

PERFORMING PARALLEL EXECUTION OF TRANSACTIONS IN A DISTRIBUTED LEDGER SYSTEM

TECHNICAL FIELD

5 **[0001]** This specification relates to transaction execution in a distributed ledger system.

BACKGROUND

10 **[0002]** Distributed ledger systems (DLSs), which can also be referred to as consensus networks, for example, blockchain networks, enable participating entities to securely, and immutably store data. Examples of blockchain networks can include: public blockchain networks, private blockchain networks, and consortium blockchain networks. A public blockchain network is open for all entities to use the DLS, and participate in the consensus process. A private blockchain network is provided for a particular entity,
15 which centrally controls read and write permissions. A consortium blockchain network is provided for a select group of entities, which control the consensus process, and includes an access control layer.

20 **[0003]** A blockchain network is a network of computing nodes that manage, update, and maintain one or more blockchain structures. A blockchain is a data structure that stores transactions in a way that allows future transactions to be verified for consistency with all prior transactions stored in the chain. The transactions are executed and recorded in the blockchain by each of the network nodes of the blockchain network.

25 **[0004]** One issue encountered in blockchain networks is the speed at which transactions are processed. Generally, network nodes in the blockchain network process transactions serially in the order they are submitted. This can lead to low transaction throughput and delays between when a transaction is submitted and when it clears.

30 **[0005]** Although multiple existing techniques can be used for executing the transactions among the network nodes of a blockchain system, a more efficient solution to executing the transactions would be advantageous.

SUMMARY

[0006] This specification describes technologies for transaction execution in a distributed ledger system (e.g., a blockchain network). These technologies generally involve performing parallel execution of transactions by a network node in a distributed ledger system. The described technologies can improve processing speed of transactions in a blockchain network and increase transaction throughput of the blockchain network.

[0007] This specification also provides one or more non-transitory computer-readable storage media coupled to one or more processors and having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations in accordance with embodiments of the methods provided herein.

[0008] This specification further provides a system for implementing the methods provided herein. The system includes one or more processors, and a computer-readable storage medium coupled to the one or more processors having instructions stored thereon which, when executed by the one or more processors, cause the one or more processors to perform operations in accordance with embodiments of the methods provided herein.

[0009] It is appreciated that methods in accordance with this specification may include any combination of the aspects and features described herein. That is, methods in accordance with this specification are not limited to the combinations of aspects and features specifically described herein, but also include any combination of the aspects and features provided.

[0010] The details of one or more embodiments of this specification are set forth in the accompanying drawings and the description below. Other features and advantages of this specification will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

[0011] FIG. 1 depicts an example of an environment that can be used to execute embodiments of the specification.

[0012] FIG. 2 depicts an example of an architecture in accordance with embodiments of the specification.

[0013] FIG. 3A depicts an example of a serial execution order for transactions in a blockchain network in accordance with embodiments of the specification.

[0014] FIG. 3B depicts an example of a parallel execution order for transactions in a blockchain network in accordance with embodiments of the specification.

[0015] FIG. 3C depicts an example of an execution order for failed transactions in a blockchain network in accordance with embodiments of the specification.

5 [0016] FIG. 4 depicts an example of a process that can be executed in accordance with embodiments of the specification.

[0017] FIG. 5 depicts examples of modules of an apparatus in accordance with embodiments of the specification.

[0018] Like reference symbols in the various drawings indicate like elements.

10

DETAILED DESCRIPTION

[0019] This specification describes technologies for transaction execution in a distributed ledger system (e.g., a blockchain network). These technologies generally involve performing parallel execution of transactions such as smart contract transactions by a network node in a distributed ledger system. The described technologies can improve processing speed of smart contract transactions in a blockchain network and increase transaction throughput of the blockchain network.

15 [0020] To provide further context for embodiments of this specification, and as introduced above, distributed ledger systems (DLSs), which can also be referred to as consensus networks (e.g., made up of peer-to-peer nodes), and blockchain networks, enable participating entities to securely, and immutably conduct transactions, and store data. Although the term blockchain is generally associated with particular networks, and/or use cases, blockchain is used herein to generally refer to a DLS without reference to any particular use case.

20 [0021] A blockchain is a data structure that stores transactions in a way that the transactions are immutable. Thus, transactions recorded on a blockchain are reliable and trustworthy. A blockchain includes one or more blocks. Each block in the chain is linked to a previous block immediately before it in the chain by including a cryptographic hash of the previous block. Each block also includes a timestamp, its own cryptographic hash, and one or more transactions. The transactions, which have already been verified by the nodes of the blockchain network, are hashed and encoded into a Merkle tree. A Merkle

25

30

tree is a data structure in which data at the leaf nodes of the tree is hashed, and all hashes in each branch of the tree are concatenated at the root of the branch. This process continues up the tree to the root of the entire tree, which stores a hash that is representative of all data in the tree. A hash purporting to be of a transaction stored in the tree can be quickly verified by determining whether it is consistent with the structure of the tree.

[0022] Whereas a blockchain is a decentralized or at least partially decentralized data structure for storing transactions, a blockchain network is a network of computing nodes that manage, update, and maintain one or more blockchains by broadcasting, verifying and validating transactions, etc. As introduced above, a blockchain network can be provided as a public blockchain network, a private blockchain network, or a consortium blockchain network. Embodiments of this specification are described in further detail herein with reference to a consortium blockchain network. It is contemplated, however, that embodiments of this specification can be realized in any appropriate type of blockchain network.

[0023] In general, a consortium blockchain network is private among the participating entities. In a consortium blockchain network, the consensus process is controlled by an authorized set of nodes, which can be referred to as consensus nodes, one or more consensus nodes being operated by a respective entity (e.g., a financial institution, insurance company). For example, a consortium of ten (10) entities (e.g., financial institutions, insurance companies) can operate a consortium blockchain network, each of which operates at least one node in the consortium blockchain network.

[0024] In some examples, within a consortium blockchain network, a global blockchain is provided as a blockchain that is replicated across all nodes. That is, all consensus nodes are in perfect state consensus with respect to the global blockchain. To achieve consensus (e.g., agreement to the addition of a block to a blockchain), a consensus protocol is implemented within the consortium blockchain network. For example, the consortium blockchain network can implement a practical Byzantine fault tolerance (PBFT) consensus, described in further detail below.

[0025] FIG. 1 is a diagram illustrating an example of an environment 100 that can be used to execute embodiments of this specification. In some examples, the example

environment 100 enables entities to participate in a consortium blockchain network 102. The example environment 100 includes computing devices 106, 108, and a network 110. In some examples, the network 110 includes a local area network (LAN), wide area network (WAN), the Internet, or a combination thereof, and connects web sites, user devices (e.g., computing devices), and back-end systems. In some examples, the network
5 110 can be accessed over a wired and/or a wireless communications link. In some examples, the network 110 enables communication with, and within the consortium blockchain network 102. In general the network 110 represents one or more communication networks. In some cases, the computing devices 106, 108 can be nodes
10 of a cloud computing system (not shown), or each computing device 106, 108 can be a separate cloud computing system including the multiple computers interconnected by a network and functioning as a distributed processing system.

[0026] In the depicted example, the computing systems 106, 108 can each include any appropriate computing system that enables participation as a node in the consortium
15 blockchain network 102. Example computing devices include, without limitation, a server, a desktop computer, a laptop computer, a tablet computing device, and a smartphone. In some examples, the computing systems 106, 108 hosts one or more computer-implemented services for interacting with the consortium blockchain network 102. For example, the computing system 106 can host computer-implemented services of
20 a first entity (e.g., user A), such as a transaction management system that the first entity uses to manage its transactions with one or more other entities (e.g., other users). The computing system 108 can host computer-implemented services of a second entity (e.g., user B), such as a transaction management system that the second entity uses to manage its transactions with one or more other entities (e.g., other users). In the example of FIG.
25 1, the consortium blockchain network 102 is represented as a peer-to-peer network of nodes, and the computing systems 106, 108 provide nodes of the first entity, and second entity respectively, which participate in the consortium blockchain network 102.

