. XBlock-EOS contains **1,882,112 MB** (≈ 1.88 TB) raw data (after compressing JSON format with the highest compression level in zip). The collected raw data includes three types of blockchain data: *blocks*, *transaction receipts*, and *action traces*. Since it is difficult to analyze the massive raw blockchain data, we process and classify the collected EOSIO raw data into seven datasets: *(1) Block, Transaction and Action*, *(2) Internal and External EOS Transfer Action*, *(3) Contract Information*, *(4) Contract Invocation*, *(5) Token Action*, *(6) Account Creation*, *(7) Resource Management*. After processing EOSIO raw data, we obtain well-processed EOSIO datasets with **203,479 MB** (≈ 198.7 GB, after compressing CSV format with the highest compression level in zip). It is non-trivial to process such a high volume of raw EOSIO data and establish the mapping from

original raw datasets to seven well-grained datasets since it requires substantial efforts in extracting various types of data as well as sophisticated knowledge on software engineering and data analytics. We also conduct statistical analysis on the seven well-processed datasets. In addition, we discuss the emerging applications enabled by XBlock-EOS.

In summary, we highlight the major contributions of this paper as follows:

- To the best of our knowledge, XBlock-EOS is the first to provide the most comprehensive on-chain well-processed EOSIO data as well as data extraction, statistics and exploration functions to analyze EOSIO blockchain datasets. In contrast to prior studies that only provided partial EOSIO data, XBlock-EOS provides both comprehensive EOSIO raw data and well-processed EOSIO datasets. In particular, XBlock-EOS includes blockchain data, smart contract data, cryptocurrency data, account creation data and resource management data. Moreover, XBlock-EOS[4] periodically keeps updating raw datasets as well as processed datasets, all of which have been synchronized with the EOSIO *mainnet*.

- The XBlock-EOS framework also provides necessary statistics and exploration functions to analyze blockchain datasets. Meanwhile, we also design and develop a new plugin, which can collect EOSIO on-chain data much faster than EOS official plugins. Moreover, we present some statistics and observations from the seven datasets. The well-processed datasets can be easily used for future in-depth data exploration and data analysis.

- This paper also outlines the research opportunities brought by XBlock-EOS. In particular, we discuss the applications of XBlock-EOS in aspects of blockchain system analysis, smart contract analysis, and cryptocurrency analysis. Most of these analyses are conductive to blockchain security enhancement. Moreover, the joint analysis of EOSIO data with other blockchain data (such as Ethereum dataset as given in XBlock-ETH [30]) can further advance data analysis of blockchain systems and promote the benign development of blockchain.

The remainder of this paper is organized as follows. Section 2 firstly gives an overview of EOSIO, highlighting its differences from other permissionless blockchain systems. Section 3 then presents raw data acquisition from EOSIO. Section 4 next presents a statistical analysis on seven refined datasets. Section 5 discusses the applications of XBlock-EOS data and Section 6 investigates the related work about XBlock-EOS. Finally, we provide a summary of this paper in Section 7.

2. Background

Figure 1 presents an overview of EOSIO blockchain, which consists of four layers from bottom to top: *peers*, *blockchain*, *smart contract*, and *token*. We next review the basic concepts in each layer of EOSIO.

[4]All the datasets of XBlock-EOS can be downloaded from `http://xblock.pro/eosio/`

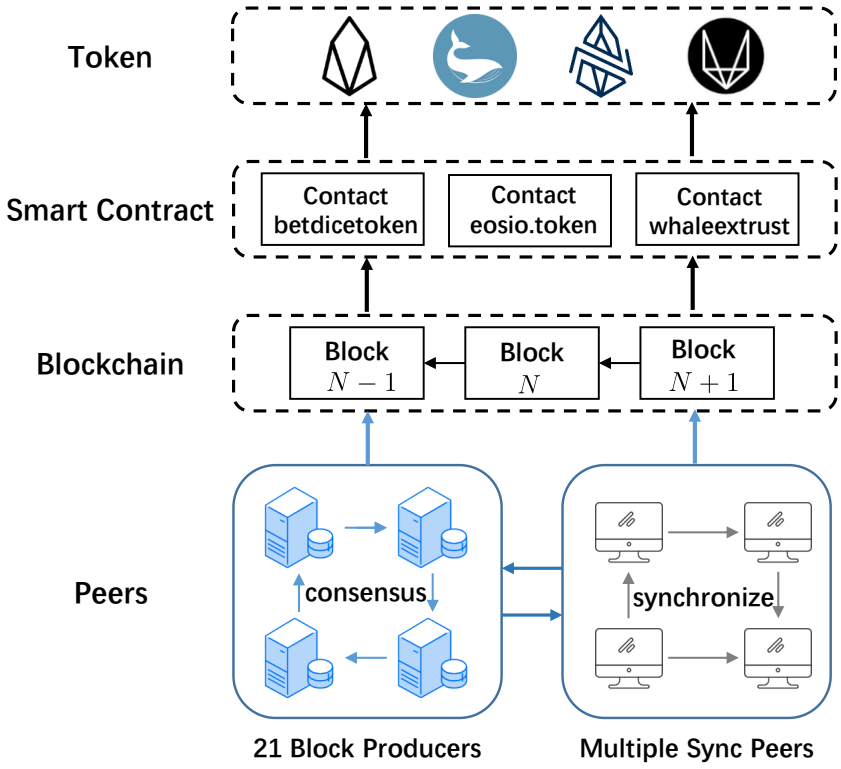


Figure 1: Overview of EOSIO Blockchain

2.1. Peer and blockchain

In short, a blockchain system is essentially a distributed ledger with a chain-like data structure consisting of a number of connected blocks. Transactions are packaged into blocks, each of which is confirmed by the entire network through a consensus protocol in a period of time. Unlike Bitcoin and Ethereum, EOSIO uses Delegated Proof-of-Stake [31] instead of Proof-of-Work or Proof-of-Stake as its consensus mechanism. In DPoS of EOSIO, only 21 block producers (consensus peers) can produce blocks and verify transactions while other sync peers only synchronize the blockchain data. In contrast, any miners in Ethereum and Bitcoin have the opportunity to undertake this work. Once the new block has been confirmed by most peers, i.e., 14 ($= 2/3 \times 21$) consensus peers in the EOSIO network, it is considered as completed. In other words, the EOSIO blockchain enhances the reliability of transaction data by copying calculation and storage across multiple peers.

Due to the integrity of the blockchain data in each permissionless peer, researchers can obtain the entire blockchain data by connecting a new peer to the blockchain network. Blockchain data essentially save all operations performed by real-world users on the blockchain network, thereby containing substantial business value. For example, a transaction is essentially an operation performed by different business parties. Therefore, a transaction may imply a potential interest relationship between any two people or entities. Big data analysis on blockchain can help understand user behavior in real-world economic systems. Moreover, the rapid technical development of blockchain systems has boosted the growing number of blockchain users as well as transactions, leading to a massive growth of blockchain data. In particular, EOSIO that generates a block every 0.5 seconds on average has much higher transaction throughput per second (tps) than Bitcoin and Ethereum. Therefore, the data growth rate of EOSIO is much higher than

that of Bitcoin and Ethereum. The analysis on such massive data is challenging while it also brings huge business values via EOSIO data analysis.

2.2. Smart contract

Smart contract, as a promising technology aimed to reshape the modern industry, was proposed earlier than blockchain [32]. However, it was not well developed until the advent of Ethereum (i.e., the first Turing-complete blockchain smart contract platform), in which smart contract really plays its role of assuring trustworthy transactions between any two parties without a third party's intervention. Blockchain-based smart contracts are essentially computer programs, in which execution states are stored on blockchain. The blockchain transactions represent the deployment or invocation of smart contracts, triggering updates to the state of blockchain. Blockchain guarantees the reliability of smart contracts by replicating the computation of smart contracts at the peers in the network.

Currently, most blockchain systems have enabled smart contracts. For example, the Bitcoin system enables users to run a simple script program during transaction execution, which can be regarded as one of the simplest smart contracts. However, Bitcoin scripts that are not Turing-complete cannot support complex logic. In contrast, the prosperous blockchain systems, such as Ethereum and EOSIO, can well support Turing-complete smart contracts. Smart contracts run in an environment called a blockchain virtual machine. In particular, in Ethereum, smart contracts run in the Ethereum virtual machine (EVM), while EOSIO smart contracts run in the WebAssembly-based EOS virtual machine (EOSVM). In order to solve the *halting problem*, Ethereum introduced a *gas mechanism* to prevent the malicious behavior of a smart contract, such as an infinite loop. Miners in Ethereum use *Gas* as a unit to measure the computation of each operation (instruction) of the smart contract since *Gas* is a scarce resource that is acquired by cryptocurrency purchase.

Unlike many public blockchain systems with gas mechanism, EOSIO solves the *halting problem* by limiting the RAM, CPU, and Network (NET) resources of smart contracts. All these three resources can be obtained only after users mortgage some EOS. Among them, RAM is used to limit the storage of the contract while it is an unrecoverable resource. In other words, the consumption of RAM only depends on how much data is stored. Contracts (or users) can delete the stored data to reclaim RAM, and sell RAM for EOS. CPU and NET limit the computation and the network transmission of the contract, respectively. They are recoverable resources and can be recovered after 24 hours when exhaustion. Users can redeem the mortgaged CPU and NET at any time and receive the corresponding EOS after three days. Consequently, EOSIO is regarded as a free blockchain platform for users, and many DApps in EOSIO are willing to provide users with these resources.

2.3. Tokens and clients

Since Ethereum introduced the standard token protocol (also known as a template) in two smart contracts: *ERC20* and *ERC721*, *Initial Coin Offering (ICO)* [33] has swept the entire cryptocurrency ecosystem. The emergence of ICO has greatly enriched the ecosystem of permissionless blockchain, making blockchain system a more flexible distributed financial system. EOSIO has no exception, and

a standard token token protocol was introduced at the beginning of its launch. EOS, as one of the most representative tokens in EOSIO, has been used for daily operations, such as transferring, leasing, creating an account, buying RAM, staking CPU, and staking NET. In addition, as shown in the top layer of Figure 1, there are three other well-known tokens *WAL*, *NUT*, *VTX*. Everyone can publish a standard token contract in EOSIO to create and issue tokens. After that, any other users and smart contracts can send or receive tokens without a third-party. In Section 4.5, we explore the data of tokens in EOSIO.

