

# DATA PROCESSING METHOD AND APPARATUS BASED ON DISTRIBUTED STORAGE, DEVICE AND MEDIUM

[0001] This application claims priority to Chinese Patent Application No. 202110188344.X,  
5 entitled "DATA PROCESSING METHOD AND APPARATUS BASED ON DISTRIBUTED  
STORAGE, DEVICE AND MEDIUM" and filed with the National Intellectual Property  
Administration, PRC on February 19, 2021, which is incorporated herein by reference in its entirety.

## FIELD

[0002] This application relates to the field of data storage technologies, and in particular to a  
10 data storage technology in a block chain network.

## BACKGROUND

[0003] At present, in a block chain system, a local storage is generally used as a data storage  
medium for a consensus node participating in consensus to store data. For example, in the process of  
a consensus node A performing block consensus on a block N through a consensus layer, an executive  
15 layer may be invoked to execute each transaction in this block N. After block consensus is reached, a  
transaction execution result of each transaction is sent to a storage layer, such that block information  
(e.g., a block header, a transaction list and a transaction execution result) of this block N is written  
into the local storage through the storage layer. In general cases, the local storage and the storage  
layer in the consensus node are two independent hard disk spaces. Therefore, the data storage  
20 performed by using the local storage in the consensus node is limited by a size of the entire hard disk  
space, resulting in a limited amount of local storage.

[0004] In addition, in other block chain systems, a consensus node may also be allowed to  
use independent databases for data storage. However, for such a block chain system, the storage  
performance is usually degraded due to the limitation of the block chain structure of the block chain  
25 maintained by the block chain system when an independent database is used for data storage. For  
example, the storage layer can continue to notify the consensus layer to execute the next block only  
after successfully storing a block in the corresponding database, which means that it is required to  
store data serially according to a generation timestamp of each block during data storage, which  
inevitably reduces the storage efficiency of data in the block chain system.

## SUMMARY

**[0005]** A data processing method and apparatus based on distributed storage, a device and a medium are provided according to embodiments of this application, which can improve the data storage efficiency while increasing a data storage capacity.

5 **[0006]** In an aspect, a data processing method based on distributed storage is provided according to an embodiment of this application, which is executed by a block chain node and includes:

detecting, after writing first block data of a first block into a local cache and a local storage, a first block processing space of a storage buffer pool associated with the local cache to obtain a first space detection result, the storage buffer pool including N pieces of block data to be stored, where N  
10 is a non-negative integer;

writing, if the first space detection result indicates that the number of block data to be stored in the first block processing space does not reach a buffer block processing threshold of the first block processing space, the first block data into the storage buffer pool based on the N pieces of block data to be stored;

15 detecting a second block processing space of a storage processing pool associated with the storage buffer pool to obtain a second space detection result, the storage processing pool being configured to store M pieces of block data to be fed back associated with a distributed database, where M is a non-negative integer; and

determining, if the second space detection result indicates that a number of block data to be fed  
20 back in the second block processing space does not reach a feedback block processing threshold of the second block processing space, block data to be combined in the storage buffer pool based on the number of the block data to be fed back and the feedback block processing threshold, performing duplication detection on the block data to be combined and the M pieces of block data to be fed back, and writing the data retained after duplication detection into the distributed database.

25 **[0007]** In another aspect, a data processing apparatus based on distributed storage is provided according to an embodiment of this application. The apparatus includes: a first detection module, a first writing module, a second detection module, and a second writing module. The first detection module is configured to detect, after writing first block data of a first block into a local cache and a local storage, a first block processing space of a storage buffer pool associated with the local cache  
30 to obtain a first space detection result, the storage buffer pool includes N pieces of block data to be stored, where N is a non-negative integer. The first writing module is configured to write, if the first space detection result indicates that a number of block data to be stored in the first block processing

space does not reach a buffer block processing threshold of the first block processing space, the first block data into the storage buffer pool based on the N pieces of block data to be stored. The second detection module is configured to detect a second block processing space of a storage processing pool associated with the storage buffer pool to obtain a second space detection result, the storage processing pool is configured to store M pieces of block data to be fed back associated with a distributed database, where M is a non-negative integer. The second writing module is configured to determine, if the second space detection result indicates that a number of block data to be fed back in the second block processing space does not reach a feedback block processing threshold of the second block processing space, block data to be combined in the storage buffer pool based on the number of the block data to be fed back and the feedback block processing threshold, perform duplication detection on the block data to be combined and the M pieces of block data to be fed back, and write data retained after duplication detection into the distributed database.