[0027] FIG. 2 depicts an example of an architecture 200 in accordance with embodiments of this specification. The example of the architecture 200 includes an entity
30 layer 202, a hosted services layer 204, and a blockchain network layer 206. In the depicted example, the entity layer 202 includes three participants, Participant A,

Participant B, and Participant C, each participant having a respective transaction management system 208.

[0028] In the depicted example, the hosted services layer 204 includes interfaces 210 for each transaction management system 210. In some examples, a respective transaction management system 208 communicates with a respective interface 210 over a network (e.g., the network 110 of FIG. 1) using a protocol (e.g., hypertext transfer protocol secure (HTTPS)). In some examples, each interface 210 provides communication connection between a respective transaction management system 208, and the blockchain network layer 206. More particularly, the interface 210 communicate with a blockchain network 212 of the blockchain network layer 206. In some examples, communication between an interface 210, and the blockchain network layer 206 is conducted using remote procedure calls (RPCs). In some examples, the interfaces 210 “host” blockchain network nodes for the respective transaction management systems 208. For example, the interfaces 210 provide the application programming interface (API) for access to blockchain network 212.

[0029] As described herein, the blockchain network 212 is provided as a peer-to-peer network including multiple nodes 214 that immutably record information in a blockchain 216. Although a single blockchain 216 is schematically depicted, multiple copies of the blockchain 216 are provided, and are maintained across the blockchain network 212. For example, each node 214 stores a copy of the blockchain. In some embodiments, the blockchain 216 stores information associated with transactions that are performed between two or more entities participating in the consortium blockchain network.

[0030] A blockchain (e.g., the blockchain 216 of FIG. 2) is made up of a chain of blocks, each block storing data. Examples of data include transaction data representative of a transaction between two or more participants. While transactions are used herein by way of non-limiting example, it is contemplated that any appropriate data can be stored in a blockchain (e.g., documents, images, videos, audio). Examples of a transaction can include, without limitation, exchanges of something of value (e.g., assets, products, services, currency). The transaction data is immutably stored within the blockchain. That is, the transaction data cannot be changed.

[0031] Before storing in a block, the transaction data is hashed. Hashing is a process of transforming the transaction data (provided as string data) into a fixed-length hash value (also provided as string data). It is not possible to un-hash the hash value to obtain the transaction data. Hashing ensures that even a slight change in the transaction data results in a completely different hash value. Further, and as noted above, the hash value is of fixed length. That is, no matter the size of the transaction data the length of the hash value is fixed. Hashing includes processing the transaction data through a hash function to generate the hash value. An example of a hash function includes, without limitation, the secure hash algorithm (SHA)-256, which outputs 256-bit hash values.

[0032] Transaction data of multiple transactions are hashed and stored in a block. For example, hash values of two transactions are provided, and are themselves hashed to provide another hash. This process is repeated until, for all transactions to be stored in a block, a single hash value is provided. This hash value is referred to as a Merkle root hash, and is stored in a header of the block. A change in any of the transactions will result in change in its hash value, and ultimately, a change in the Merkle root hash.

[0033] Blocks are added to the blockchain through a consensus protocol. Multiple nodes within the blockchain network participate in the consensus protocol, and perform work to have a block added to the blockchain. Such nodes are referred to as consensus nodes. PBFT, introduced above, is used as a non-limiting example of a consensus protocol. The consensus nodes execute the consensus protocol to add transactions to the blockchain, and update the overall state of the blockchain network.

[0034] In further detail, the consensus node generates a block header, hashes all of the transactions in the block, and combines the hash value in pairs to generate further hash values until a single hash value is provided for all transactions in the block (the Merkle root hash). This hash is added to the block header. The consensus node also determines the hash value of the most recent block in the blockchain (i.e., the last block added to the blockchain). The consensus node also adds a nonce value, and a timestamp to the block header.

[0035] In general, PBFT provides a practical Byzantine state machine replication that tolerates Byzantine faults (e.g., malfunctioning nodes, malicious nodes). This is achieved in PBFT by assuming that faults will occur (e.g., assuming the existence of independent

node failures, and/or manipulated messages sent by consensus nodes). In PBFT, the consensus nodes are provided in a sequence that includes a primary consensus node, and backup consensus nodes. The primary consensus node is periodically changed, Transactions are added to the blockchain by all consensus nodes within the blockchain network reaching an agreement as to the world state of the blockchain network. In this process, messages are transmitted between consensus nodes, and each consensus nodes proves that a message is received from a specified peer node, and verifies that the message was not modified during transmission.

[0036] In PBFT, the consensus protocol is provided in multiple phases with all consensus nodes beginning in the same state. To begin, a client sends a request to the primary consensus node to invoke a service operation (e.g., execute a transaction within the blockchain network). In response to receiving the request, the primary consensus node multicasts the request to the backup consensus nodes. The backup consensus nodes execute the request, and each sends a reply to the client. The client waits until a threshold number of replies are received. In some examples, the client waits for $f+1$ replies to be received, where f is the maximum number of faulty consensus nodes that can be tolerated within the blockchain network. The final result is that a sufficient number of consensus nodes come to an agreement on the order of the record that is to be added to the blockchain, and the record is either accepted, or rejected.

[0037] In some blockchain networks, cryptography is implemented to maintain privacy of transactions. For example, if two nodes want to keep a transaction private, such that other nodes in the blockchain network cannot discern details of the transaction, the nodes can encrypt the transaction data. An example of cryptography includes, without limitation, symmetric encryption, and asymmetric encryption. Symmetric encryption refers to an encryption process that uses a single key for both encryption (generating ciphertext from plaintext), and decryption (generating plaintext from ciphertext). In symmetric encryption, the same key is available to multiple nodes, so each node can en-/de-crypt transaction data.

[0038] Asymmetric encryption uses keys pairs that each include a private key, and a public key, the private key being known only to a respective node, and the public key being known to any or all other nodes in the blockchain network. A node can use the

public key of another node to encrypt data, and the encrypted data can be decrypted using other node's private key. For example, and referring again to FIG. 2, Participant A can use Participant B's public key to encrypt data, and send the encrypted data to Participant B. Participant B can use its private key to decrypt the encrypted data (ciphertext) and extract the original data (plaintext). Messages encrypted with a node's public key can only be decrypted using the node's private key.

[0039] Asymmetric encryption is used to provide digital signatures, which enables participants in a transaction to confirm other participants in the transaction, as well as the validity of the transaction. For example, a node can digitally sign a message, and another node can confirm that the message was sent by the node based on the digital signature of Participant A. Digital signatures can also be used to ensure that messages are not tampered with in transit. For example, and again referencing FIG. 2, Participant A is to send a message to Participant B. Participant A generates a hash of the message, and then, using its private key, encrypts the hash to provide a digital signature as the encrypted hash. Participant A appends the digital signature to the message, and sends the message with digital signature to Participant B. Participant B decrypts the digital signature using the public key of Participant A, and extracts the hash. Participant B hashes the message and compares the hashes. If the hashes are same, Participant B can confirm that the message was indeed from Participant A, and was not tampered with.

[0040] A consensus version of a blockchain can be determined based on interacting with nodes of a blockchain network. For example, a web server that is a node of the blockchain network can use a longest-chain and/or a heaviest-chain criteria to select a chain of blocks as the consensus version of the blockchain from among multiple candidate paths. The multiple candidate paths can include different blocks received at different times from different nodes of the blockchain network.