EOSIO allows any computers that meet the requirement of the protocol like p2p protocols to join the network. The official EOSIO development team provides an EOSIO client called `Nodeos`. Anyone using `Nodeos` can join the network. `Nodeos` provides a standard JSON-RPC interface through `eosio::http_plugin`[5] and `eosio::chain_plugin`[5] for users to interact with the EOSIO blockchain. Moreover, the official EOSIO development team also develops a tool named `Cleos`, which is a command-line tool that interfaces with the API exposed by `Nodeos`. Through the interfaces and tools, users can obtain block data from EOSIO. However, in order to obtain the receipts and traces data, it is necessary to replay all transactions and store the generated receipts and traces in memory through `history_plugin`[5]. Since EOSIO produces one block every 0.5 seconds, its total transaction volume is much higher than that of Ethereum. Replaying such a large number of transactions takes a long time and consumes a lot of memory. EOSIO development team develops `state_history_plugin`[5] and `mongo_db_plugin`[5], which can cache receipts and traces into database engines, such as PostgreSQL and MongoDB. These plugins do not require a lot of memory, but make transaction replay slower because it takes extra time to insert traces and receipts into the database engines. *To overcome these challenges, we developed our own plugin, which is suitable for our data acquisition and exploration.* We will show more details about data acquisition of blockchain data in Section 3.

3. Raw data extraction from EOSIO

This section describes the process of obtaining raw data from the EOSIO blockchain. Figure 2 illustrates the typical EOSIO transaction execution flow, from block N to block $N+1$, with the EOSVM of the blockchain peer in the middle. During this procedure, we collect three types of raw blockchain data: Blocks, Transaction receipts, and Action traces. We next describe the details on the composition and acquisition of each kind of raw data.

3.1. Blocks

Block data is directly stored in EOSIO blockchain. Each block mainly consists of two elements:

- **Block Header:** Block header consists of some basic information of a block, including the block producer, timestamp, transaction root, etc.
- **Transactions and Actions:** Transactions construct the body of the block and each transaction consists of one or multiple actions. Each action represents a call to a smart contract, mainly

[5] `https://eosio.github.io/eos/latest/nodeos/plugins/index`

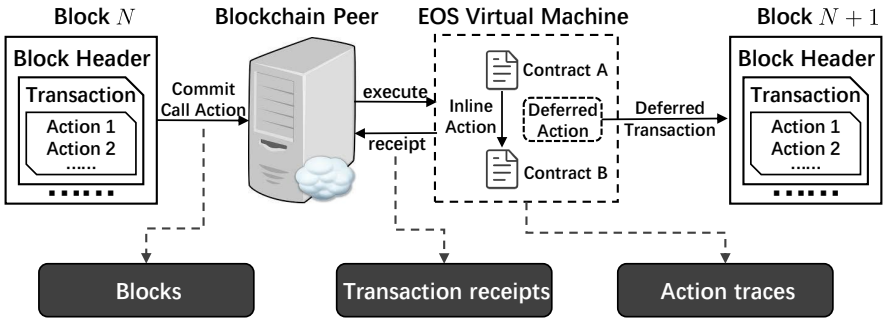


Figure 2: Raw data collection during EOSIO transaction flow

including the following fields: account (contract name), name (contract function name), data (function parameter), authorizations (authorizers).

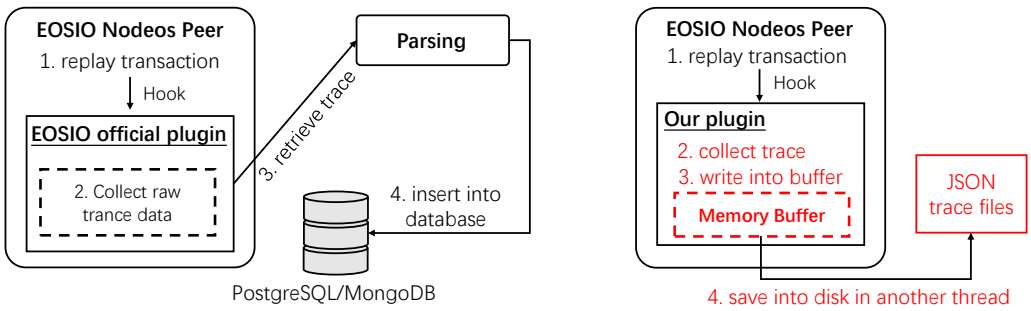
It is worth noting that EOSIO's actions can be mainly categorized into three types: *calling action*, *inline action*, and *deferred action* [34]. A *calling action* represents a user's call to a contract and an *inline action* represents a call within the contracts or between the contracts. An *inline action* is generally triggered by a *calling action* and is completed in the same transaction (block). Failure of either *inline action* or *calling action* will cause the transaction to fail. A *deferred action* is used to initiate a deferred transaction (generally packaged in a transaction of a block in the future), and its execution result does not affect the original transaction. It is important to note that only *calling action* and *deferred action* are explicitly packaged into a transaction in a block.

At present, the EOSIO development team provides users with `Nodeos` to synchronize data on the *mainnet* (the main network of EOSIO). There are two major manners to synchronize data: 1) starting `Nodeos` from the genesis block, and 2) downloading the blocks from some EOSIO backup service provider such as EOS Amsterdam[6], and starting `Nodeos` from the specified block. In order to obtain the data faster, we adopt the second method. By activating the `chain_plugin` and `http_plugin`, users can obtain each block through the RPC interface of Nodeos. However, the block data only contains partial information about *calling action* and *deferred action*, which is not enough to comprehensively analyze blockchain users. In addition, through these block data, we cannot obtain the resource consumption of the transaction (e.g., RAM) and the details of the transaction execution (e.g., what errors occurred and which other contracts were called during the transaction execution).

3.2. Action traces

Action trace data is essentially the detailed run-time data of each action that is generated in EOSVM (e.g., calls within or between the contracts, transferring EOS tokens from a contract to others). With action trace data, we can collect the detailed information about the inline actions and deferred actions (transactions). Combined with the information of the block data, we can collect the complete information about a transaction, such as which contracts and which functions are called, whether it is a deferred transaction.

[6] `https://snapshots.eosamsterdam.net/`



(a) EOSIO official plugin (b) Our `history_file_plugin`

Figure 3: Comparison of EOSIO official plugin with our `history_file_plugin`

Action trace data cannot be obtained or observed from the block data, but can be recorded during the transaction execution. Therefore, we need to replay all transactions and collect all action traces in this procedure. The EOSIO development team provides the `history_plugin` to cache the generated traces in memory. The `history_plugin` supports extremely fast querying of traces, but it also consumes huge memory due to the huge transaction volume of EOSIO. At present, almost all EOSIO full peers have closed this plugin. To address this problem, the EOSIO development team develops `state_history_plugin` and `mongo_db_plugin`, which insert the traces into database engine. These plugins aim to reduce the memory requirement and support the convenient and fast query of traces. However, these plugins would also slow down the replay procedure because it takes extra time for the official plugins to insert the traces into the database engines.

We first analyze the working flow of EOSIO official plugins. As shown in Figure 3(a), `Nodeos` will hook `state_history_plugin` or `mongo_db_plugin` when one of them is activated. These plugins collect the raw trace data when replaying transactions, then retrieve the traces and parse them into the well-formatted data being suitable for some specific database engines (such as PostgreSQL and MongoDB). Finally, the formatted traces are inserted into the database according to certain indexes. However, data parsing and insertion may slow down the replay procedure, which is not conducive to the rapid collection of trace data, especially for massive EOSIO data. Moreover, parsing such huge trace data requires extensive computing and storage resources, thereby resulting in the frequent crash of the `Nodeos` peer. It often requires substantial time, memory, and storage to collect traces with these plugins.

To address this issue, we design and develop a new plugin namely `history_file_plugin` to support the rapid collection of EOSIO trace data for subsequent processing and analysis in our XBlock-EOS. Figure 3 shows a comparison between EOSIO official plugin and our `history_file_plugin`. As shown in Figure 3(b), our `history_file_plugin` collects raw trace data and writes them into *Memory Buffer* when replaying transactions. Then, another thread asynchronously reads trace data from *Memory Buffer*, serializes, and saves them into storage devices (e.g., HDDs or SSDs) periodically. Our plugin can directly save traces as multiple files in the JSON format, i.e., a semi-structured data type similar to that in MongoDB collected by official plugins. Therefore, its data collection speed is much faster than other plugins. For example, under the same machine with a 12-core Intel Core i7-5820K@3.30GHz processor and 128GB memory, our `history_file_plugin` only takes about one day to collect the trace data of the first 20 million blocks

while EOSIO official plugins take more than a week for the same amount of trace data. Meanwhile, `history_file_plugin` also supports the collection of traces for specific block intervals. Moreover, the collected raw trace data (i.e., JSON format) also contain duplicates and redundant data. To address this challenge, we also implement several scripts to mitigate the redundant information while preserving the necessary information, and consequently save the trace data into tabular files (i.e., CSV format), which are well suitable for most machine learning tools.

In summary, our `history_file_plugin` can better meet the needs of fast data collection for data collectors and facilitate the further data analysis tasks. The modified `Nodeos` source code has also been published on the ***XBLOCK.PRO*** website[11].

3.3. Transaction receipts

In EOSIO blockchain, transaction receipts are generated after transactions are executed, which can be read by external clients or people, but cannot be obtained by the internal EOSVM. A transaction receipt records the execution of a transaction. More importantly, it contains resource consumption information for a transaction. Since EOSIO's resource management model is significantly different from other blockchains, it is necessary to collect transaction receipts and analyze user (transaction) resource usage. It can help us better understand the user and contract resource usage characteristics in EOSIO and the resource ecology of the EOSIO blockchain.

The collection of transaction receipts is similar to the collection of action traces. By activating `history_file_plugin`, `Nodeos` can save the receipts of all transactions in a certain block interval as JSON format files. We then quickly screen out the useful information from these files.

In short, there are three kinds of raw blockchain data that can be obtained from EOSIO: Blocks, Action traces, and Transaction receipts. Our plugin speeds up the collection of raw data and saves it in the JSON format according to the block number, similar to that saved in MongoDB using `mongo_db_plugin`. JSON is a semi-structured data format, which nevertheless is not the most common format used by data analytics (e.g., machine learning or data mining tools). Moreover, both EOSIO official plugins and our developed plugin collect EOSIO trace data containing duplicated and redundant information. It is necessary to simplify data representation and fasten data analysis for further research. Therefore, we implement some corresponding scripts to extract the necessary information from the collected trace data (i.e., JSON format) and save them as CSV format files. In particular, we obtain seven well-processed datasets to be elaborated in Section 4. The well-processed datasets can be easily analyzed by most data analysis tools. It is worth noting that these scripts can run in sync with our plugin to maximize the collection speed. All scripts have also been published on the ***XBLOCK.PRO*** website. These tools can be modified and tuned to fulfill different data analytics tasks.