**[0008]** In another aspect, a computer device is provided according to an embodiment of this application, which includes a memory and a processor. The memory is connected to the processor, the memory is configured to store a computer program, and the processor is configured to invoke the computer program to cause the computer device to perform the method according to the embodiment of this application.

**[0009]** In another aspect, a computer-readable storage medium is provided according to an embodiment of this application, which stores a computer program. The computer program is applicable to be loaded and executed by a processor, to cause a computer device including the processor to perform the method according to the embodiment of this application.

**[0010]** In another aspect, a computer program product or a computer program is provided according to an embodiment of this application. The computer program product or the computer program includes computer instructions, and the computer instructions are stored in a computer-readable storage medium. A processor of a computer device reads the computer instructions from the computer-readable storage medium and executes the computer instructions to cause the computer device to perform the method provided in this embodiment of this application.

**[0011]** According to this embodiment of this application, in the process of performing data storage by using a distributed database, first block data in a first block (that is, block data of a currently executed block) may be written into a local cache and a local storage (it is to be noted that the local cache and the local storage here are two storage spaces that are opened up in a storage layer of a block chain node). In this embodiment of this application, a first block processing space of a storage buffer pool associated with the local cache may be detected to obtain a first space detection result. The

storage buffer pool located in the storage layer includes N pieces of block data to be stored at present, where N is a non-negative integer. Further, if the first space detection result indicates that the number of block data to be stored in the first block processing space does not reach a buffer block processing threshold (e.g., 10) of the first block processing space, the block chain node may write the first block data into the storage buffer pool based on the N pieces of block data to be stored. Further, the block chain node may detect a second block processing space of a storage processing pool associated with the storage buffer pool to obtain a second space detection result, the storage processing pool is configured to store M pieces of block data to be fed back associated with a distributed database, where M is a non-negative integer. If the second space detection result indicates that the number of block data to be fed back in the second block processing space does not reach a feedback block processing threshold of the second block processing space, the block chain node may determine block data to be combined in the storage buffer pool based on the number of the block data to be fed back and the feedback block processing threshold, perform duplication detection on the block data to be combined and the M pieces of block data to be fed back, and write the data retained after duplication detection into the distributed database. It can be seen that, in the process of data storage in this embodiment of this application, the distributed database is used instead of the local storage in order to increase the data storage capacity. In addition, in order to improve the efficiency of data reading and writing in the process of performing data storage by using the distributed database, the storage layer is expanded in this embodiment of the application. That is, the local cache, the local storage, the storage buffer pool, the storage processing pool and the like are expanded in the storage layer to reconstruct a new storage layer. In this way, in the process of using this storage layer to obtain the data stored in the distributed database, not only parallel storage of multiple blocks can be realized, but also the limitation of a block chain structure maintained by the entire block chain system can be eliminated. In other words, the block chain node in this embodiment of this application can improve the data storage capacity and the data storage efficiency in the process of storing multiple blocks in parallel in the distributed database.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0012]** FIG. 1 is a schematic diagram of a network architecture according to an embodiment of this application;

**[0013]** FIG. 2 is a schematic diagram of a scenario of data storage based on distributed storage according to an embodiment of this application;

**[0014]** FIG. 3 is a schematic flowchart of a data processing method based on distributed storage according to this application;

**[0015]** FIG. 4 is a schematic diagram of a scenario in which the data processing method is performed on a first block and a second block according to an embodiment of this application;

**[0016]** FIG. 5 is a schematic flowchart of a data processing method based on distributed storage according to an embodiment of this application;

5 **[0017]** FIG. 6 is a schematic diagram of a scenario in which target data is read according to an embodiment of this application;

**[0018]** FIG. 7 is a schematic diagram of a scenario of a storage layer according to an embodiment of this application;

**[0019]** FIG. 8 is a schematic diagram of a scenario of a data packaging unit according to an  
10 embodiment of this application;

**[0020]** FIG. 9 is a diagram showing data interaction according to an embodiment of this application;

**[0021]** FIG. 10 is a schematic structural diagram of a data processing apparatus based on distributed storage according to this application; and

15 **[0022]** FIG. 11 is a schematic structural diagram of a computer device according to an embodiment of this application.