[0041] As described above, a blockchain network enables participants to conduct transactions, for example, to buy/sell goods, and/or services. In some embodiments, each participant is associated with one or more accounts. A transaction can involve one or more participants and execution of the transactions may affect one or more accounts of the one or more participants. As an example, a fund transfer transaction from Participant

A to Participant B can result in a decrease of fund in Account A of Participant A and an increase of fund in Account B of Participant B.

[0042] In some embodiments, record-keeping models are used to record transactions and the corresponding accounts between participants. Examples of record-keeping models include an unspent transaction output (UTXO) model and an account model (also referred to as account-based model or account/balance model).

[0043] In the UTXO model, the assets on the chain are in the form of transactions. Each transaction spends output from prior transactions and generates new outputs that can be spent in subsequent transactions. A participant's unspent transactions are tracked, and a balance that the participants has to spend is calculated as the sum of the unspent transactions. Each transaction takes one or more unspent outputs (and only unspent outputs) as input and can have one or more outputs. The requirement that only unspent outputs may be used in further transactions is necessary to prevent double spending and fraud.

[0044] The account model performs record keeping and manages account balances like a traditional bank. Under this model, an account can have an address and a corresponding account balance. Assets on the chain are represented as the balance of the account. Each transfer transaction can have an account address of a transferred asset and an account address of a received asset. The transaction amount is directly updated on the balance of the account. The account model is efficient, as each transaction may only need to validate that the sending account has enough balance to pay for the transaction. In addition to supporting the transaction validation and proof function, the account model can fully support smart contracts, especially those that require state information or involve the multiple parties.

[0045] In some embodiments, a transaction includes a message packet sent by an external account to another account on the blockchain. The transaction may include a sender's signature, a recipient's address, and a token that the sender transfers to the recipient. The transaction may also include information about a smart contract. Every transaction can be a record on the blockchain.

[0046] In some embodiments, a smart contract is a computer program designed to disseminate, verify, and/or execute a contract by a data processing system, e.g., a

blockchain consensus network. Smart contracts allow for trusted transactions without participation of a third party. The transactions are traceable and irreversible.

[0047] In some embodiments, transactions in the blockchain system may include multiple types such as transfers, contract deployments, contract calls, contract updates, deposits, etc. In some embodiment, regardless of the type of transaction, a transaction can includes a sender, a recipient, an amount of the transfer, data required for the contract, a hash of the transaction, and a signature.

[0048] In some embodiments, a transaction can be categorized as a first-type transaction or a second-type transaction depending on whether accounts affected by execution of the transaction can be pre-determined or ascertained before the execution of the transaction. For the first-type transaction, one or more accounts affected by execution of the first-type transaction can be pre-determined before the execution of the first-type transaction. Examples of the first-type transaction can include a fund transfer transaction as described above, where the accounts (e.g., Account A of Participant A and Account B of Participant B) affected by the fund transfer transaction can be determined before the execution of the fund transfer transaction between Participant A and Participant B.

[0049] For the second-type transaction, one or more accounts affected by execution of the second-type transaction cannot be pre-determined or ascertained before the execution of the second-type transaction. Examples of the second-type transaction can include a smart contract transaction such as a call of a smart contract. A smart contract transaction can involve one or more participants in execution of a smart contract. The accounts affected by the execution of the smart contract transaction may depend on a current state of the blockchain at the time of execution, and thus cannot be ascertained before the actual execution of the smart contract transaction. As such, two or more smart contract transactions may not be executed in parallel. Because a smart contract call can lead to the execution of instructions making up the smart contract, it may not be possible to determine the scope of accounts a particular contract call will affect. For example, consider a smart contract that takes a particular account and a payment amount as parameters, and applies the payment amount to the particular account if some condition is true. Because a caller of this smart contract specifies the particular account and the condition depends on the state of the blockchain when the smart contract is executed, it

may not be possible to ascertain, from the definition of the smart contract itself (e.g., its source code), which accounts a particular call to the smart contract will affect. In some embodiments, the contract call may be a transaction with the potential to affect all accounts in the blockchain network. Therefore the contract call cannot be executed in parallel with any other transactions.

[0050] To provide further context for embodiments of the specification, FIG. 3A depicts an example of a serial execution order 300 for transactions in a blockchain network in accordance with embodiments of the specification. As shown, the execution order 300 includes multiple transactions (302a-d, 304a-c, 306a-c, and 308a-b) sequenced according to the order in which they are to be executed by network nodes of a blockchain network. The execution order 300 is a serial execution order where each individual transaction of transactions 302a-d, 304a-c, 306a-c, and 308a-b is executed one by one. The execution order 300 can be the same execution order among all consensus nodes (e.g., network nodes that participate in a consensus protocol) of a blockchain network. For example, the execution order 300 can be an agreed execution order of the multiple transactions after a consensus process performed by all consensus nodes of the blockchain network. The serial execution order 300 can be used to ensure that the final execution results of different blockchain nodes are consistent.

[0051] In some embodiments, each of the multiple transactions includes a second-type transaction such as a smart contract transaction. As mentioned, the accounts affected by the execution of the second-type transaction cannot be pre-determined or ascertained before the execution of the second-type transaction because the execution of the second-type transaction may depend on a current or latest state of the blockchain in the blockchain network. In some embodiments, to estimate accounts affected by execution of a second-type transaction, the second-type transaction can be pre-executed by a network node, for example, before it is the second-type transaction's turn for execution in the multiple transactions. For example, the second-type transaction can be pre-executed by the network node before performing a consensus process of the multiple transactions.

[0052] For example, after receiving a smart contract transaction, a network node can add the smart contract transaction into a transaction list in a cache. When the CPU or one of the processors or cores of the network node is idle, the network node can remove the

smart contract transaction from the transaction list in the cache, and pre-execute the smart contract transaction based on the latest state of the blockchain of the network node at the time of the pre-execution, for example, before the network node performs a consensus process of all transactions in the transaction list. As such, one or more accounts affected by the pre-execution of the smart contract transaction can be determined after the pre-execution. The one or more accounts affected by the pre-execution of the smart contract transaction can be used as an estimate or prediction of one or more accounts affected by the actual execution of the smart contract transaction. In some embodiments, if the one or more accounts affected by the pre-execution of the smart contract transaction are not the same as the one or more accounts affected by the actual execution of the smart contract transaction, the pre-execution of the smart contract transaction can be rolled back to undo any changes to the account due to the pre-execution. As such, the account status are not affected.

[0053] In some embodiments, a network node (e.g., the network node that receives the smart contract transaction from a client and pre-executes the smart contract transaction) can record the one or more accounts affected by the pre-execution of the smart contract transaction, for example, by writing into the smart contract transaction message as an additional field or element in a data structure of the smart contract transaction message. Both the smart contract transaction and the corresponding one or more accounts affected by the pre-execution of the smart contract transaction can be subject to a consensus process performed by all the network nodes. This can avoid repeated pre-execution operations of the smart contract transaction by other network nodes, and thus save the computational resources.

[0054] In some embodiments, the multiple transactions 302a-d, 304a-c, 306a-c, and 308a-b are transactions received during an epoch of a consensus process. In some embodiments, a consensus process or mechanism is designed to achieve reliability in a network involving the multiple nodes. For example, a blockchain network relies on consensus mechanisms to reach agreement among network nodes of the blockchain network. An epoch of a consensus represents a round of consensus among the multiple network nodes of the blockchain network. For example, each network node may periodically collect pending transactions and submit their respectively received pending

transactions to a consensus process so as to obtain a list of transactions to be executed by each network node in the blockchain network.