It is well known that EOSIO is enabling millions of transactions per second. So, it is challenging to handle this huge amount of transactions. Although the average transaction throughput of EOSIO *mainnet* is only about 56.42 tps in fact (as shown in Section 4.1), it is nearly 4 times as high as that of Ethereum and 8 times as high as that of Bitcoin. According to the above analysis, if we apply the official plugins to the raw data collection of XBlock-EOS, it will require higher hardware resources and take

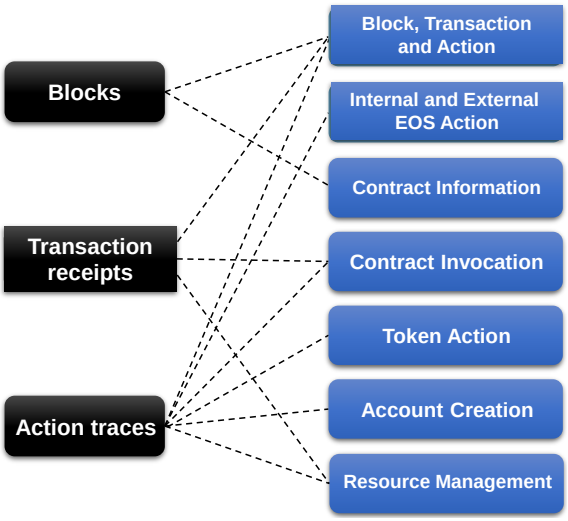


Figure 4: Mapping from raw data to seven datasets

longer time to collect EOSIO trace data. In addition, subsequent data processing needs to constantly query from the database engines, thereby significantly affecting the performance of data insertion and slowing down data collection. The official plugins prevent data collection and processing from running simultaneously, thereby greatly limiting the scalability. Our `history_file_plugin` not only speeds up the raw data collection, but also separates data collection and data processing. It allows data collection and data processing to be carried out simultaneously, consequently improving the scalability of XBlock-EOS. XBlock-EOS regularly downloads relevant snapshot and backup data from EOSIO backup service providers (e.g., EOS Amsterdam), and collects new data from a specific block number to keep up with the *mainnet*. Thus, our developed plugin as well as other corresponding tools can greatly enhance the scalability of XBlock-EOS to handle the huge amount of transactions on the EOSIO *mainnet*.

4. Data exploration of EOSIO

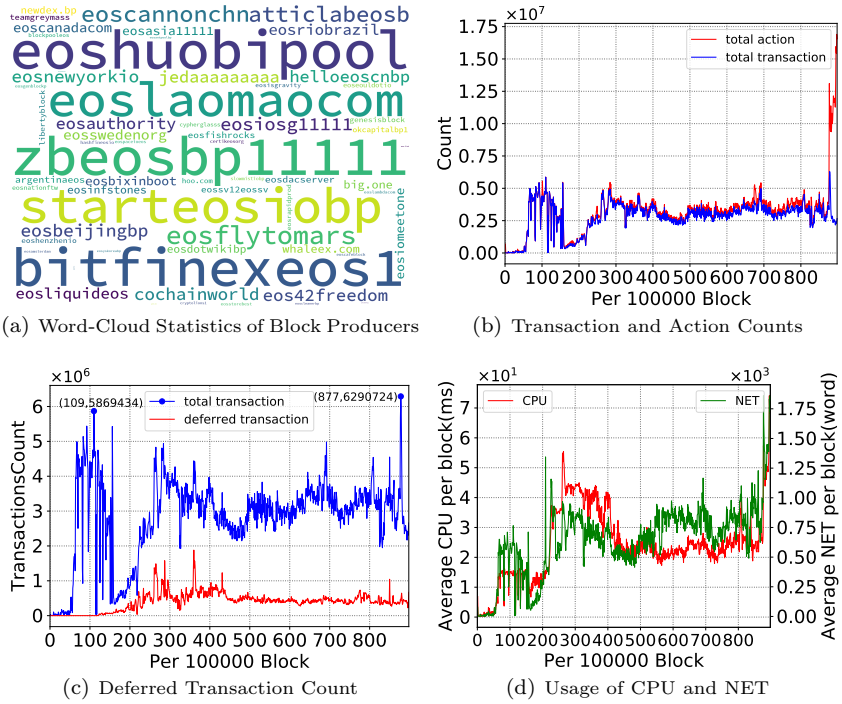
In this section, we process the obtained raw data from EOSIO and divide the raw data into seven datasets: (1) Block, Transaction and Action, (2) Internal and External EOS Transfer Action, (3) Contract Information, (4) Contract Invocation, (5) Token Action, (6) Account Creation, (7) Resource Management. Figure 4 shows the categorical relationship from the raw data to the seven datasets. We can observe that *Action traces* are the most widely used in data processing. Next, we will introduce how these seven datasets are generated, and show some statistics and observations about the datasets.

4.1. Dataset 1: Block, Transaction and Action

In order to investigate the basic statistic information of EOSIO, we extract the blocks, intra-block transactions and intra-transaction actions from EOSIO. In particular, there are 89,800,000 blocks, 2,533,292,528 transactions and 2,916,530,553 actions (excluding inline actions). We also calculate the average CPU and NET usage of every block according to the `cpu_usage_us` and `net_usage_words` of the transaction. Moreover, we count the information of block producers to measure the degree of decentralization of EOSIO.

Table 1: Statistics of Dataset 1

Statistics	Values
No. of Blocks	89,800,000
No. of Transactions	2,533,292,528
No. of Deferred transactions	357,455,192
No. of Actions (excluding inline actions)	2,916,530,553
No. of Block producers	63
Mean of Transaction Count per Block	28.21



(a) Word-Cloud Statistics of Block Producers

(b) Transaction and Action Counts

(c) Deferred Transaction Count

(d) Usage of CPU and NET

Figure 5: Statistics of Dataset 1 (better viewed in color)

There are only 63 unique block producers who however have generated 89,800,000 blocks, as shown in Table 1. In contrast to Ethereum, there are 5,122 unique miner addresses generating 8,100,000 blocks [30]. It implies that EOSIO does not have strong decentralization as Ethereum since few producers in EOSIO generate most of the blocks. Figure 5(a) shows the word-cloud statistics of account names of block producers in EOSIO. The word-cloud result also shows that several accounts almost dominate block production. These accounts have essentially been controlled by some exchanges, such as `eoshuobipool`, `zbeosbp11111`, and `bitfinexeos1`.

As shown in Table 1, the average number of transactions per block is 28.21. In other words, the average transaction throughput of EOSIO is 56.42 tps (transactions per second) since EOSIO produces a block every 0.5 seconds (i.e., tps $= 28.21/0.5 = 56.42$). When the network is active, as shown in Figure 5(c) (when blocks reaching 87,700,000), the throughput can reach about 126 tps ($\approx \frac{6,290,724}{10^5 \times 0.5}$). It shows that EOSIO does achieve significant performance improvement compared with Bitcoin and Ethereum. However, EOSIO still has a long way to go to reach its goal of million-level tps.

In EOSIO, an operation that a user interacts with the blockchain is represented as an action, and a transaction can contain one or multiple actions. Figure 5(b) plots the count of actions (represented by

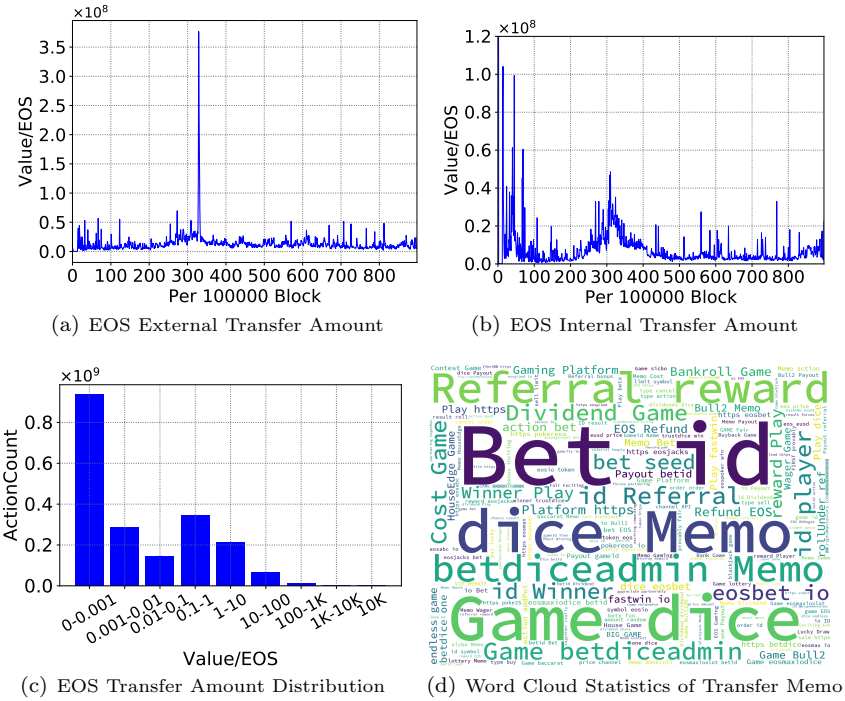
the red curve) and the count of transactions (represented by the blue curve). It is shown in Figure 5(b) that, most of the time, the count of actions is quite close to that of transactions (i.e., the count of actions is only slightly higher than that of transactions), implying that every transaction nearly contains only one action. In addition, when blocks reaching 87,600,000, the count of actions surges; this effect was essentially caused by an EIDOS *airdrop*. Anyone can transfer any amount of EOS to the contract `eidosonecoin`, and then receive the same amount of EOS and some EIDOS tokens from `eidosonecoin`. The amount of the obtained EIDOS tokens depends on the number of EOS-transfer actions but not on the transfer amount. Therefore, to gain more EIDOS tokens, many users include a number of transfer actions of 0.0001 EOS to `eidosonecoin` into a transaction.

EOSIO introduces a delayed communication mode to support initiating a transaction to be executed in the future. As shown in Table 1, there are 357,455,192 deferred transactions, accounting for about 1/7 of the total transactions. Meanwhile, as shown in Figure 5(c), when blocks reaching 35,900,000, the count of deferred transactions is close to 1/2 of that of total transactions. This shows that deferred transactions are also a daily requirement for users to interact with the EOSIO blockchain. For example, the system contract `eosio.system` in EOSIO will trigger a delayed transaction *refund* after the users redeem their mortgaged CPU or NET resources. The *refund* transaction will return the corresponding EOS to the users after three days.