## DESCRIPTION OF EMBODIMENTS

**[0023]** The technical solutions in the embodiments of this application are clearly and completely described below with reference to the drawings in the embodiments of this application.  
20 Apparently, the described embodiments are merely some rather than all of the embodiments of this application. All other embodiments obtained by a person of ordinary skill in the art based on the embodiments of this application without creative efforts shall fall within the protection scope of this application.

**[0024]** Referring to FIG.1, FIG. 1 is a schematic diagram of a network architecture according  
25 to an embodiment of this application. The network architecture shown in FIG. 1 may be applied to a block chain system. The block chain system may be a distributed system formed by connecting multiple nodes through network communication. The block chain system may include, but is not limited to, a block chain system corresponding to an alliance chain.

**[0025]** The block chain is an application of computer technologies such as distributed data  
30 storage, point-to-point transmission, a consensus mechanism, and an encryption algorithm, which is mainly used to organize data in chronological order and encrypt it into a ledger, so that the data will

not be tampered and forged, in addition, the block chain is also used to verify, store and update data. The block chain is essentially a decentralized database, each node in the database stores an identical block chain, and a block chain network includes core nodes and light nodes, where the core nodes may be responsible for the consensus of the entire block chain network, that is, the core nodes may be consensus nodes in the block chain network. The process of writing transaction data into a ledger (e.g., a local ledger) in the block chain network may be as follows: a client sends the transaction data to a light node, and then the transaction data is passed between the light nodes in the block chain network in a baton form until the consensus node receives the transaction data; the consensus node then packages the transaction data into a block, in order to reach consensus with other consensus nodes in the future; and after the consensus is reached, multiple blocks carrying the transaction data may be written into the distributed database in parallel through the storage layer, which can break through the limitations of the block chain structure of the block chain from the root, thereby effectively improving the storage efficiency of data storage.

**[0026]** The block chain system may include a smart contract. In the block chain system, the smart contract may be understood as a code executed by each node (including the consensus node) of the block chain, and any logic may be executed through the smart contract to obtain a result. For example, a user may initiate a transaction service request through the client to call a smart contract that has been deployed on the block chain. Subsequently, a data node or a light node on the block chain may transmit the transaction service request to each consensus node to call the smart contract running in each consensus node to execute a transaction service requested by the user. It is to be understood that the block chain may include one or more smart contracts, and these smart contracts may be distinguished by contract calling addresses, contract identify documents (ID) or contract names. However, in the transaction service request initiated by the client, a contract calling address or a contract identify document or a contract name of the smart contract may be carried to specify the smart contract that needs to be run. If the smart contract specified by the client is a contract that is used to read data (that is, a service contract), each consensus node may read data by accessing the local cache created in the storage layer. At last, consensus nodes will verify between each other whether transaction execution results are consistent (that is, whether a consensus is reached), and if so, the transaction execution results may be stored in the respective local caches and local storages, and returned to the client. It is to be noted that the local cache here is a system memory created in the storage layer; and the local storage here is a hard disk space created in the storage layer for data storage. In this way, a case of failing to read data can be avoided, which is caused due to data lost in the system memory when a consensus node is down or the system fails. That is, the consensus node can read data through the local storage created in the storage layer.

