October 2019

DISTRIBUTED LEDGER ADDRESSING

Jeff Venable

DISTRIBUTED LEDGER ADDRESSING

AUTHOR:

Jeff Venable

ABSTRACT

Proposed herein is an object-granularity addressing scheme to perform input/output operations on a blockchain.

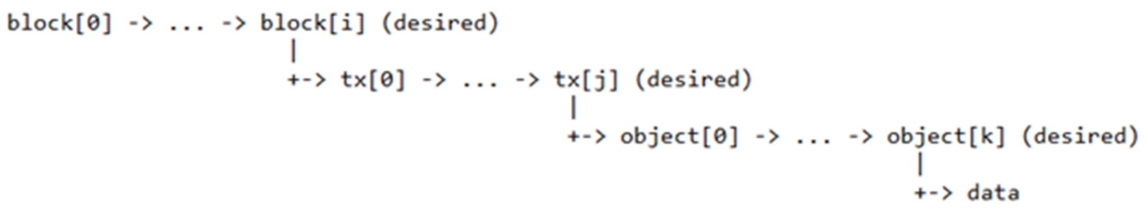
DETAILED DESCRIPTION

Storage of blockchains can become an issue as the blockchain grows in size. Size of a blockchain can be particularly relevant for certain applications, such as Internet of Things (IoT) applications, because devices that make up the IoT network tend to have a small memory. The size of a blockchain can also be relevant if the nodes of a blockchain network are routing devices, such as switches and/or routers, because such devices tend to have small local memories. Thus, referencing data within a distributed ledger needs to be network efficient, require only small storage size both at rest and in flight, and should avoid costly lookup resolutions (e.g., transaction hash -> file offset key/value index).

For a traditional blockchain, a key-value index is built in a database and the key-value index mapping from Transaction Hash (unique value, typically SHA-256) to an offset in a file that represents the blockchain is stored on a disk. Therefore, to find a transaction, one passes in the hash of the transaction and, after a database key-value query, it is known where on the disk the transaction is stored.

Proposed herein is an object-granularity addressing scheme to perform input/output operations on a blockchain. The addressing scheme reflects an object hierarchy according to which the blockchain is built.

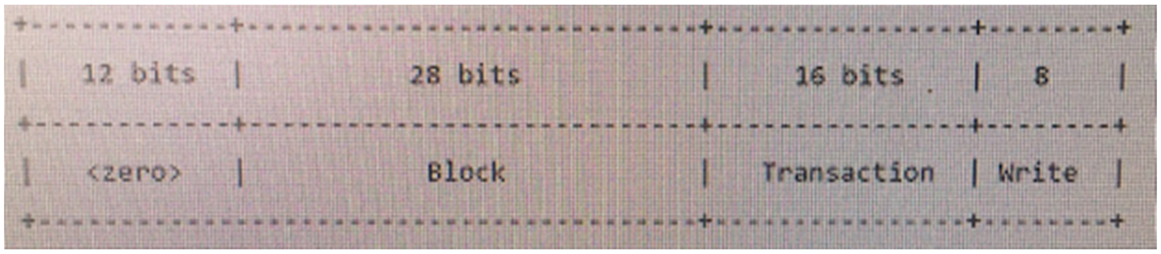
In accordance with embodiments presented herein, a blockchain includes one or more blocks and each block includes one or more transaction. In addition, each transaction includes one or more transaction objects, as shown in FIG. 1, below.



```
block[0] -> ... -> block[i] (desired)
                    |
                    +-> tx[0] -> ... -> tx[j] (desired)
                                          |
                                          +-> object[0] -> ... -> object[k] (desired)
                                                                   |
                                                                   +-> data
```

FIG. 1

As used herein, the term "object" may refer to a block, a transaction, or a transaction object. Conceptually, a blockchain can be viewed as a list (e.g., array or linked list) of blocks, where each block is a list of transactions and each transaction is a list of transaction objects. In the hierarchy defined above, the blockchain may be referred to as a top level or higher level object than a block, a block may be referred to as a higher level object than a transaction, and a transaction may be referred to as a higher level object than a transaction object. FIG. 2, below, illustrates an example addressing scheme in accordance with embodiments presented herein.



```
+-----------+-----------------------+--------------+--------+
|  12 bits  |        28 bits        |   16 bits    |   8    |
+-----------+-----------------------+--------------+--------+
|  <zero>   |         Block         | Transaction  | Write  |
+-----------------------------------+--------------+--------+
```

FIG. 2

The "write" bit region in FIG. 2, above, refers to a "transaction object." In the addressing scheme proposed herein, certain regions or bit ranges of an address are reserved for certain objects. Each bit range of an address represents an index of an object of the blockchain hierarchy. The bit ranges are ordered such that the bit ranges with the more significant digits represent higher order objects of the blockchain hierarchy. Each block can be accessed by an index within the blockchain, each transaction can be accessed by an index within a block, and each transaction object can be accessed by an index within the transaction.

The addressing scheme above has many applications. One of the applications is to improve the way a computer stores and retrieves data in memory. For example, every transaction in a block of a blockchain may contain several signatures, such as a signature from the submitter of the transaction, signatures from one or more endorsers who approved the smart contract resulting in the transaction, signatures from participants in the transaction other than the approvers, *etc*. Signing a transaction with a signatures includes adding the public key of the signer to the transaction. A public key can be, for example, 1024, 2048, or 4096 bits in size. The same public key may have been previously published within the blockchain. For every signature, rather than adding the public key, a pointer to the public key previously published may be included in place of the key. The pointer would be in the form of an address that follows the addressing scheme described herein. If the pointer is 64 bits, for example, then substitution of the pointer for the public keys is a 16 to 1, a 32 to 1, or a 64 to 1 compression ratio without loss of information. Pointers can be used throughout the blockchain, for any repeated object, not just public keys.

Another use case may involve a smart contract that would like to push out a new network configuration globally to all nodes of a blockchain. In such examples, rather than sending out the configuration details themselves, the smart contract can push out a 64 bit address and every node can look up the configuration.

The addressing scheme presented herein can allow iteration through the blockchain by object, rather than by a fixed size offset. Objects in a blockchain may have variable sizes. Data on each block may be organized using transaction separators and transaction object separators to mark the boundaries of the objects. If not all transactions are the same size, transactions can be iterated through by adding a 1 to the least significant transaction bit of the address, which will create an address pointing to the next transaction in the sequence independent from the size of the transaction. One can iterate through objects by adding a 1 to the least most significant bit of the address region representing that object.

In addition, objects can be compared using the addressing scheme to determine which object was placed on the blockchain before the other object. The object that is represented by an address region has a higher numerical value than another object of the same type was placed on the blockchain later in time.

Preferably, the address size is the same size or smaller size than at least two registers of the CPU of the device on which the blockchain is stored. This way, a single address fits into a single register, and computations can be performed quickly using CPU registers. One possibility is to load two addresses into two registers and quickly compare them within one CPU cycle.