CPU and NET are necessary resources for transaction execution in EOSIO. CPU is used for computation, and NET is used for network transmission between block producers. Figure 5(d) shows the statistics of average block CPU usage (in milliseconds) and NET usage (in words, and 1 word = 8 bytes) versus block count. We observe that both CPU usage and NET usage have similar trend to that of the count of transactions (as shown in Figure 5(b)). It is because the transaction count can directly affect both CPU and NET usage. However, we can see that the changing amplitude of CPU usage is different from that of NET usage at some moments. For example, the increment of CPU usage is higher than that of NET usage when blocks falling into the range from 23,0000,00 to 43,000,000. This is due to the increment in deferred transactions as shown in Figure 5(c). Generally, a deferred transaction is triggered by another original transaction. When the block producers receive the original transaction, they can generate the content of the deferred transaction during the execution process and saved them locally for future execution. Therefore, deferred transactions require almost no NET resources, whose contents rarely need to be transmitted over the network.

4.2. Dataset 2: Internal and External EOS Transfer Action

EOS, as the most representative cryptocurrency of EOSIO, was created and issued by the contract account `eosio.token` when the EOSIO *mainnet* was launched. In EOSIO, EOS transfers can be divided into *external* transfers and *internal* transfers. In general, an external transfer action represents a direct transfer from users to users (or from users to contracts), which is recorded in the transaction of the block. The internal transfer action is essentially an inline action, which is triggered by another action and is not be observed in the block. For example, when a user buys RAM or stakes EOS for CPU and NET, an inline action that transfers EOS to the system account (i.e., `eosio.ram` and `eosio.ramfee`) is triggered.



(a) EOS External Transfer Amount (b) EOS Internal Transfer Amount

(c) EOS Transfer Amount Distribution (d) Word Cloud Statistics of Transfer Memo

Figure 6: Statistics of Dataset 2 (better viewed in color)

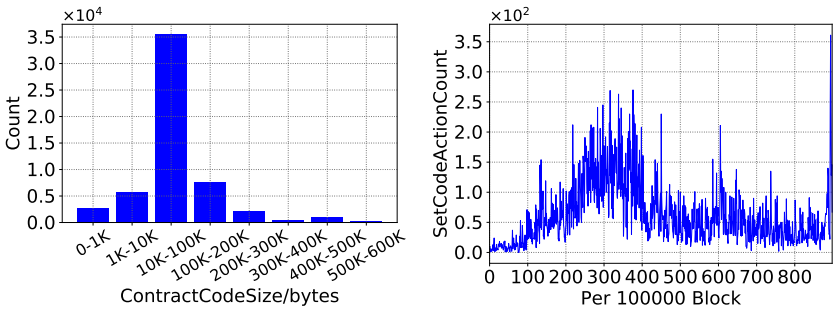
As shown in Table 2, there are 1,356,748,049 internal transfers and 653,529,552 external transfers that occur among 1,156,658 accounts.

Figures 6(a) and Figure 6(b) show the total internal transaction amount and the total external transaction amount of every 100,000 blocks, respectively. It can be seen that EOS internal and external transfers are active around the block 33,000,000, matching with the most active time of the Gambling and Games DApps [35]. In EOSIO, users have the right to write memos into an EOS transfer action. Figure 6(c) shows the visualization of the word-cloud statistics of EOS transfer memos. The result shows that many memos contain words related to gambling and games, such as *Bet*, *Dice*, *Game*, etc. It implies that the Gambling and Games DApps are prevalent in EOSIO, which is attributable to free charge for transferring money in EOSIO.

Table 2: Statistics of Dataset 2

Statistics	**Values**
No. of Internal EOS Transfers	1,356,748,049
No. of External EOS Transfers	653,529,552
No. of Accounts	1,156,658
Mean Amount of EOS	9.64
Maximum Amount of EOS	99,999,990.01

The values of EOS have a large variance as the maximum value is 99,999,990.01 EOS (about 100 million dollars when writing this paper), but the mean is only 9.64 EOS as shown in Table 2. The distribution of EOS transfer amount is shown in Figure 6(c). We can find that most EOS transfer actions fall into the range from 0.0001 to 10, and very few transfer actions exceed 1,000 EOS, indicating that most transfer actions in EOSIO only transfer a small amount of EOS.



(a) Contract code size distribution (b) Count of created contracts

Figure 7: Statistics of Dataset 3

4.3. Dataset 3: Contract Information

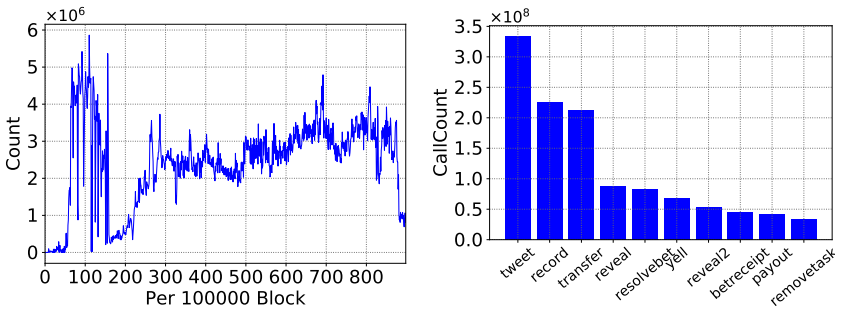
Similar to Ethereum, EOSIO also supports Turing-complete smart contracts. Users can deploy a new contract on an account through the interface `SetCode` of the system account `eosio`. It is worth noting that users can easily update or delete contract code with the same interface while this action is not allowed in Ethereum. In order to investigate all smart contracts in EOSIO, we process the raw data to obtain basic information about the smart contracts, including the *creation (update) time*, *contract code*, and *code size*. Here, we name the action of setting contracts code to empty through `SetCode` as `SetEmptyCode`, being equivalent to the removal of the contract (the contract can be deployed again on the same account later).

According to the statistics as shown in Table 3, there are only 5,594 contracts, but there are 55,735 `SetCode` actions and 1,747 `SetEmptyCode` actions. It means that most contracts have been updated multiple times after deploying. In addition, the number of contracts in EOSIO is much smaller than that in Ethereum [30], because deploying a contract in EOSIO needs to buy expensive RAM to store the contract code. Figure 7(b) shows the statistics of the total number of `SetCode` actions of every 100,000 blocks. It can be seen that when blocks reaching around 33,000,000, the creation or update of contracts is most active; this phenomenon matches the time when the gambling and game project parties launched a large number of new games or activities [35].

Table 3: Statistics of Dataset 3

Statistics	**Values**
No. of Created Contracts	5,594
No. of Contract `SetCode` Actions	55,735
No. of Contract `SetEmptyCode` Actions	1,747
Mean of Contract Hex Code Size	75,470.34

Regarding the contract code, we convert the contract code into hexadecimal code and obtain the size. Figure 7(a) shows the statistics of the contract code size distribution. In particular, the average contract code size is 75,470.34 bytes, which is much larger than that of Ethereum [30]. It is because there are more simple test contracts (with small size) in Ethereum while fewer test contracts are deployed in EOSIO. In addition, we observe that most contracts fall into the size with a range from 10k bytes to 100k bytes, implying that many contracts may look similar to each other. We will show that these contracts are



(a) Count of Contract Invocations (b) Call Type Distribution

Figure 8: Satistics of Dataset 4

related to token and gambling in the exploration of Section 4.4, further confirming the fact that there are certain similarities between the contracts.

4.4. Dataset 4: Contract Invocation

Table 4: Statistics of Dataset 4

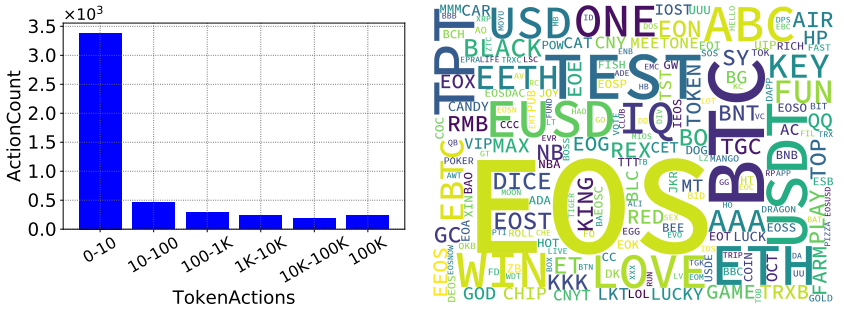
Statistics	**Values**
No. of Contract Invocation Actions	2,189,162,705
No. of Calls with Errors	14,751
No. of Authorization accounts	775,082

Unlike Ethereum, all actions (transactions) in EOSIO are completed through calling contracts, including common EOS transfers. There are several system contract accounts in EOSIO, such as `eosio`, `eosio.token`, `eosio.msig`, and so on. These system accounts are responsible for the daily affairs in EOSIO, such as transferring EOS, buying RAM, staking CPU or NET, etc. In order to investigate the contract ecology of EOSIO, we extract the invocation data of all contracts except the system contracts. The contract invocation dataset includes *calling time*, *authorizer*, *called contract*, and *calling function*. As shown in Table 4, 775,082 authorization accounts initiated a total number of 2,189,162,705 contract invocations, among which 14,751 contract invocations contain errors.

Figure 8(a) shows the count of contract invocations of every 100,000 blocks. It can be seen that when blocks are in the interval from 5,000,000 to 12,000,000, the count of contract invocations has a periodic peak. It is because a contract namely *blocktwitter* periodically launches a large number of actions named *tweets* for pressure testing, only carrying a message "*WE LOVE BM*". Figure 8(b) shows the top-10 most frequently-called functions, which account for 54.08% of all contract invocations. It indicates that most of the calling functions concentrate on some of these frequently-called functions. In addition, the function `tweet` mentioned above ranks the first, while the function `transfer` related to tokens ranks the third. We also found that the functions related to Gambling and Games DApps, such as `reveal`, `resolvebet`, and `reveal2`, also appear in the top-10 functions. Literally, these functions often represent lottery actions for Gambling and Games DApps.