**[0027]** It is to be understood that the network architecture shown in FIG. 1 may include a core node (i.e., a consensus node) cluster, a light node cluster, and a user terminal cluster. The core node cluster may include multiple core nodes, and the multiple core nodes here may specifically include a node 10a, a node 10b, a node 10c, and a node 10d shown in FIG. 1. As shown in FIG. 1, the node 10a, the node 10b, the node 10c and the node 10d may be connected through a network to form a consensus network 100a shown in FIG. 1. It can be understood that, in the consensus network 100a, the node 10a, the node 10b, the node 10c and the node 10d may all perform data interaction through network connection. In addition, the user terminal cluster may include multiple user terminals, and the multiple user terminals here may specifically include a user terminal 3000a, a user terminal 3000b, ..., and a user terminal 3000c shown in FIG. 1. As shown in FIG. 1, the user terminal 3000a, the user terminal 3000b, ..., and the user terminal 3000c may respectively perform network connection with the node 10c, so as to perform data interaction through the network connection with the node 10c. In addition, the light node cluster may include multiple light nodes, and the multiple light nodes here may specifically include a server 4000a and a server 4000b shown in FIG. 1. As shown in FIG. 1, the server 4000a and the server 4000b may be connected to the node 10c via a network, respectively, so as to perform data interaction through the network connection with the node 10c.

**[0028]** In this embodiment of this application, each core node (e.g., the node 10a, the node 10b, the node 10c, and the node 10d) in the consensus network 100a may be referred to as a block chain node. It is to be understood that these block chain nodes may be configured to maintain the same block chain (e.g., a block chain 10e shown in FIG. 1). A peer-to-peer (P2P) network may be formed between any two block chain nodes in the consensus network 100a. The peer-to-peer network may use a P2P protocol, which is an application layer protocol running over a transmission control protocol (TCP). In a distributed system, any device such as server or a terminal may join the block chain network to serve as a block chain node. Each block chain node may include a hardware layer, an intermediate layer, an operating system layer and an application layer.

**[0029]** In this embodiment of this application, a block chain node may be bound to any one (e.g., any individual user, any enterprise, any institution and other entity objects) accessing the block chain network, and the block chain network formed by these block chain nodes may be referred to as an alliance chain network. Therefore, there is a one-to-one correspondence between the nodes 10a, 10b, 10c, and 10d shown in FIG. 1 and the corresponding one (i.e., an entity object in the corresponding service scenario) that needs to be connected to the alliance chain network. The service scenario here may include an electronic bill scenario, a social scenario, a credit purchase scenario, a credit scenario, or the like. In this case, a target service under the corresponding service scenario may specifically include an electronic bill service, a social service, a credit purchase service, a credit

service, etc., and the specific services under the corresponding service scenarios are not listed one by one here.

**[0030]** Since each entity object may correspond to a block chain node, the enterprise user (that is, the above-mentioned enterprise) may be used as the entity object in this embodiment of the application. In this case, the block chain node associated with each enterprise user may be the same block chain node (for example, the above-mentioned node 10c shown in FIG. 1 may perform data interaction with respective user terminals corresponding to multiple enterprise users). For example, in an electronic bill system of the block chain, an electronic bill service (such as a registration service, a billing service, or a bill transfer service) corresponding to each billing enterprise may be referred to as a transaction service. A billing enterprise A may perform data interaction with the node 10c shown in FIG. 1 through the user terminal 3000a shown in FIG. 1 to complete a corresponding transaction; by analogy, a billing enterprise B may perform data interaction with the node 10c shown in FIG. 1 through the user terminal 3000b shown in FIG. 1 to complete a corresponding transaction; and a billing enterprise C may perform data interaction with the node 10c shown in FIG. 1 through the user terminal 3000c shown in FIG. 1 to complete a corresponding transaction.

**[0031]** In this embodiment of this application, the above entity objects (e.g., the billing enterprise A, the billing enterprise B and the billing enterprise C) that transmit transaction service requests for electronic bill service may be referred to as target users; the block chain nodes that receive transaction service requests sent by the target users may be referred to as light nodes; and the block chain nodes participating in the consensus on the transaction service request may be referred to as core nodes.