[0055] In some embodiments, the order in which the transactions are received by each node may be different from the order in which the transactions are sent by participants. In some embodiments, the consensus operation of each node on the transaction will further lead to the uncertainty of the transaction order of the list of transactions after the consensus is performed. In some embodiments, each network node sorts or orders the multiple transactions according to certain rules before executing the multiple transactions, and the final execution results of each node can be consistent as long as the ordering rules or protocol of the nodes are the same among the network nodes of the blockchain network.

[0056] In some embodiments, based on the estimated accounts affected by the execution of the smart contract transaction (e.g., the accounts affected by the pre-execution of the smart contract transaction), the smart contract transactions can be divided into one or more groups, where accounts affected by pre-execution of smart contract transactions in one group do not overlap with accounts affected by pre-execution of smart contract transactions in another group. For example, given Transaction 1 affecting Account A and Account B, Transaction 2 affecting the Account B and Account C, Transaction 3 affects Account D and Account E, Transaction 1 and Transaction 2 affect a common account, Account B, and thus cannot be executed at the same time. As such, Transactions 1, 2, and 3 can be divided into two groups, where group I includes Transactions 1 and 2 that affect a common account, Account B, and group II includes Transaction 3. In some embodiments, the relative execution order of the two transactions, Transaction 1 and Transaction 2, can be arbitrary. Group I and group II, however, can be executed in parallel as they do not affect any common account. In some embodiments, as long as each network node divides the groups in the same way, and the execution order of transactions within the group is the same, a consistent final execution result of each node can be ensured.

[0057] As another example, as shown in FIG. 3B, the transactions 302a-d, 304a-c, 306a-c, and 308a-b as shown in FIG. 3A can be divided into 4 groups, for example, based on whether pre-execution of the transactions affect one or more common transaction

entities (e.g., a transferee or a sender, a transferor or a receiver, or their corresponding accounts) or have dependency, for example, in affecting one or more same or common accounts. As shown in FIG. 3B, transactions 302a-d represent a first group 340a of smart contract transactions that affect a first common transaction entity according to the pre-execution results of the transactions 302a-d; transactions 304a-c represent a second group 340b of smart contract transactions that affect a second common transaction entity according to the pre-execution results of the transactions 304a-c; transactions 306a-c represent a third group 340c of smart contract transactions that affect a third common transaction entity according to the pre-execution results of the transactions 306a-c; and transactions 308a-b represent a fourth group 340d of smart contract transactions that affect a fourth common transaction entity according to the pre-execution results of the transactions 308a-b. Between each two of the groups 340a, 340b, 340c, and 340d, transactions in one group do not affect a same account as transactions in another group do, the pre-execution results of the transactions.

[0058] If two or more transactions could affect one or more common accounts, the two or more transactions may not be executed in parallel at the same time and the two or more transactions can be grouped into a single group. In other words, within a single group, pre-execution of the smart contract transactions in the single group affect one or more same accounts; whereas between two different groups, one or more accounts affected by pre-execution of the smart contract transactions in one group do not overlap with one or more accounts affected by pre-execution of the smart contract transactions in the other group. As a result, the smart contract transactions in the single group are to be executed in serial, whereas the smart contract transactions in the different groups can be executed in parallel. The relative execution order between or among the two or more transactions may be arbitrary, for example, determined according to certain protocol or ordering rules that are agreed by all the network nodes in the blockchain network. In some embodiments, as long as each network node divides the groups in the same way, and the pre-execution order of transactions within the group is the same, a consistent final pre-execution result of each node can be ensured.

[0059] FIG. 3B depicts an example of a parallel execution order 350 for transactions in a blockchain network in accordance with embodiments of the specification. According

to the parallel execution order 350, the groups 340a, 340b, 340c, and 340d of smart contract transactions can be executed in parallel by the network nodes of the blockchain network. Executing the transaction groups 340a, 340b, 340c, and 304d in parallel can leverage multi-core or multi-thread processing power of each network node, and lead to increases in the processing speed and transaction throughput in the blockchain network, as the network is now executing four transactions at any one time in parallel rather than just one if all transactions were executed serially.

[0060] In some embodiments, each network node of the blockchain network executes the smart contract transactions of each group in parallel, for example, according to the parallel execution order 350 based on a current or latest state of the blockchain of blockchain network. In some embodiments, one or more accounts affected by the actual execution of the smart contract transaction may be different from the one or more accounts affected by the pre-execution of the smart contract transaction because the latest state of the blockchain at the time of actual execution may be different from the latest state of the blockchain of blockchain network at the time of pre-execution or execution of a previous smart contract transaction may affect the execution of a current transaction and the one or more accounts affected by the execution of the current transaction. In this case, the execution of the smart contract transaction can be rolled back or undone. Such a smart contract transaction can be referred to as a failed smart contract transaction and added to a list of failed transactions. The list of failed transactions can be re-executed serially after parallel execution of all the other transactions. In some embodiments, the transactions in the list of failed transactions can be sorted according to certain rules agreed by all the network nodes in the blockchain network so as to ensure a consistent execution result across all the blockchain network.

[0061] FIG. 3C depicts an example of an execution order 390 for failed transactions in a blockchain network in accordance with embodiments of the specification. In this example, after the actual execution of the groups 340a, 340b, 340c, and 340d of smart contract transactions according to the parallel execution order 350, it can be determined smart contraction transactions 308a and 308b are failed transaction because one or more accounts affected by the actual execution of the smart contract transactions 308a and 308b are different from the one or more accounts affected by the pre-execution of the

smart contract transactions 308a and 308b, respectively. In this case, the actual execution of the smart contract transactions 308a and 308b are rolled back. The smart contract transactions 308a and 308b are put into a list of failed transactions and are re-executed after the parallel execution of the actual execution of the groups 340a, 340b, 340c and 340d of smart contract transactions according to the parallel execution order 350.

[0062] In some embodiments, for each network node in the blockchain network, as long as smart contract transactions are grouped according to the same rule (e.g., based on the pre-execution results of the smart contract transactions), the order of transactions within the group is consistent, failed transactions are rolled back and re-executed in serial according to a same rule after the actual execution of the other smart contract transactions, a consistent final execution result among all the network nodes in the blockchain network can be obtained.

[0063] FIG. 4 depicts an example of a process 400 that can be executed in accordance with embodiments of the specification. In some embodiments, the process 400 may be performed using one or more computer-executable programs executed using one or more computing devices. For example, the process 400 may be performed by each network node in a blockchain network. For clarity of presentation, the description that follows generally describes method 400 in the context of the other figures in this description. It will be understood that method 400 may be performed, for example, by any suitable system, environment, software, and hardware, or a combination of systems, environments, software, and hardware, as appropriate. In some embodiments, various steps of method 400 can be run in parallel, in combination, in loops, or in any order.

[0064] At 402, multiple transactions to be executed in a blockchain network are received by a network node of the blockchain network. The network node is one of the multiple network nodes of the blockchain network. The multiple transactions can include, for example, the transactions 302a-d, 304a-c, 306a-c, and 308a-b as shown in FIG. 3A. In some embodiments, each of the multiple transactions can include a smart contract transaction such as a call to a smart contract. In some embodiments, each of the multiple transactions includes a transaction for which one or more accounts affected by executing the transaction is not deterministic (i.e., cannot be ascertained) before executing the transaction. In other words, execution of each of the multiple transactions can affect one

or more accounts but the one or more accounts cannot be pre-determined or ascertained before the execution of the each of the multiple transactions. In some embodiments, a pre-execution of the each of the multiple transactions can be used to predict or estimate the one or more accounts affected by the actual execution of the each of the multiple transactions.