Table 5: Statistics of Dataset 5

Statistics	Values
No. of Token Contracts	1,826
No. of Created and Issued Tokens	4,811
No. of Token Transfer Actions	1,128,111,142
No. of Holder Accounts	1,295,389



(a) ERC20 Popularity Distribution (b) Word-Cloud Statistics of ERC20 Tokens

Figure 9: Statistics of Dataset 5 (better viewed in color)

4.5. Dataset 5: Token Action

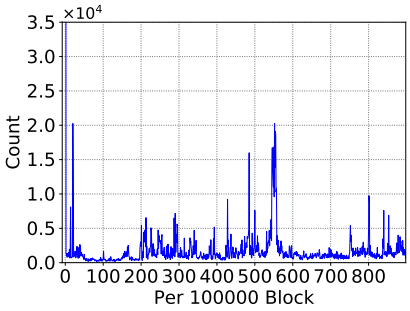
From the prior analysis in Section 4.3 and Section 4.4, we observe that token contracts are active in EOSIO. Next, we will further investigate token contracts. In EOSIO, a contract that contains three standard functions: `create`, `issue`, and `transfer` can be regarded as a standard token contract. According to this condition, we extract the token action dataset from the raw data. The token action dataset contains basic information for each token, including *name (symbol)*, *creation time*, *issuer*, *total issued amount*, and so on.

As shown in Table 5, 1,826 contracts are considered as standard token contracts, and a total of 4,811 tokens have been created and issued. It implies that in EOSIO, a contract can issue multiple tokens, which is different from that of Ethereum. In addition, a total of 1,128,111,142 token transfers occurred in 1,295,389 holding accounts. Generally, the number of holding accounts is larger than the exact number of human holders, because a real-world user often has several accounts. In addition, token issuers can send tokens directly to any account without permission, being commonly known as *Token Airdrop*.

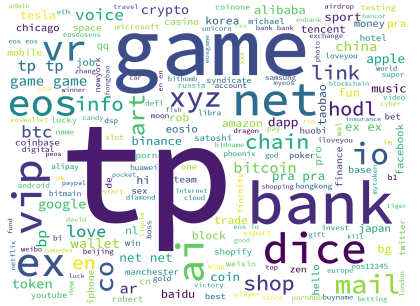
Figure 9(a) shows the distribution of the transfer count of each standard type of token in EOSIO. We can easily observe the Matthew effect [36] from Figure 9(a): the rich get richer, as most of the transfer actions occur on a few token contracts. Up to 80.02% of token contracts have less than 100 transfers. Figure 9(b) shows the word-cloud statistics of token names. It can be seen that the most common word is *EOS*, which is the name of the native cryptocurrency of EOSIO. Other common words are *BTC*, *ETH*, *USDT*, etc, which are the names of well-known cryptocurrency tokens. Meanwhile, we have also found that many token names contain the word *TEST*, indicating that these token contracts are used for testing.

Table 6: Statistics of Dataset 6

Statistics	Values
No. of *NewAccount* Actions	1,636,043
No. of Creator	45,350



(a) Created Account Count



(b) Work Cloud Statistics of Account Names

Figure 10: Statistics of Dataset 6 (better viewed in color)

4.6. Dataset 6: Account Creation

In most public blockchain systems, creating a new address (account) is easy and free. However, in EOSIO, creating a new account requires a creator to buy RAM for storing account information. In addition, the creator will generally stake some CPU and NET resources for the new account to initiate transactions. In order to investigate the account creation in EOSIO, we extract the account creation dataset from the raw data, which mainly includes the *creation time*, *creator*, *account name*, etc. It is worth noting that, in EOSIO, the account names are allowed with up to the 12-character length and each character is only allowed to use either digits `[1-5]` or alphabet letters `[a-z]`. As shown in Table 6, there are 1,636,043 different accounts (*or NewAccounts*), which were created by only 45,350 account creators. It shows that one account creator can create a large number of accounts. Specifically, each creator in EOSIO created 36.08 accounts on average. In addition, the number of the created accounts is much larger than that of the authorization accounts or token holders mentioned in the above analysis, also implying that many accounts are not active in EOSIO.

Figure 10(a) shows the created accounts of every 100,000 blocks. It can be seen that a large number of accounts were created during the initial launch of the EOSIO *mainnet*. These accounts are almost created by the system account `eosio` and their names are similar to each other, implying that they are mainly used for testing. For example, many accounts are prefixed with `hex` or `ha2`. For most of the time, the created accounts of every 100,000 blocks fall roughly in the range from 1,000 to 5,000, except for the time around the block 55,000,000. This time coincided with the time when EOSIO's total account number exceeded one million and EOSIO officially launched the REX mechanism. Figure 10(b) shows the word-cloud statistics of account names. We can find that most accounts name contain *tp* because a company called TokenPocke (TP) provides wallet services such as account creation in EOSIO. In addition, many accounts contain words, such as *game*, *dice*, *bank*, etc., further implying that the Gambling and Games DApps are quite popular in EOSIO community.

4.7. Dataset 7: Resource Management

Unlike most public blockchain systems (such as Ethereum and its variants) that adopt *gas* mechanism, EOSIO prevents malicious behaviors of contracts by limiting RAM, CPU, and NET resources. Users need to buy RAM to store information in EOSIO. The price of RAM is mainly determined by the supply-and-demand model of the market, and its core is the *bancor protocol* [37]. In addition, users need to stake CPU and NET for transaction calculation and network transmission. The amount of CPU or NET resources obtained by users is mainly determined by the proportion of EOS staked by them to that of the entire network. Since the EOSIO *mainnet* went live, the problem of insufficient CPU to complete even the simplest transfers has been criticized. Therefore, EOSIO officially launched the REX mechanism on May 1, 2019 to support the leasing service of CPU and NET to alleviate the problem. Users who cannot stake sufficient CPU or NET resources can rent from others in the system.

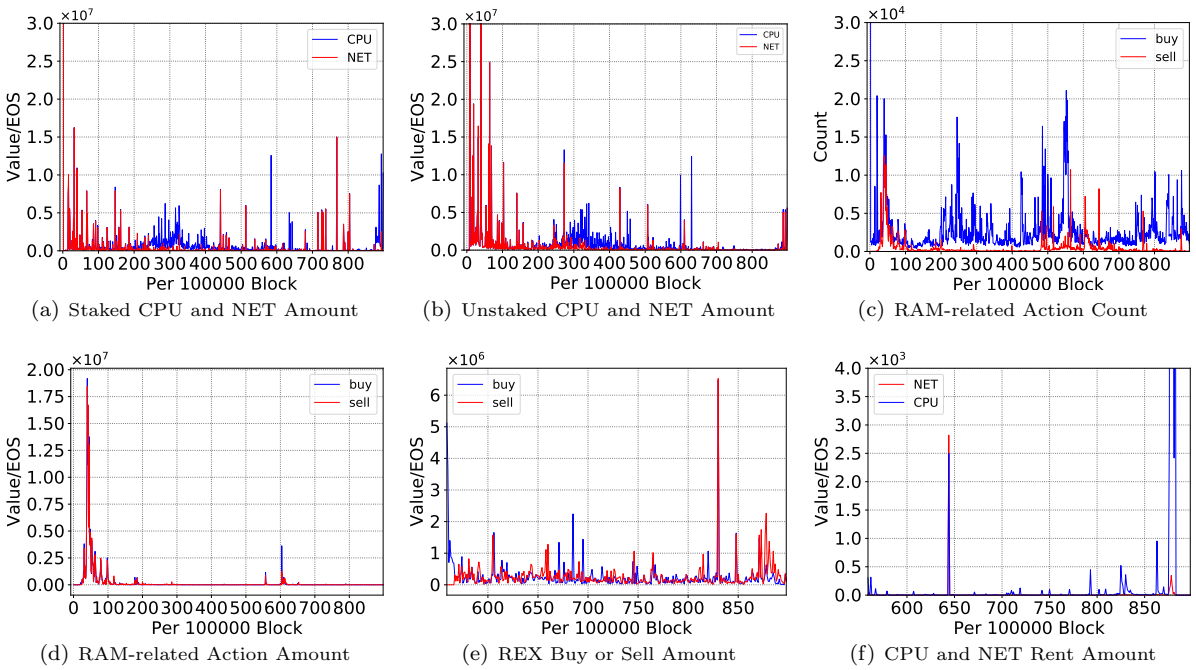
In order to investigate the resource management of EOSIO, we extract the actions related to CPU, NET, RAM, and REX from the raw data. As shown in Table 7, there are 5,474,353 CPU-related actions, including 3,805,742 `stakecpu` actions and 1,668,611 `unstakecpu` actions. Meanwhile, there are 3,100,820 NET-related actions, including 2,324,444 `stakenet` actions and 776,376 `unstakenet` actions. Figures 11(a) and 11(b) show the amount of EOS staked and unstaked, respectively. No matter being staked or unstaked, the amount of EOS corresponding to CPU is higher than that of NET, implying that CPU is a more "*important*" resource compared with NET in EOSIO. In particular, when blocks reaching about 33,000,000, the amount of EOS staked surges because a large number of Gambling and Games DApps stake substantial CPU resource for users to gamble at this time.

Table 7: Statistics of Dataset 7

Statistics	**Values**
No. of CPU-related Actions	5,474,353
No. of NET-related Actions	3,100,820
No. of RAM-related Actions	2,983,276
No. of REX-related Actions	404,355

Since the launch of the EOSIO *mainnet*, the speculation in RAM prices has continued. Some users hoarded RAM at low prices and sold at high prices to earn the difference profits. As shown in Table 7, there are a total number of 2,983,276 RAM-related actions, including 2,546,849 `buyram` actions and 436,427 `sellram` actions. As shown in Figure 11(c), the count of `buyram` actions of every 100,000 blocks is significantly larger than that of `sellram`. Meanwhile, there are multiple peaks in both `buyram` and `sellram`. However, most of the time, there is not much difference between the EOS amount of `buyram` and that of `sellram` of every 100,000 blocks. It implies that users may buy RAM multiple times and sell it at once.

