A Storage Architecture of Blockchain for Time-Series Data

1st Ziru Yu
School of Information Science and Technology Engineering
University of Paris-Saclay[1]
Guangzhou ForChain Technology Co,Ltd [2]
Paris, France
yuzirufish@gmail.com

2nd Yefan Cai
Guangzhou Forchain Technology Co, Ltd
Guangzhou, China
caiyefan@forchain.com.cn

3rd Weibin Hong
Guangzhou Forchain Technology Co, Ltd
Guangzhou, China
hongweibin@forchain.com.cn

***Abstract*—The blockchain technology is becoming more and more important in different areas. However, it is not suitable for storing time-series data because of its rapid growth. This paper starts from the traditional storage model of the blockchain and proposes a novel storage architecture intended for time-series data. Some implementation details are given as well. The system mainly includes three modules: block storage, index storage and time-series data storage. The time-series database is used as the basic storage tool so it can fully utilize the corresponding feature such as elasticity , automation, efficiency, etc. By proper parameter configuration, the system can retain data within a certain time window, automatically clear expired blocks, free up storage space for dynamic growth, and provide data aggregation statistics that can be directly applied to data analysis systems. Therefore, it's more suitable for time-series data in some areas.**

Keywords—storage, time-series, blockchain

I. INTRODUCTION

Blockchain is a technical framework that is based on the principles of Bitcoin, and its name is derived from the latter's data storage structure[1-2]. The storage model of a typical blockchain system is a chain structure in which many chronologically generated block files are concatenated in series. In addition to the block files, the blockchain node usually uses the key-value database as a secondary index, stores the ledger state and historical transaction, to support the query and processing of the ledger.

With the development of Internet, there is more and more time-series data. [3]. The time-series database is a special database designed for time-series data, providing a more efficient model, such as InfluxDB and OpenTSDB. Because the blockchain node uses key-value database , it is not appropriate to directly store the time-series data in the blockchain.

In this paper, we propose a novel storage architecture of blockchain for time-series data. By combining the blockchain with the time-series database, this architecture can work more efficiently in some areas than the traditional key-value storage model.

II. ARCHITECTURE

A. System Structure

The storage architecture is shown in Figure 1, which mainly includes three modules: block storage, index storage and time-series data storage.

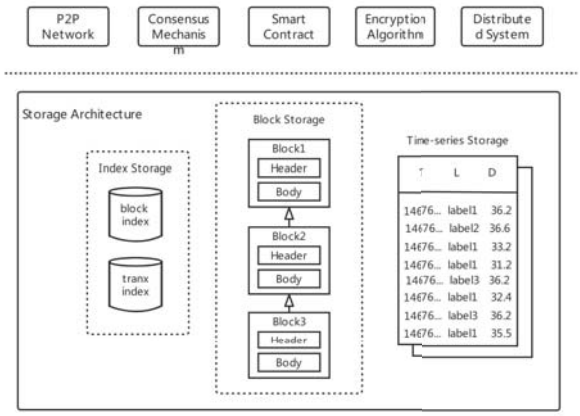

Fig. 1. The storage architecture of blockchain.

The block storage module is implemented based on the file system for storing block files, each block file includes a block header and a block body. The block header includes some meta data. The block body contains the specific content of each transaction, organized by Merkle tree. This module is similar to the typical blockchain storage structure.

The index storage module is implemented based on LevelDB, including the block index and the transaction index. The block index records the file address and relative offset of a certain block and the transaction index records the block where the transaction is located. This module is mainly used for auxiliary index of block and transaction, which realizes fast search and improves processing speed.

The time-series data storage module is implemented based on a time-series database, such as InfluxDB or OpenTSDB. This type of database enables high-concurrency inserts and queries, provides multiple data aggregation statistics, and automatically eliminates expire data based on retention policies. With the support of the time-series database, this module stores time-series data and encapsulates unified aggregate statistics to provide an interface for upper-layer applications.

III. IMPLEMENTATION

A. Data Query

The data query interface provided by the storage system includes block query, transaction query and time-series data query, as shown in Figure 2.

The query mode of block includes time stamp, block number and block hash. Similarly, the query mode of

978-1-7281-5781-8/19/$31.00 ©2019 IEEE

transaction includes transaction number and the block number. Both of these queries are obtained by the index storage module. The process will firstly locate the target item according to the index and the structured data will be returned by the engine after deserializing.