[0065] In some embodiments, the multiple transactions do not need to be executed by the network node pursuant to a predetermined or mandatory order. In other words, the relative execution order among the multiple transactions are not of essence as long as all the network nodes in the blockchain network execute the multiple transactions according to a same order.

[0066] In some embodiments, each of the network nodes of the blockchain network can receive a respective number of transactions to be executed in the blockchain network, for example, from one or more clients connected to the respective network nodes of the blockchain network. In some embodiments, the transactions include all transactions received from all network nodes of the blockchain network, for example, during a time period (e.g., an epoch of a consensus process). The transactions can form a transaction list that is subject to a consensus process performed by all network nodes of the blockchain network.

[0067] At 404, for each of the multiple transactions, the transaction is pre-executed by the network node based on a first current state of a blockchain in the blockchain network before performing a consensus process of the plurality of transactions, and one or more accounts affected by the pre-executing the transaction are determined. In some embodiments, the first current state of a blockchain in the blockchain network can be the current or latest state of the blockchain at the time of the pre-execution of the transaction (e.g., before the final order of execution of the multiple transactions is determined). In some embodiments, the transaction is pre-executed by the network node when one or more processors of the network node is idle. In some embodiments, a transaction can be pre-executed by the network node while the network node is receiving another transaction or performing other operations, for example, by leveraging multi-core or parallel processing capabilities of the network node. In some embodiments, the pre-

execution of the transactions can make better use of the computational resources or processing power of the network node without introducing additional delay or latency.

[0068] In some embodiments, the transaction that has been pre-executed can be rolled-back so as to avoid any changes to the first state of the blockchain in the blockchain network. In some embodiments, the transaction that has been pre-executed
5 can be rolled-back before performing a consensus process of the number of transactions. In some embodiments, the pre-execution of the transaction can be performed on a copy of a data structure storing the first current state of the blockchain (e.g., a world state or global state Merkle Patricia Tree (MPT) tree) so that the first current state of the
10 blockchain in the blockchain network remains unchanged due to the pre-executing of the second-type transaction.

[0069] In some embodiments, the one or more accounts affected by the pre-executing the transaction can be recorded or saved, for example, as a list or another data structure together with the transaction. In some embodiments, the one or more accounts affected
15 by the pre-executing the transaction can also be subject to a consensus process performed by all network nodes of the blockchain network for performing consensus of the one or more accounts affected by the pre-executing the transaction. By recording the one or more accounts affected by the pre-executing the transaction and submit them for the consensus process by the network node can avoid repeated pre-execution operations of
20 the transaction by other network nodes, and thus save the computational resources.

[0070] At 406, for each of the multiple transactions, a consensus process of the multiple transactions and the one or more accounts affected by the pre-executing the transaction are performed by the network node. The consensus process can be performed, for example, according to a consensus algorithm or protocol adopted by the blockchain
25 network.

[0071] At 408, the multiple transactions are divided into one or more groups of transactions by the network node based on the one or more accounts affected by the pre-executing the transaction for each of the multiple transactions. Each group of transactions include one or more transactions that affect one or more common transaction entities.
30 Between each two different groups of transactions, any transaction in one group does not affect any common transaction entity with any transaction in another group. The common

transaction entity can include, for example, a transferee, a transferor, an account of a transferee, or an account of a transferor associated with a transaction.

[0072] For example, FIG. 3B shows an example of a division of the transactions 302a-d, 304a-c, 306a-c, and 308a-b into four groups 340a-d based on the one or more accounts affected by the pre-executing the transactions 302a-d, 304a-c, 306a-c, and 308a-b.

[0073] At 410, the multiple transactions are executed by executing the one or more groups of transactions in parallel based on a second current state of the blockchain in the blockchain network. For example, FIG. 3B shows an example where the transactions 302a-d, 304a-c, 306a-c, and 308a-b are executed by executing the four groups 340a-d of smart contract transactions in parallel according to the parallel execution order 350. In some embodiments, the four groups 340a-d of smart contract transactions are executed in parallel based on the second current state of the blockchain in the blockchain network such as the current or latest state of the blockchain at the time of the executing the respective transactions (e.g., at the time of the parallel execution of the transactions 302a-d, 304a-c, 306a-c, and 308a-b). In some embodiments, the second current state of the blockchain is different from the first current state of the blockchain in the blockchain network. For example, the second current state of the blockchain is a latter state than the first current state of the blockchain. In some instances, data saved in the blockchain in the second current state may be different from the data saved in the blockchain in the first current state. In this case, the executing of a transaction based on the second current state of the blockchain may affect different accounts than those affected by pre-executing the transaction based on the first current state of the blockchain.

[0074] At 412, for each of the multiple transactions, one or more accounts affected by the executing the transaction are determined. For example, once the transaction is executed, the one or more accounts affected by the executing the transaction can be ascertained.

[0075] At 414, it is determined if the one or more accounts affected by the executing the transaction are the same as the one or more accounts affected by the pre-executing the transaction and if the one or more accounts affected by the executing the transaction are not affected by any previously executed transactions in the multiple transactions.

[0076] At 416, in response to determining that, for one of the multiple transactions, the one or more accounts affected by the executing the transaction are the same as the one or more accounts affected by the pre-executing the transaction and the one or more accounts affected by the executing the transaction are not affected by any previously executed transactions in the multiple transactions, the execution of the transaction is committed. In some embodiments, committing the execution of the multiple transactions can include one or more of writing the execution results of the multiple transactions into the blockchain of the blockchain network, or returning the execution results of the multiple transactions to one or more clients of the blockchain network.

[0077] At 418, in response to determining that, for one of the multiple transactions, the one or more accounts affected by the executing the transaction are not the same as the one or more accounts affected by the pre-executing the transaction, or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the multiple transactions, the execution of the transaction is rolled back.

[0078] At 420, such a transaction can be re-executed after executing the one or more groups of transactions in parallel. In some embodiments, such a transaction can be identified as a failed transaction (e.g., the transaction 308a or 308b as shown in FIG. 3C).

[0079] In some embodiments, one or more failed transactions out of the multiple transactions can be identified, wherein for each of the one or more failed transactions, one or more accounts affected by the executing the failed transaction are not the same as one or more accounts affected by the pre-executing the failed transaction or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the multiple transactions. The one or more failed transactions can be re-executed after executing the one or more groups of transactions in parallel. In some embodiments, all the failed transactions can be added to a failed transaction list. All the failed transactions in the failed transaction list can be re-executed in serial after executing the one or more groups of transactions in parallel.

[0080] In some embodiments, after the re-execution of the failed transactions, the process 400 proceeds to 416 where the re-execution of failed second-type transactions transaction is committed.

[0081] In some embodiments, the multiple transactions are executed by the network node in a same order as the multiple transactions are executed by any other network node of the multiple network nodes of the blockchain network. For example, each network node can determine, according to a protocol agreed by the multiple network nodes of the blockchain network, a sequential order of executing one or more transactions within each of the one or more groups; and a sequential order of executing the one or more failed transactions after executing the one or more groups of transactions in parallel. In some embodiments, as long as the multiple transactions are grouped according to the same rule (e.g., based on the pre-execution results of the smart contract transactions), the execution order of transactions within the group is consistent, failed transactions are rolled back and re-executed in serial after the actual execution of the other smart contract transactions, for example, according to a same rule, a consistent final execution result among all the network nodes in the blockchain network can be obtained.