In order to solve the problem that users do not have enough EOS to stake CPU, EOSIO officially launched the CPU/NET leasing mechanism, i.e., the REX mechanism, on May 1, 2019 (around the block 56,000,000). Users can store some EOS tokens in REX pool through `buyrex` action to lease to others, and retrieve EOS and get the corresponding rent at any time through `sellrex` action. Meanwhile, users can



(a) Staked CPU and NET Amount (b) Unstaked CPU and NET Amount (c) RAM-related Action Count

(d) RAM-related Action Amount (e) REX Buy or Sell Amount (f) CPU and NET Rent Amount

Figure 11: Statistics of Dataset 7 (better viewed in color)

rent CPU or NET from the REX pool by `rentcpu` or `rentnet` actions. In EOSIO, the rental income of the lessor will be affected by the supply-and-demand relationship between the lessor and lessee, so there is also speculation. As shown in Table 7, there are 404,355 REX-related actions, including 127,318 `buyrex` actions, 51,942 *sellrex* actions, 211,075 `rentcpu` actions, and 14,020 `rentnet` actions. Figure 11(e) shows the EOS amount of `buyrex` and `sellrex` actions of every 100,000 blocks. Around the block 87,000,000, the EOS amount of both `buyrex` and `sellrex` are as high as 6 million EOS; it indicates that a large amount of funds enter and exit the REX pool at this time. In addition, from Figure 11(f), we can see that the EOS amount of both `rentcpu` and `rentnet` actions increase sharply around the block 64,400,000. For most of the time, the EOS amount of `rentcpu` actions is larger than that of `rentnet` actions, implying that users have higher demands for CPU.

5. Applications of XBlock-EOS

XBlock-EOS framework can support a diversity of emerging applications. As shown in Figure 12, we categorize the applications of XBlock-EOS framework into three types: 1) Blockchain analysis in Section 5.1, 2) Smart contract analysis in Section 5.2, and 3) Cryptocurrency analysis in Section 5.3. This categorization is mainly based on the top-3 layers in the EOSIO architecture. Meanwhile, we also discuss the research opportunities in each layer, all of which contribute to building a more secure and healthier blockchain system. For example, as the key feature of blockchain, the decentralization analysis helps us to evaluate the security of a blockchain system. Meanwhile, resource management analysis helps us gain insight into the resource usage of the blockchain network and explore a more robust resource mechanism to avoid potential attacks (such as DDoS attacks). In addition, contract vulnerability detection is conducive to the security of contract funds, thereby preventing unexpected

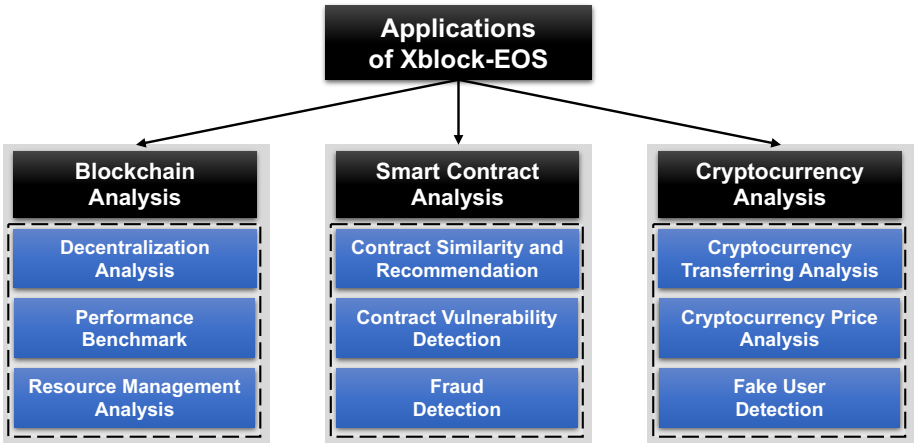


Figure 12: Applications of XBlock-EOS

capital losses.

5.1. Blockchain analysis

XBlock-EOS that has processed a large amount of data from EOSIO blockchain system can be used to support the following applications for real-world users.

5.1.1. Decentralization analysis

Decentralization, as one of blockchain key characteristics, is also the core that brings numerous merits. However, there are few studies on the decentralization evaluation of different blockchain systems, especially for DPoS-based EOSIO. In the study [20], Wang et al. proposed a measurement study for the Bitcoin mining pool. Besides, Gencer et al. proposed a measurement study on the decentralization level of the Bitcoin and Ethereum networks [38]. These studies mainly focus on PoW-based blockchain systems and only consider few metrics such as network bandwidth, network structure, and mining power. In contrast, XBlock-EOS is responsible for the provision of massive processed data of *DPOS-based blockchains* with multiple decentralization metrics. In addition, XBlock-EOS datasets can also be used to analyze the decentralization of users, contract owners, and block producers. Furthermore, the decentralization analysis on XBlock-EOS datasets can also be used to compare with blockchain systems (such as Bitcoin and Ethereum) with other consensus mechanisms like PoW.

5.1.2. Performance benchmark

Performance, especially for the transaction throughput is crucial to blockchains, thereby attracting extensive attention recently. There are many studies on blockchain performance optimization, such as Omniledger [39], RapidChain [40], and Monoxide [41]. These studies often require a substantial amount of transaction data to evaluate the performance improvement. For example, Monoxide used historical Ethereum transaction data to evaluate the performance of its optimization scheme [41]. In order to compare the performance of different optimization schemes fairly, a common benchmark for the real-world user cases for blockchain is needed although Zheng et al. [42], Xu et al. [43] and BlockBench [44] proposed preliminary performance evaluation frameworks for different types of blockchain systems. More importantly, these performance evaluation frameworks need to simulate user behaviors and generate

data similar to real-world blockchain systems. For example, Xu et al. use Hyperledger Caliper to generate transaction workloads, and perform experimental measurements to verify the correctness of their model [43]. In this regard, the large amount of data in the XBlock-EOS framework can be regarded as an effective benchmark because XBlock-EOS data has been generated by users in the real world.

5.1.3. Resource management analysis

Unlike most public blockchains using the gas mechanism, EOSIO prevents the malicious behavior of smart contracts by limiting CPU, NET, and RAM resources. In EOISO, the total amount of these three resources is fixed so users (or contracts) need to compete for these resources to meet their own demands. Therefore, since the EOSIO *mainnet* went live, speculation in resource price has continued, especially for CPU and RAM. We can learn from Section 4.7 that the count of actions related to RAM and CPU fluctuates sharply at some moments. In addition, the launch of the REX mechanism has also caused a speculative boom. Analyzing the user behaviors on resource management and identifying some speculative models in EOSIO can help users predict the price of resources and stagger the peak of speculation, thereby saving money for buying or staking resources. At present, there are some studies on the resource (gas) mechanism such as [26, 27] while few of them focus on the EOSIO's resource management model.

In addition, the security of the blockchain resource management model is also a hot topic that attracts much attention. For example, it is reported in 2016 that the Ethereum *mainnet* suffered from a large-scale DDoS attack, which severely blocked the entire network. The attackers leveraged the vulnerability of inappropriate setting *EXTCODESIZE* instruction in the gas mechanism to launch this DDoS attack. Therefore, the security of the resource management model is extremely important for blockchain systems. Currently, there are a few studies on the security of gas mechanism. For example, Chen et al. proposed an adaptive gas cost mechanism for Ethereum to defend against under-priced DoS attacks [26]. Meanwhile, Lee et al. reported some threads to the new EOSIO's resource management model and proposed some mitigation methods [16]. The analysis on the new EOSIO's management resource model can be compared with those of other blockchain systems (e.g., the gas mechanism) to promote the development of blockchain in this aspect. In summary, the resource management analysis in EOSIO will not only bring huge economic value but also promote the in-depth technology mature of blockchain systems, especially in security.

5.2. Smart contract analysis

Similar to Ethereum, EOSIO also supports smart contracts. Priory analysis presented in Section 4.4 shows that when the network is active, the count of contract invocations nearly reaches 6,000,000 per 100,000 blocks, i.e., near 120 invocations per second. It implies massive smart contracts in EOSIO are in an active state and the analysis of EOSIO smart contracts is worthwhile for conducting in-depth research in the future. We summarize the potential applications of XBlock-EOS on smart contracts as follows.

5.2.1. Contract similarity and recommendation

From the analysis in Sections 4.3 and 4.4, we observe that there is a great similarity between *smart contracts* and *contract invocations*. Code similarity evaluation and detection, especially similarity detection have been a traditional research topic in software engineering [45, 46]. Recently, some studies have also been focused on the similarity analysis of smart contract. Norvill et al. proposed a framework to group together similar contracts within the Ethereum network, and further automate labeling unknown contracts [47]. He et al. performed a detailed similarity comparison of a large number of contracts to investigate the correlation between code reuse and vulnerabilities [48]. In addition, finding similar contracts is helpful to develop new contracts. For example, developers can estimate the effects of new contracts by estimating user behaviors before deploying a new contract. In the study [49], Huang et al. proposed a method based on existing smart contracts to recommend distinguishing codes for updating smart contracts, which can help developers improve the code. Similarly, in terms of users, recommending similar smart contracts will help users find suitable contracts for them.

5.2.2. Contract vulnerability detection

Smart contract security has been a hot research topic in the blockchains, especially in Ethereum, EOSIO, and other blockchain systems. A series of attacks in Ethereum, such as TheDAO attack, have caused huge economic losses [50]. Since EOSIO went online, a number of vulnerabilities have been discovered from EOSIO's smart contracts, including fake EOS transfer, fake transfer notice and flawed random numbers generators [29]. These vulnerabilities have also brought huge economic losses, especially for Gambling and Games DApps. In order to prevent malicious attacks on smart contracts, it is an important step to perform vulnerability detection before launching online. Recently, many studies have focused on the vulnerability detection of smart contracts in Ethereum, such as Oyente [24] and Zeus [25]. Meanwhile, a few studies have performed similar analysis on EOSIO smart contracts. For example, EVulHunter can detect fake transfer vulnerabilities for EOSIO's Smart Contracts at Webassembly-level [29]. In some cases, the vulnerability detection of smart contracts may be inspired by traditional software vulnerability detection approaches, which can also verify the code. Some studies have focused on verifying the contract codes (also called "*bytecodes*" or "*opcodes*"). In this respect, the contract code data collected by XBlock-EOS can also be applied to contract vulnerability detection.

5.2.3. Fraud detection

The rapid development and prosperous popularization of smart contracts bring huge economic values. Meanwhile, smart contracts have become a means employed by malicious users to design scams to make exorbitant profits. For example, crowdfunding contracts ostensibly bring a promised return to attract victims to invest. The study [21] shows that the Ponzi scam can deceive others' cryptocurrencies. Currently, a few studies have proposed several methods for detecting fraudulent contracts and corresponding fraud activities in Ethereum [51, 52]. In addition, the study [28] gain some insights that some (contract) accounts in EOSIO are *bot-like* and can be used for malicious and fraudulent purposes including Bonus Hunting, Clicking Fraud, etc. Fortunately, most of these studies are based on the analysis of money

transfers, contract codes, and contract invocations, all of which have been contained in XBlock-EOS. Therefore, XBlock-EOS can also support further research on fraud detection.