**[0032]** In an embodiment, in the consensus network 100a, since the node 10c may synchronize data with other block chain nodes that have network connections (also referred to as session connections), that is, the node 10c may synchronize corresponding service data information from other block chain nodes (for example, can synchronize transaction service requests sent by other enterprise users from other block chain nodes), the core nodes associated with respective enterprise users may be different block chain nodes. For example, the billing enterprise A may perform data interaction with the node 10c shown in FIG. 1 through the user terminal 3000a shown in FIG. 1. The billing enterprise B may perform data interaction with the node 10b shown in FIG. 1 through the user terminal 3000b shown in FIG. 1. The billing enterprise C may perform data interaction with the node 10a shown in FIG. 1 through the user terminal 3000c shown in FIG. 1. Since these nodes in the block chain network maintain the same block chain, a network load in the block chain network can be effectively balanced by randomly allocating the transaction service requests sent by different user

terminals to the block chain nodes associated with the block chain 10e, thereby improving the processing efficiency of the service data corresponding to the corresponding service.

**[0033]** In another example, in a crediting system, the credit service requested by the target user (for example, a user corresponding to the user terminal 3000a) may be understood as another transaction service. The target user may perform data interaction with the node 10c shown in FIG. 1 through the user terminal 3000a shown in FIG. 1, to transmit a transaction service request for the corresponding service to the node 10c shown in FIG. 1. That is, in the crediting system, one block chain node may be configured for each user accessing the crediting system in order to receive a transaction service request sent by the corresponding user. It is to be understood that each user in this embodiment of this application may also be referred to as an entity object, for example, the entity object requesting the aforementioned credit service may be the aforementioned individual user, enterprise user, or the like.

**[0034]** The light node may forward, on reception of the transaction service request, the transaction service request initiated by the target user to the core node, to verify, through the core node, the legality of the transaction service request initiated by the target user. The core node may add the transaction service requested by the first user to a transaction pool in a case that the transaction service request passes the legality verification, so that transaction data associated with the transaction service can be subsequently packaged into a block. Therefore, the consensus nodes in the consensus network 100a perform block consensus, and write, after the block consensus is reached, block data of the block into the local cache and the local storage, thereby facilitating subsequent parallel storage of block data of multiple blocks based on the above distributed storage.

**[0035]** For ease of understanding, further referring to Figure 2, FIG. 2 is a schematic diagram of a scenario of data storage based on distributed storage according to an embodiment of this application. A block chain node shown in FIG. 2 may be any one of the core nodes in the embodiment corresponding to FIG. 1. The block chain node may perform parallel storage of multiple blocks through a consensus layer 100, an executive layer 200, a storage layer 300 and a distributed storage 400.

**[0036]** The main functions of the consensus layer 100 may include a block packaging function, a block broadcasting function and a block consensus function. It is to be understood that in the process of performing block consensus through the block consensus function, the consensus layer 100 may call the executive layer 200 shown in FIG. 2 to execute respective transaction services in the blocks, to perform transaction verification on respective transaction services in a block for which consensus is to be reached currently, to determine to reach a block consensus of the block 1 in a case that the

transaction verification is successful. In this case, as shown in FIG. 2, the block chain node may refer to a transaction execution result of each transaction service executed by the executive layer 200 as block data of the block 1, and may transmit the block data of the block 1 to the storage layer shown in FIG. 2.

5     **[0037]**         In this embodiment of this application, blocks N currently received by the executive layer 200 and transmitted by the consensus layer 100 may be referred to as first blocks, and block data of the first blocks that has been executed by the executive layer 200 may be referred to as first block data of the first block. It is to be understood that the first block data in the first block may include, but is not limited to, the transaction execution result of each transaction service in the first  
10    block. For example, the first block data in the first block may also include block header information (i.e., first block header information) of the first block. In an embodiment, the number of transaction services in the first block may be K1. In this case, the first block data in the first block may further include a transaction list associated with the K1 transaction services, where K1 may be a positive integer.

15    **[0038]**         After executing a certain transaction in the block, the executive layer 200 may call a smart contract in a contract virtual machine (referred to as a virtual machine) to execute a corresponding transaction service, and may transmit each transaction execution result to the storage layer 300 shown in FIG. 3 in a case that each transaction in the block is executed.