[0082] FIG. 5 is a diagram of an example of modules of an apparatus 500 in accordance with embodiments of this specification. The apparatus 500 can be an example embodiment of a blockchain network node configured to perform parallel execution of smart contract transactions, wherein the blockchain network is a consortium blockchain network. The apparatus 500 can correspond to the embodiments described above, and the apparatus 500 includes the following: a receiver or a receiving module 502 for receiving multiple transactions; a pre-executing module 504 for pre-executing each of the multiple transactions based on a first current state of a blockchain in the blockchain network before performing a consensus process of the plurality of transactions; a first determining module 506 for determining one or more accounts affected by the pre-executing each of the multiple transactions; a consensus module 508 for performing a consensus process of the multiple transactions and the one or more accounts affected by the pre-executing the transaction, a divider or dividing module 510 for dividing the multiple transactions into a group of transactions based on the one or more accounts affected by the pre-executing the transaction for each of the multiple transactions; an execution module 512 for executing the multiple transactions by executing the one or more groups of transactions in parallel based on a second current state of the blockchain in the blockchain network; a second determining module 514 for

determining one or more accounts affected by the executing one of the multiple transactions, and determine if the one or more accounts affected by the executing the transaction are the same as one or more accounts affected by the pre-executing the transaction and if the one or more accounts affected by the executing the transaction are not affected by any previously executed transactions in the multiple transactions; and a committing module 516 for committing the executing a transaction in response to determining that the one or more accounts affected by the executing the transaction are the same as the one or more accounts affected by the pre-executing the transaction and the one or more accounts affected by the executing the transaction are not affected by any previously executed transactions in the multiple transactions.

[0083] In an optional embodiment, the apparatus 500 further includes the following: a rolling back module 518 for rolling back the executing one of the multiple transactions in response to determining that the one or more accounts affected by the executing the transaction are not the same as the one or more accounts affected by the pre-executing the transaction, or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the multiple transactions; and a re-executing module 520 for re-executing the transaction after executing the one or more groups of transactions in parallel.

[0084] In an optional embodiment, the apparatus 500 further includes the following: a recording module 522 for recording the one or more accounts affected by the pre-executing each of the multiple transactions for performing consensus of the one or more accounts affected by the pre-executing the transaction.

[0085] In an optional embodiment, the multiple transactions are executed by the network node in a same order as the multiple transactions are executed by any other network node of the multiple network nodes of the blockchain network.

[0086] In an optional embodiment, the apparatus 500 further includes the following: an identifying module 524 for identifying one or more failed transactions, wherein for each of the one or more failed transactions, one or more accounts affected by the executing the failed transaction are not the same as one or more accounts affected by the pre-executing the failed transaction or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the multiple

transactions; and the re-executing module 520 for re-executing the one or more failed transactions after executing the one or more groups of transactions in parallel.

[0087] In an optional embodiment, the apparatus 500 further includes the following: a third determining module 526 for determining a sequential order of executing one or more transactions within each of the one or more groups; and a fourth determining module 528 for determining a sequential order of executing the one or more failed transactions after executing the one or more groups of transactions in parallel.

[0088] In an optional embodiment, each group of transactions includes one or more transactions that affect one or more common transaction entities; and between each two different groups of transactions, any transaction in one group does not affect any common transaction entity with any transaction in another group.

[0089] In an optional embodiment, the common transaction entity includes a transferee, a transferor, an account of a transferee, or an account of a transferor associated with a transaction.

[0090] In an optional embodiment, each of the multiple transactions includes a smart contract transaction.

[0091] In an optional embodiment, each of the multiple transactions includes a transaction for which one or more accounts affected by executing the transaction is not deterministic before executing the transaction.

[0092] In an optional embodiment, pre-executing, by the network node, the transaction includes pre-executing the transaction, by the network node, when one or more processors of the network node is idle.

[0093] The system, apparatus, module, or unit illustrated in the previous embodiments can be implemented by using a computer chip or an entity, or can be implemented by using a product having a certain function. A typical embodiment device is a computer, and the computer can be a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email receiving and sending device, a game console, a tablet computer, a wearable device, or any combination of these devices.

[0094] For an embodiment process of functions and roles of each module in the apparatus, references can be made to an embodiment process of corresponding steps in the previous method. Details are omitted here for simplicity.

[0095] Because an apparatus embodiment basically corresponds to a method embodiment, for related parts, references can be made to related descriptions in the method embodiment. The previously described apparatus embodiment is merely an example. The units described as separate parts may or may not be physically separate, and parts displayed as units may or may not be physical units, may be located in one position, or may be distributed on the multiple network units. Some or all of the modules can be selected based on actual demands to achieve the objectives of the solutions of the specification. A person of ordinary skill in the art can understand and implement the embodiments of the present application without creative efforts.

[0096] Referring again to FIG. 5, it can be interpreted as illustrating an internal functional module and a structure of a transaction execution apparatus. The transaction execution apparatus can be an example of a blockchain network node configured to perform parallel execution of smart contract transactions. The transaction execution apparatus can be an example of a blockchain network node configured to perform parallel execution of smart contract transactions. An execution body in essence can be an electronic device, and the electronic device includes the following: one or more processors; and a memory configured to store an executable instruction of the one or more processors.

[0097] The one or more processors are configured to receive multiple transactions; pre-execute each of the multiple transactions based on a first current state of a blockchain in the blockchain network before performing a consensus process of the plurality of transactions; and determine one or more accounts affected by the pre-executing the transaction; perform a consensus process of the multiple transactions and the one or more accounts affected by the pre-executing the transaction; divide the multiple transactions into one or more groups of transactions based on the one or more accounts affected by the pre-executing the transaction for each of the multiple transactions; execute the multiple transactions by executing the one or more groups of transactions in parallel based on a second current state of the blockchain in the blockchain network; determine one or more

accounts affected by the executing the each of the multiple transactions; determine if the one or more accounts affected by the executing the transaction are the same as one or more accounts affected by the pre-executing the transaction and if the one or more accounts affected by the executing the transaction are not affected by any previously executed transactions in the multiple transactions; and in response to determining that, for one of the multiple transactions, the one or more accounts affected by the executing the transaction are the same as the one or more accounts affected by the pre-executing the transaction and the one or more accounts affected by the executing the transaction are not affected by any previously executed transactions in the multiple transactions, commit the executing the transaction.

[0098] Optionally, the one or more processors are configured to, in response to determining that, for one of the multiple transactions, the one or more accounts affected by the executing the transaction are not the same as the one or more accounts affected by the pre-executing the transaction, or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the multiple transactions, roll back the executing the transaction; and re-execute the transaction after executing the one or more groups of transactions in parallel.

[0099] Optionally, the one or more processors are configured to, for each of the multiple transactions, recording the one or more accounts affected by the pre-executing the transaction for performing consensus of the one or more accounts affected by the pre-executing the transaction.

[00100] Optionally, the multiple transactions are executed by the network node in a same order as the multiple transactions are executed by any other network node of the multiple network nodes of the blockchain network.

[00101] Optionally, the one or more processors are configured to identify one or more failed transactions, wherein for each of the one or more failed transactions, one or more accounts affected by the executing the failed transaction are not the same as one or more accounts affected by the pre-executing the failed transaction or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the multiple transactions; and re-execute the one or more failed transactions after executing the one or more groups of transactions in parallel.

[00102] Optionally, the one or more processors are configured to, according to a protocol agreed by the multiple network nodes of the blockchain network: determine a sequential order of executing one or more transactions within each of the one or more groups; and determine a sequential order of executing the one or more failed transactions after executing the one or more groups of transactions in parallel.

[00103] Optionally, each group of transactions includes one or more transactions that affect one or more common transaction entities; and between each two different groups of transactions, any transaction in one group does not affect any common transaction entity with any transaction in another group.

[00104] Optionally, the common transaction entity includes a transferee, a transferor, an account of a transferee, or an account of a transferor associated with a transaction.

[00105] Optionally, each of the multiple transactions includes a smart contract transaction.