The query mode of time-series data includes label, time range and some complex conditions, as well as some common aggregation functions. These operations are supported by the underlying time-series database. The process will utilize the database to fetch the data and return the structured data after processing.

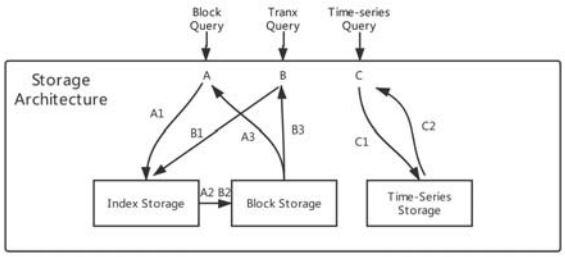

Fig. 2. Data query procedure.

B. Data Insert

The data insert procedure is shown in Figure 3, which can be described as follow.

1. Calculate the meta information of the block file, including block hash, block size, time stamp, etc.

2. The block file is stored in the file system through the block storage module, and the location information of the block file is obtained.

3. Store the location information of the file into the index storage module.

4. Write the operation content into the transaction, that is, the time-series data, through the time-series data storage module, and write the data into the corresponding underlying time-series database.

5. The result is fed back to the client. If any step in the middle fails, the previous operation will be rolled back and the error information is returned to the client.

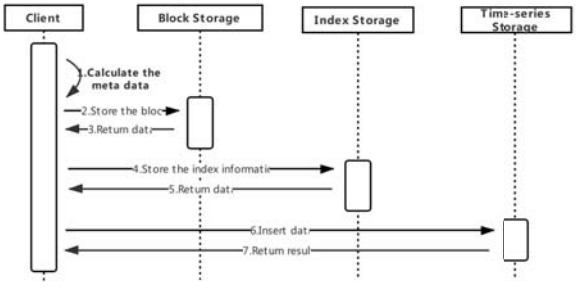

Fig. 3. Data insert procedure.

C. Data Cleanup

After the node is started, you need to configure related parameters. The start up process is described as follows:

1. When the initial node starts, the expiration time and the cleaning interval are set by means of environment variables or configuration files.

2. After setting the parameters, the node starts a timing task, sets the execution interval and periodically executes the task of data cleanup.

3. Pass the expiration time parameter to the underlying time-series database, and the database itself completes the cleaning of the expired data according its retention policy.

After the timing task is triggered, the process of data cleanup is shown in Figure 4.

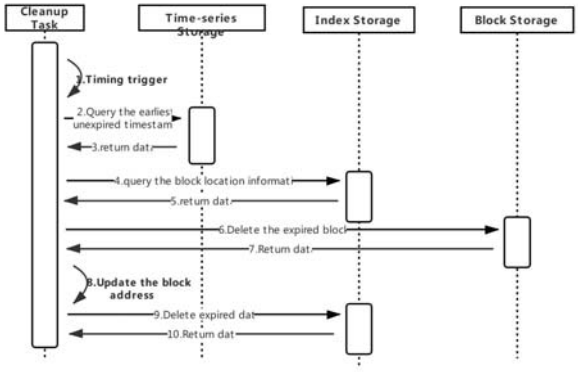

Fig. 4. The process of data cleanup.

1. Query the time stamp of the earliest unexpired data through the time-series database, and if not, end the operation.

2. Through the index data module, query the block location information corresponding to the time stamp obtained in step 1.

3. Through the block data module, locate the block obtained in step 2, delete all the block files generated before the block, and update the address of the genesis block.

4. Delete all expired data related to the index storage module.

5. The result is fed back to the client. If any step in the middle is wrong, the previous operation will be rolled back and the execution is attempted again.

IV. Conclusion

In this paper, we design a blockchain storage architecture for time-series data. The system has native support for time-series data in the underlying storage tool. By proper parameter configuration, the system can retain data within a certain time window, automatically clear expired blocks, free up storage space for dynamic growth, and provide data aggregation statistics that can be directly applied to data analysis systems.

References

[1] Crosby M, Pattanayak P, Verma S, et al. Blockchain technology: Beyond bitcoin[J]. Applied Innovation, 2016, 2(6-10): 71.

[2] Pilkington M. 11 Blockchain technology: principles and applications[J]. Research handbook on digital transformations, 2016, 225.

[3] Naqvi S N Z, Yfantidou S, Zimányi E. Time series databases and influxdb[J]. Studienarbeit, Université Libre de Bruxelles, 2017.