5.3. Cryptocurrency analysis

Since the ICO waves in 2017, blockchain-based cryptocurrencies have received much attention due to decentralization and cost reduction. Ethereum, EOSIO and other blockchain systems contain a large number of cryptocurrencies including native cryptocurrencies such as ETH and EOS, as well as tokens issued in accordance with a certain protocol. For example, many tokens are issued in EOSIO using the contract *eosio.token*. It is shown in CoinMarketCap[7] that until now, there are more than 5,000 tokens used for third-party exchange with a market value of up to 15 billion dollars. Therefore, analyzing the cryptocurrency data on the blockchain can bring great economic value. We summarize the potential applications of XBlock-EOS on cryptocurrency, as described below.

5.3.1. Cryptocurrency transferring analysis

Cryptocurrency transferring analysis is common in cryptocurrency analysis. Chen et al. [22] conducted a graphical analysis of Ether transfers and derive some interesting insights. Moreover, Victor et al. [53] and Chen et al. [23] analyzed the ERC20/ERC721 token transfers network in Ethereum. Most recently, Huang et al. [28] proposed the graph analysis on EOS transfers to assist in detecting some bots and fraudulent activities in EOSIO. Following the analysis of cryptocurrency transfers, the further analysis on user behaviors can be done. For example, different tokens may form different user communities.

In addition, due to the anonymity of cryptocurrencies, blockchain has become a tool for money laundering. Cryptocurrency transferring analysis will help to identify and detect money laundering behaviors on the blockchain, consequently promoting the development of blockchain regulation. The massive processed data of XBlock-EOS will further facilitate the research on the above issues.

5.3.2. Cryptocurrency price analysis

The price of blockchain-based cryptocurrencies is susceptible to a number of different factors, such as government policies, technological innovation, socioeconomic status and related business activities. The price of cryptocurrencies is of greatest concern to the general public. Some recent studies have focused on the analysis and prediction of cryptocurrency prices [54, 55, 56]. These studies generally consist of three steps: (1) collect the price data of cryptocurrency from centralized third-party exchanges, (2) analyze the correlation between cryptocurrency prices and other potential factors, (3) forecast the future prices and potential profits. The data extracted by our XBlock-EOS can be used for the analysis to evaluate potential factors affecting the prices of cryptocurrencies, especially in the second step analysis, which is the most critical step. In addition, EOSIO data collected by XBlock-EOS collects can be jointly analyzed with other datasets collected from other blockchain systems (such as Ethereum [30]) to investigate the prices of cryptocurrencies and even the correlation between different cryptocurrency prices from multiple perspectives.

[7] `https://coinmarketcap.com/all/views/all/`

5.3.3. Fake user detection

Fake user detection has always been a hot research topic in social networks [57, 58], which can help to avoid the economic loss caused by malicious activities. The cryptocurrency users in blockchain systems also form a community-like social network, in which fake users controlled by some developers can falsify clicks to improve the rankings of DApps and even launch malicious activities. Generally speaking, the rankings of DApps or cryptocurrencies are mainly based on some objective indicators related to user activities, such as daily active users and daily user transaction volumes. Malicious developers may employ this mechanism to create fake users through various methods to increase the activity rankings of DApps. The study [28] shows that more than 30% of accounts in EOSIO were bot-like accounts through graph analysis. In addition, DAppReview[8] labels cryptocurrencies with fake users, but this fake user detection method is mainly conducted in a black box and requires manual operations. At present, there are few studies on fake user detection on DApps or cryptocurrencies. Compared to permissioned blockchain systems, there may be more fake account activities on permissionless blockchain systems. Fake user detection is of great significance for the healthy development of the permissionless blockchains. In this regard, our XBlock-EOS can support the further study on fake user detection so as to establish healthier blockchains.

6. Related work and discussion

In this section, we introduce and discuss some existing studies on blockchain data analysis. We categorize the state-of-the-art literature into two categories: *Data collection* and *Data analysis*.

Data collection is the prerequisite for blockchain data analysis while it is often challenging due to the massive volumes of blockchains. At present, the studies on blockchain data collection are mainly focused on Bitcoin and Ethereum while few of them are concentrated on EOISO. For example, DataEther [59] is a tool with code modification of the Ethereum clients to obtain the Ethereum data, while Google BigQuery [60] imports data from Bitcoin and Ethereum to enable researchers to analyze data online. XBlock-ETH [30] provides a large number of well-processed Ethereum datasets while does not include the EOSIO data. Regarding EOSIO's data collection tools, some blockchain data browsers provide data APIs for developers to use and analyze. For example, *EOSPark*[9] provides a web interface to support comprehensive queries on EOSIO data as well as some simple analysis tools on EOSIO data including blocks, transactions, contracts and tokens. In addition, *eosq*[10] provides high-precision queries on EOSIO data, supporting the queries on the detailed calling information of each transaction. However, these third-party blockchain data service providers have a number of limitations on user permissions and data usage. It is impossible for researchers to collect all the EOSIO data through these platforms. In short, most of the above data tools only provide users with tools or API services while they do not provide the up-to-date well-processed datasets.

[8] `http://dapp.review`
[9] `https://eospark.com/`
[10] `https://eosq.app/`

Data analysis of blockchain data has mainly focused on Ethereum and Bitcoin, especially Ethereum. The studies on Ethereum data analysis mainly include transaction analysis, fraud detection, smart contract security, and token analysis [21, 51, 22, 23, 25, 24]. Compared with Ethereum, EOSIO has higher volumes of various types of blockchain data. However, there are relatively few studies on the EOSIO data. Huang et al. characterize the activities in EOSIO including money transfers, account creation and contracts to detect bots and fraudulent activities [28]. In addition, EVulHunter [29] presents the first systematic attempt to automatically detect fake-transfer vulnerabilities of EOSIO's smart contract at Webassembly-level. Although these studies publish some specific types of EOSIO data, they can only be applicable to specific analysis. Moreover, datasets released by these studies lack maintenance and update.

To our best knowledge, XBlock-EOS is the first to provide such comprehensive EOSIO raw data as well as well-processed datasets. All datasets in our XBlock-EOS are public and conveniently accessible to various data users (from data analysts to DApp developers). The analysis on these data can bring huge economic values and promote the further benign development of EOSIO blockchain. Moreover, XBlock-EOS keeps updating the datasets regularly to maintain the latest EOSIO data being synchronized with the EOSIO *mainnet*.

7. Conclusion and future work

In this paper, we introduce a data collection framework of EOSIO data namely XBlock-EOS, which contains a well-processed up-to-date on-chain data of EOSIO, including blocks, transactions, actions, contracts, tokens, accounts, and resources. Moreover, this paper also presents comprehensive statistics and exploration of these processed datasets. We also discuss the emerging applications based on XBlock-EOS and outline future research opportunities. Currently, the XBlock-EOS datasets have been published on the ***XBLOCK.PRO*** website[11], through which every user can easily obtain them.

Our XBlock-EOS is promising to promote the studies in EOSIO and advance the development of blockchains. The future improvements of XBlock-EOS are described as follows:

(1) Collect off-chain data from exchanges and open-source communities: Off-chain data is also very important for blockchain data analysis, as it provides off-chain behavior information for blockchain users and developers. Our XBlock-EOS will offer the off-chain data in the future.

(2) Explore more features: This paper introduces the basic characteristics of the EOSIO data. Compared to Ethereum, EOSIO can be considered as a completely novel public blockchain system. EOSIO's architecture and design principles are very different from those of Ethereum. In the future, we will further explore the features of EOSIO data.

(3) Perform a combination analysis with other blockchain systems: In recent years, the rapid development of blockchain technologies as well as the prosperous blockchain applications have attracted a large number of users and developers. The joint analysis of EOSIO with other blockchain systems will be conducted in the future.

[11] `http://xblock.pro`

Acknowledgements

The work described in this paper was supported by the National Key Research and Development Program (2016YFB1000101), the National Natural Science Foundation of China (61722214), Key-Area Research and Development Program of Guangdong Province (2019B020214006) and Macao Science and Technology Development Fund under Macao Funding Scheme for Key R & D Projects (0025/2019/AKP).

References

[1] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system (2008).

[2] D. Berdik, S. Otoum, N. Schmidt, D. Porter, Y. Jararweh, A survey on blockchain for information systems management and security, Information Processing & Management 58 (1) (2021) 102397. doi:https://doi.org/10.1016/j.ipm.2020.102397.
URL http://www.sciencedirect.com/science/article/pii/S030645732030892X

[3] H. Baniata, A. Anaqreh, A. Kertesz, Pf-bts: A privacy-aware fog-enhanced blockchain-assisted task scheduling, Information Processing & Management 58 (1) (2021) 102393. doi:https://doi.org/10.1016/j.ipm.2020.102393.
URL http://www.sciencedirect.com/science/article/pii/S0306457320308888

[4] W. Liang, W. Huang, J. Long, K. Zhang, K. Li, D. Zhang, Deep Reinforcement Learning for Resource Protection and Real-Time Detection in IoT Environment, IEEE Internet of Things Journal 7 (7) (2020) 6392–6401.

[5] Q. Zhao, S. Chen, Z. Liu, T. Baker, Y. Zhang, Blockchain-based privacy-preserving remote data integrity checking scheme for IoT information systems, Information Processing & Management 57 (6) (2020) 102355. doi:https://doi.org/10.1016/j.ipm.2020.102355.
URL http://www.sciencedirect.com/science/article/pii/S0306457320308505

[6] L. Tseng, X. Yao, S. Otoum, M. Aloqaily, Y. Jararweh, Blockchain-based database in an iot environment: challenges, opportunities, and analysis, Cluster Computing (2020) 1–15.

[7] Y. Wu, H.-N. Dai, H. Wang, Convergence of blockchain and edge computing for secure and scalable iiot critical infrastructures in industry 4.0, IEEE Internet of Things Journal (2020) 1–1.

[8] O. Alfandi, S. Otoum, Y. Jararweh, Blockchain Solution for IoT-based Critical Infrastructures: Byzantine Fault Tolerance, in: NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, 2020, pp. 1–4.