[00106] Optionally, each of the multiple transactions includes a transaction for which one or more accounts affected by executing the transaction is not deterministic before executing the transaction.

[00107] Optionally, pre-executing, by the network node, the transaction includes pre-executing the transaction, by the network node, when one or more processors of the network node is idle.

[00108] The techniques described in this specification produce one or more technical effects. For example, the specification discloses techniques that allow parallel execution of transactions by a network node in a distributed ledger system while guaranteeing the same execution order of the transactions performed by each network node of the distributed ledger system to ensure consistency of execution results of the transactions in the distributed ledger system. In some embodiments, smart contract transactions that can be executed in parallel are identified and grouped together, for example, based on pre-execution results of the smart contract transactions. In some embodiments, the technical effects and advantages are achieved, *inter alia*, by putting in different groups transactions that do not affect any common transaction entity or have dependency on one another (e.g., do not affect the same accounts in the blockchain network). Thus, the technique identifies groups of transactions can be executed in parallel with each other by a single

network node. In some embodiments, in the case of smart contract transactions, if the actual execution of one or more smart contract transactions affects different accounts than were identified by the pre-execution of the one or more smart contract transactions, the execution of one or more smart contract transactions is rolled back or undone, and then re-executed in serial after parallel execution of the rest smart contract transactions, thereby ensuring the correctness of the results at a modest computational cost relative to the benefit of generally parallel execution.

[00109] In some embodiments, the described techniques thus can improve the processing speed of the transactions and increase transaction throughput in a blockchain network. For example, pre-execution of the smart contract transactions can be done by a network node when one or more processors of the network node is idle, which make better use of the computational resources or processing power of the network node without introducing additional delay or latency. In some embodiments, by dividing the transactions into different groups after achieving a consensus by performing a consensus process and before executing the transactions, the multiple groups of transactions can be executed independently in parallel by taking advantage of a multi-processor or multi-core network node or the multiple computers in a cluster of computers to increase the execution speed of the network node and the efficiency of the overall blockchain network. In some embodiments, the described techniques do not require entry (e.g., manually) of a list of accounts of affected by the execution of the smart contract transactions, thus is not subject to possibility of entry errors or the unpredictability of affected accounts of certain smart contract transactions.

[00110] Described embodiments of the subject matter can include one or more features, alone or in combination.

[00111] For example, in a first embodiment, a method for executing the multiple transactions in a blockchain network, wherein the blockchain network includes the multiple network nodes, the method includes receiving, by a network node of the blockchain network that includes the multiple network nodes, the multiple transactions to be executed in a blockchain network; for each of the multiple transactions, pre-executing, by the network node, the transaction based on a first current state of a blockchain in the blockchain network before performing a consensus process of the plurality of

transactions; and determining, one or more accounts affected by the pre-executing the transaction; performing a consensus process of the multiple transactions and the one or more accounts affected by the pre-executing the transaction; dividing, by the network node, the multiple transactions into one or more groups of transactions based on the one or more accounts affected by the pre-executing the transaction for each of the multiple transactions; executing the multiple transactions by executing the one or more groups of transactions in parallel based on a second current state of the blockchain in the blockchain network; for each of the multiple transactions, determining one or more accounts affected by the executing the transaction; and determining if the one or more accounts affected by the executing the transaction are the same as one or more accounts affected by the pre-executing the transaction and if the one or more accounts affected by the executing the transaction are not affected by any previously executed transactions in the multiple transactions; and in response to determining that, for one of the multiple transactions, each of the one or more accounts affected by the executing the transaction are the same as the one or more accounts affected by the pre-executing the transaction and the one or more accounts affected by the executing the transaction are not affected by any previously executed transactions in the multiple transactions, committing the executing the transaction.

[00112] The foregoing and other described embodiments can each, optionally, include one or more of the following features:

[00113] A first feature, combinable with any of the following features, further including, in response to determining that, for one of the multiple transactions, the one or more accounts affected by the executing the transaction are not the same as the one or more accounts affected by the pre-executing the transaction, or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the multiple transactions, rolling back the executing the transaction; and re-executing the transaction after executing the one or more groups of transactions in parallel.

[00114] A second feature, combinable with any of the following features, further including: for each of the multiple transactions, recording the one or more accounts

affected by the pre-executing the transaction for performing consensus of the one or more accounts affected by the pre-executing the transaction.

5 [00115] A third feature, combinable with any of the following features, wherein the multiple transactions are executed by the network node in a same order as the multiple transactions are executed by any other network node of the multiple network nodes of the blockchain network.

[00116] A fourth feature, combinable with any of the following features, further including: identifying one or more failed transactions, wherein for each of the one or more failed transactions, one or more accounts affected by the executing the failed transaction are not the same as one or more accounts affected by the pre-executing the failed transaction or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the multiple transactions; and re-executing the one or more failed transactions after executing the one or more groups of transactions in parallel.

15 [00117] A fifth feature, combinable with any of the following features, further including, according to a protocol agreed by the multiple network nodes of the blockchain network: determining a sequential order of executing one or more transactions within each of the one or more groups; and determining a sequential order of executing the one or more failed transactions after executing the one or more groups of transactions in parallel.

20 [00118] A sixth feature, combinable with any of the following features, wherein: each group of transactions includes one or more transactions that affect one or more common transaction entities; and between each two different groups of transactions, any transaction in one group does not affect any common transaction entity with any transaction in another group.

[00119] A seventh feature, combinable with any of the following features, wherein the common transaction entity includes a transferee, a transferor, an account of a transferee, or an account of a transferor associated with a transaction.

30 [00120] An eighth feature, combinable with any of the following features, wherein each of the multiple transactions includes a smart contract transaction.

[00121] A ninth feature, combinable with any of the following features, wherein each of the multiple transactions includes a transaction for which one or more accounts affected by executing the transaction is not deterministic before executing the transaction.

[00122] A tenth feature, combinable with any of the following features, wherein pre-executing, by the network node, the transaction includes pre-executing the transaction, by the network node, when one or more processors of the network node is idle.

[00123] Embodiments of the subject matter and the actions and operations described in this specification can be implemented in digital electronic circuitry, in tangibly-embodied computer software or firmware, in computer hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer programs, e.g., one or more modules of computer program instructions, encoded on a computer program carrier, for execution by, or to control the operation of, data processing apparatus. For example, a computer program carrier can include one or more computer-readable storage media that have instructions encoded or stored thereon. The carrier may be a tangible non-transitory computer-readable medium, such as a magnetic, magneto optical, or optical disk, a solid state drive, a random access memory (RAM), a read-only memory (ROM), or other types of media. Alternatively, or in addition, the carrier may be an artificially generated propagated signal, e.g., a machine-generated electrical, optical, or electromagnetic signal that is generated to encode information for transmission to suitable receiver apparatus for execution by a data processing apparatus. The computer storage medium can be or be part of a machine-readable storage device, a machine-readable storage substrate, a random or serial access memory device, or a combination of one or more of them. A computer storage medium is not a propagated signal.

[00124] A computer program, which may also be referred to or described as a program, software, a software application, an app, a module, a software module, an engine, a script, or code, can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages; and it can be deployed in any form, including as a stand-alone program or as a module, component, engine, subroutine, or other unit suitable for executing in a computing environment, which

environment may include one or more computers interconnected by a data communication network in one or more locations.

[00125] A computer program may, but need not, correspond to a file in a file system. A computer program can be stored in a portion of a file that holds other programs or data, e.g., one or more scripts stored in a markup language document, in a single file dedicated to the program in question, or in the multiple coordinated files, e.g., files that store one or more modules, sub programs, or portions of code.

[00126] Processors for execution of a computer program include, by way of example, both general- and special-purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive the instructions of the computer program for execution as well as data from a non-transitory computer-readable medium coupled to the processor.

[00127] The term “data processing apparatus” encompasses all kinds of apparatuses, devices, and machines for processing data, including by way of example a programmable processor, a computer, or the multiple processors or computers. Data processing apparatus can include special-purpose logic circuitry, e.g., an FPGA (field programmable gate array), an ASIC (application specific integrated circuit), or a GPU (graphics processing unit). The apparatus can also include, in addition to hardware, code that creates an execution environment for computer programs, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

[00128] The processes and logic flows described in this specification can be performed by one or more computers or processors executing one or more computer programs to perform operations by operating on input data and generating output. The processes and logic flows can also be performed by special-purpose logic circuitry, e.g., an FPGA, an ASIC, or a GPU, or by a combination of special-purpose logic circuitry and one or more programmed computers.

[00129] Computers suitable for the execution of a computer program can be based on general or special-purpose microprocessors or both, or any other kind of central processing unit. Generally, a central processing unit will receive instructions and data from a read only memory or a random access memory or both. Elements of a computer

can include a central processing unit for executing instructions and one or more memory devices for storing instructions and data. The central processing unit and the memory can be supplemented by, or incorporated in, special-purpose logic circuitry.

[00130] Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to one or more storage devices. The storage devices can be, for example, magnetic, magneto optical, or optical disks, solid state drives, or any other type of non-transitory, computer-readable media. However, a computer need not have such devices. Thus, a computer may be coupled to one or more storage devices, such as, one or more memories, that are local and/or remote. For example, a computer can include one or more local memories that are integral components of the computer, or the computer can be coupled to one or more remote memories that are in a cloud network. Moreover, a computer can be embedded in another device, e.g., a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a Global Positioning System (GPS) receiver, or a portable storage device, e.g., a universal serial bus (USB) flash drive, to name just a few.

[00131] Components can be “coupled to” each other by being commutatively such as electrically or optically connected to one another, either directly or via one or more intermediate components. Components can also be “coupled to” each other if one of the components is integrated into the other. For example, a storage component that is integrated into a processor (e.g., an L2 cache component) is “coupled to” the processor.

[00132] To provide for interaction with a user, embodiments of the subject matter described in this specification can be implemented on, or configured to communicate with, a computer having a display device, e.g., a LCD (liquid crystal display) monitor, for displaying information to the user, and an input device by which the user can provide input to the computer, e.g., a keyboard and a pointing device, e.g., a mouse, a trackball or touchpad. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input. In addition, a computer can interact with a user by sending documents to and receiving documents from a device that is used by the user; for example, by sending web pages to a web browser on

a user's device in response to requests received from the web browser, or by interacting with an app running on a user device, e.g., a smartphone or electronic tablet. Also, a computer can interact with a user by sending text messages or other forms of message to a personal device, e.g., a smartphone that is running a messaging application, and
5 receiving responsive messages from the user in return.

[00133] This specification uses the term "configured to" in connection with systems, apparatus, and computer program components. For a system of one or more computers to be configured to perform particular operations or actions means that the system has installed on it software, firmware, hardware, or a combination of them that in operation
10 cause the system to perform the operations or actions. For one or more computer programs to be configured to perform particular operations or actions means that the one or more programs include instructions that, when executed by data processing apparatus, cause the apparatus to perform the operations or actions. For special-purpose logic circuitry to be configured to perform particular operations or actions means that the
15 circuitry has electronic logic that performs the operations or actions.

[00134] While this specification contains many specific embodiment details, these should not be construed as limitations on the scope of what is being claimed, which is defined by the claims themselves, but rather as descriptions of features that may be specific to particular embodiments. Certain features that are described in this
20 specification in the context of separate embodiments can also be realized in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiments can also be realized in the multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially be claimed as such, one or more features
25 from a claimed combination can in some cases be excised from the combination, and the claim may be directed to a subcombination or variation of a subcombination.

[00135] Similarly, while operations are depicted in the drawings and recited in the claims in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all
30 illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of

various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into the multiple software products.

[00136] Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some cases, multitasking and parallel processing may be advantageous.

What is claimed is:

CLAIMS

1. A computer-implemented method for executing a plurality of transactions in a blockchain network, wherein the blockchain network comprises a plurality of network nodes, the method comprising:

receiving, by a network node of the blockchain network, a plurality of transactions to be executed in a blockchain network;

for each of the plurality of transactions,

pre-executing, by the network node, the transaction based on a first current state of a blockchain in the blockchain network before performing a consensus process of the plurality of transactions; and

determining, one or more accounts affected by the pre-executing the transaction;

performing a consensus process of the plurality of transactions and the one or more accounts affected by the pre-executing the transaction;

dividing, by the network node, the plurality of transactions into one or more groups of transactions based on the one or more accounts affected by the pre-executing the transaction for each of the plurality of transactions;

executing the plurality of transactions by executing the one or more groups of transactions in parallel based on a second current state of the blockchain in the blockchain network;

for one of the plurality of transactions,

determining one or more accounts affected by the executing the transaction; and

in response to determining that the one or more accounts affected by the executing the transaction are the same as the one or more accounts affected by the pre-executing the transaction and are not affected by any previously executed transactions, committing the executing the transaction.

2. The method of claim 1, in response to determining that, for one of the plurality of transactions, the one or more accounts affected by the executing the transaction are not

the same as the one or more accounts affected by the pre-executing the transaction, or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the plurality of transactions,

rolling back the executing the transaction; and

5 re-executing the transaction after executing the one or more groups of transactions in parallel.

3. The method of any preceding claim, further comprising: for each of the plurality of transactions, recording the one or more accounts affected by the pre-executing the transaction for performing consensus of the one or more accounts affected by the pre-executing the transaction.

4. The method of any preceding claim, wherein the plurality of transactions are executed by the network node in the same order as the plurality of transactions are executed by any other network node of the plurality of network nodes of the blockchain network.

5. The method of any preceding claim, further comprising:

identifying one or more failed transactions, wherein for each of the one or more failed transactions, one or more accounts affected by the executing the failed transaction are not the same as one or more accounts affected by the pre-executing the failed transaction or the one or more accounts affected by the executing the transaction are affected by any previously executed transactions in the plurality of transactions; and

re-executing the one or more failed transactions after executing the one or more groups of transactions in parallel.

6. The method of claim 5, further comprising, according to a protocol agreed by the plurality of network nodes of the blockchain network:

determining a sequential order of executing one or more transactions within each of the one or more groups; and

determining a sequential order of executing the one or more failed transactions after executing the one or more groups of transactions in parallel.

7. The method of any preceding claim, wherein:

each group of transactions comprises one or more transactions that affect one or more common transaction entities; and

between each two different groups of transactions, any transaction in one group does not affect any common transaction entity with any transaction in another group.

8. The method of claim 7, wherein the one or more common transaction entities comprise one or more of a transferee, a transferor, an account of a transferee, or an account of a transferor associated with a transaction.

9. The method of any preceding claim, wherein each of the plurality of transactions comprises a smart contract transaction.

10. The method of any preceding claim, wherein each of the multiple transactions includes a transaction for which one or more accounts affected by executing the transaction is not deterministic before executing the transaction.

11. The method of any preceding claim, wherein pre-executing, by the network node, the transaction comprises pre-executing the transaction, by the network node, when one or more processors of the network node is idle.

12. A system, comprising:

one or more processors; and

one or more computer-readable memories coupled to the one or more processors and having instructions stored thereon which are executable by the one or more processors to perform the method of any of claims 1-11.

13. An apparatus for executing a plurality of transactions in a blockchain network, the apparatus comprising a plurality of modules for performing the method of any one of claims 1 to 11.