[9] X. Pan, X. Pan, M. Song, B. Ai, Y. Ming, Blockchain technology and enterprise operational capabilities: An empirical test, International Journal of Information Management 52 (2020) 101946. doi:https://doi.org/10.1016/j.ijinfomgt.2019.05.002.
URL http://www.sciencedirect.com/science/article/pii/S0268401219301471

[10] Q. Chen, G. Srivastava, R. M. Parizi, M. Aloqaily, I. A. Ridhawi, An incentive-aware blockchain-based solution for internet of fake media things, Information Processing & Management 57 (6) (2020) 102370. doi:https://doi.org/10.1016/j.ipm.2020.102370.
URL http://www.sciencedirect.com/science/article/pii/S0306457320308657

[11] C. Oham, R. A. Michelin, R. Jurdak, S. S. Kanhere, S. Jha, B-ferl: Blockchain based framework for securing smart vehicles, Information Processing & Management 58 (1) (2021) 102426. doi:https://doi.org/10.1016/j.ipm.2020.102426.
URL http://www.sciencedirect.com/science/article/pii/S0306457320309183

[12] J. Li, J. Wu, G. Jiang, T. Srikanthan, Blockchain-based public auditing for big data in cloud storage, Information Processing & Management 57 (6) (2020) 102382. doi:https://doi.org/10.1016/j.ipm.2020.102382.
URL http://www.sciencedirect.com/science/article/pii/S0306457320308773

[13] B. Putz, M. Dietz, P. Empl, G. Pernul, Ethertwin: Blockchain-based secure digital twin information management, Information Processing & Management 58 (1) (2021) 102425. doi:https://doi.org/10.1016/j.ipm.2020.102425.
URL http://www.sciencedirect.com/science/article/pii/S0306457320309195

[14] Z. Zheng, S. Xie, H.-N. Dai, H. Wang, Blockchain challenges and opportunities: A survey, International Journal of Web and Grid Services 14 (4) (2018) 352–375.

[15] G. Wood, Ethereum: A secure decentralised generalised transaction ledger, Ethereum project yellow paper 151 (2014) (2014) 1–32.

[16] S. Lee, D. Kim, D. Kim, S. Son, Y. Kim, Who spent my EOS on the (in) security of resource management of EOS. IO, in: 13th USENIX Workshop on Offensive Technologies WOOT 19), 2019.

[17] H.-N. Dai, Z. Zheng, Y. Zhang, Blockchain for internet of things: A survey, IEEE Internet of Things Journal 6 (5) (2019) 8076 – 8094.

[18] Z. Zheng, S. Xie, H. Dai, X. Chen, H. Wang, An overview of blockchain technology: Architecture, consensus, and future trends, in: Big Data (BigData Congress), 2017 IEEE International Congress on, IEEE, 2017, pp. 557–564.

[19] P. Katsiampa, Volatility estimation for bitcoin: A comparison of garch models, Economics Letters 158 (2017) 3–6.

[20] C. Wang, X. Chu, Q. Yang, Measurement and analysis of the bitcoin networks: A view from mining pools, arXiv preprint arXiv:1902.07549 (2019).

[21] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, Y. Zhou, Detecting ponzi schemes on ethereum: Towards healthier blockchain technology, in: Proceedings of the 27th International Conference on World Wide Web, WWW, ACM, 2018.

[22] T. Chen, Y. Zhu, Z. Li, J. Chen, X. Li, X. Luo, X. Lin, X. Zhange, Understanding ethereum via graph analysis, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 1484–1492.

[23] T. Chen, Y. Zhang, Z. Li, X. Luo, T. Wang, R. Cao, X. Xiao, X. Zhang, Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum, in: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1503–1520.

[24] L. Luu, D.-H. Chu, H. Olickel, P. Saxena, A. Hobor, Making smart contracts smarter, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS, ACM, 2016, pp. 254–269.

[25] S. Kalra, S. Goel, M. Dhawan, S. Sharma, Zeus: Analyzing safety of smart contracts., in: NDSS, 2018.

[26] T. Chen, X. Li, Y. Wang, J. Chen, Z. Li, X. Luo, M. H. Au, X. Zhang, An adaptive gas cost mechanism for ethereum to defend against under-priced dos attacks, in: International Conference on Information Security Practice and Experience, Springer, 2017, pp. 3–24.

[27] G. A. Pierro, H. Rocha, The influence factors on ethereum transaction fees, in: 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), IEEE, 2019, pp. 24–31.

[28] Y. Huang, H. Wang, L. Wu, G. Tyson, X. Luo, R. Zhang, X. Liu, G. Huang, X. Jiang, Characterizing eosio blockchain, arXiv preprint arXiv:2002.05369 (2020).

[29] L. Quan, L. Wu, H. Wang, EVulHunter: Detecting Fake Transfer Vulnerabilities for EOSIO's Smart Contracts at Webassembly-level, arXiv preprint arXiv:1906.10362 (2019).

[30] P. Zheng, Z. Zheng, H.-N. Dai, XBlock-ETH: Extracting and Exploring Blockchain Data From Ethereum, IEEE Open Journal of the Computer Society 1 (2020) 95 – 106. doi:10.1109/OJCS.2020.2990458.

[31] D. Larimer, Delegated proof-of-stake (dpos), Bitshare whitepaper (2014).

[32] N. Szabo, The idea of smart contracts (1997).

[33] U. W. Chohan, Initial coin offerings (ICOs): Risks, regulation, and accountability (2019) 165–177.

[34] B. Xu, D. Luthra, Z. Cole, N. Blakely, EOS: An architectural, performance, and economic analysis (2018).

[35] T. Min, H. Wang, Y. Guo, W. Cai, Blockchain games: A survey, in: 2019 IEEE Conference on Games (CoG), IEEE, 2019, pp. 1–8.

[36] R. K. Merton, The matthew effect in science: The reward and communication systems of science are considered, Science 159 (3810) (1968) 56–63.

[37] E. Hertzog, G. Benartzi, G. Benartzi, Bancor protocol: continuous liquidity for cryptographic tokens through their smart contracts, White paper (2017).

[38] A. E. Gencer, S. Basu, I. Eyal, R. Van Renesse, E. G. Sirer, Decentralization in bitcoin and ethereum networks, arXiv preprint arXiv:1801.03998 (2018).

[39] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, B. Ford, Omniledger: A secure, scale-out, decentralized ledger via sharding, in: 2018 IEEE Symposium on Security and Privacy (SP), IEEE, 2018, pp. 583–598.

[40] M. Zamani, M. Movahedi, M. Raykova, Rapidchain: Scaling blockchain via full sharding, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2018, pp. 931–948.

[41] J. Wang, H. Wang, Monoxide: Scale out blockchains with asynchronous consensus zones, in: 16th USENIX Symposium

on Networked Systems Design and Implementation, NSDI), 2019, pp. 95–112.

[42] P. Zheng, Z. Zheng, X. Luo, X. Chen, X. Liu, A detailed and real-time performance monitoring framework for blockchain systems, in: Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP, ACM, 2018, pp. 134–143.

[43] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, A. V. Vasilakos, Latency performance modeling and analysis for hyperledger fabric blockchain network, Information Processing & Management 58 (1) (2021) 102436. doi:https://doi.org/10.1016/j.ipm.2020.102436.
URL http://www.sciencedirect.com/science/article/pii/S0306457320309298

[44] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, K.-L. Tan, Blockbench: A framework for analyzing private blockchains, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 1085–1100.

[45] M. Chilowicz, E. Duris, G. Roussel, Syntax tree fingerprinting for source code similarity detection, in: 2009 IEEE 17th International Conference on Program Comprehension, IEEE, 2009, pp. 243–247.

[46] L. Luo, J. Ming, D. Wu, P. Liu, S. Zhu, Semantics-based obfuscation-resilient binary code similarity comparison with applications to software plagiarism detection, in: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ACM, 2014, pp. 389–400.

[47] R. Norvill, B. B. F. Pontiveros, R. State, I. Awan, A. Cullen, Automated labeling of unknown contracts in ethereum, in: 2017 26th International Conference on Computer Communication and Networks (ICCCN), IEEE, 2017, pp. 1–6.

[48] N. He, L. Wu, H. Wang, Y. Guo, X. Jiang, Characterizing code clones in the ethereum smart contract ecosystem, arXiv preprint arXiv:1905.00272 (2019).

[49] Y. Huang, Q. Kong, N. Jia, X. Chen, Z. Zheng, Recommending differentiated code to support smart contract update, in: Proceedings of the 27th International Conference on Program Comprehension, IEEE Press, 2019, pp. 260–270.

[50] M. I. Mehar, C. L. Shier, A. Giambattista, E. Gong, G. Fletcher, R. Sanayhie, H. M. Kim, M. Laskowski, Understanding a revolutionary and flawed grand experiment in blockchain: the dao attack, Journal of Cases on Information Technology 21 (1) (2019) 19–32.

[51] W. Chen, Z. Zheng, E. C.-H. Ngai, P. Zheng, Y. Zhou, Exploiting blockchain data to detect smart ponzi schemes on ethereum, IEEE Access 7 (2019) 37575–37586.

[52] C. F. Torres, M. Steichen, R. State, The art of the scam: Demystifying honeypots in ethereum smart contracts, in: Proceedings of the 28th USENIX Conference on Security Symposium, SEC'19, USENIX Association, Berkeley, CA, USA, 2019, pp. 1591–1607.
URL http://dl.acm.org/citation.cfm?id=3361338.3361449

[53] F. Victor, B. K. Lüders, Measuring ethereum-based erc20 token networks, in: International Conference on Financial Cryptography and Data Security, 2019.

[54] C. Lamon, E. Nielsen, E. Redondo, Cryptocurrency price prediction using news and social media sentiment, SMU Data Sci. Rev 1 (3) (2017) 1–22.

[55] J. Abraham, D. Higdon, J. Nelson, J. Ibarra, Cryptocurrency price prediction using tweet volumes and sentiment analysis, SMU Data Science Review 1 (3) (2018) 1.

[56] W. Mensi, K. H. Al-Yahyaee, S. H. Kang, Structural breaks and double long memory of cryptocurrency prices: A comparative analysis from bitcoin and ethereum, Finance Research Letters 29 (2019) 222–230.

[57] Q. Cao, M. Sirivianos, X. Yang, T. Pregueiro, Aiding the detection of fake accounts in large scale social online services, in: Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, USENIX Association, 2012, pp. 15–15.

[58] O. Varol, E. Ferrara, C. A. Davis, F. Menczer, A. Flammini, Online human-bot interactions: Detection, estimation, and characterization, in: Eleventh international AAAI conference on web and social media, 2017.

[59] T. Chen, Z. Li, Y. Zhang, X. Luo, A. Chen, K. Yang, B. Hu, T. Zhu, S. Deng, T. Hu, Dataether: Data exploration framework for ethereum, in: Proceedings of the 39th IEEE International Conference on Distributed Computing Systems, 2019.

[60] J. Tigani, S. Naidu, Google BigQuery Analytics, John Wiley & Sons, 2014.