20    **[0039]**         The storage layer 300 may be responsible for storing block information (e.g., a block header, a transaction list, an execution result and other block data) of the first block into the distributed storage 400 shown in FIG. 2. The distributed storage 400, as a data storage medium, may be integrated on other distributed servers independently of the block chain node. As shown in FIG. 2, the storage layer 300 may include a disaster recovery module 310, a local cache 320 and a persistence module 330 shown in FIG. 2. The local cache 320 here may be a local cache module created in the block  
25    chain node. The local cache 320 may be a system memory of the block chain node.

30    **[0040]**         As shown in FIG. 2, the disaster recovery module 310 may be located in a storage hard disk of the block chain node. The local storage 312 shown in FIG. 2 may be configured to store (i.e., back up) the same block data as the local cache 320. In this way, in a case that the block chain node is down or the system fails, the execution of other transaction service in the block is not affected by the block data lost in the system memory. In this case, the block chain node may use the disaster recovery module 310 to read target data required to execute other transaction service in the block from the storage hard disk of this node. The local storage 312 may be a local storage unit with a persistence capability in the disaster recovery module 310.

**[0041]** As shown in FIG. 2, the block chain node may notify the consensus layer 100 shown in FIG. 2 to process a next block of the first block when writing the first block data of the first block to the local storage 312 and the local cache 320 shown in FIG. 2. It can be understood that, in this embodiment of this application, a next block of the first block may be referred to as a new first block.

5 In order to facilitate distinguishing between blocks processed by the consensus layer 100 at different times, in this embodiment of this application, a block (i.e., the aforementioned new first block) processed at the next time from the current time may be referred to as a second block, and block data of the second block may be referred to as second block data.

**[0042]** As shown in FIG. 2, the persistence module 330 in the storage layer 300 may include  
10 a storage buffer pool 331 and a storage processing pool 332 shown in FIG. 2. The storage buffer pool 331 may be configured to store the first block data of the executed first block sent by the executive layer 200. The first block of data may be combined with data (e.g., N pieces of block data to be stored) currently stored in the storage buffer pool 331, thereby reducing the waste of a storage space of the storage buffer pool 331 caused by repeated storage of the same data. The size of a block processing  
15 space (i.e., a first block processing space) of the storage buffer pool 331 may be adaptively configured according to actual service requirements.

**[0043]** In addition, in this embodiment of this application, after writing the first block data of the first block into the local cache 320 and the local storage 312, the block chain node may detect the first block processing space of the storage buffer pool through a first refreshing thread, and allow to  
20 write the first block data into the storage buffer pool 331 based on the aforementioned N pieces of block data to be stored if it is determined according to a space detection result (i.e., a first space detection result) that the storage buffer pool 331 has a space for processing the block data of the current block. It is to be understood that, by configuring the space size of the storage buffer pool 331, the problem of data read errors caused by fast block execution can be fundamentally solved. For  
25 example, in a case that the block is executed fast due to improper configuration of the space size, the block chain node may clear data in the local cache 320 that has not been stored in the distributed database 400 (e.g., some data that has been stored for a long time), resulting in data read errors. Therefore, the block chain node may perform buffer refreshing on the first block processing space of the storage buffer pool 331 through the first refreshing thread after writing the first block data into  
30 the storage buffer pool 331. In this way, in a case that the refreshed first block processing space has space for processing the above-mentioned next block, the block chain node may allow further reception of the second block data of the executed second block transmitted by the executive layer 200.

**[0044]** Further, the data in the storage buffer pool 331 may be referred to as data to be stored after the block chain node writes the first block data into the storage buffer pool 331. In this way, if it is determined according to a space detection result (i.e., a second space detection result) that the storage processing pool 332 has space to store data after the block chain node detects a block processing space (i.e., a second block processing space) of the storage processing pool 332 shown in FIG. 2 through a second refreshing thread, a corresponding amount of block data (e.g., one or more block data) is acquired from the storage buffer pool 331 as block data to be combined, and duplication detection is performed on the acquired block data to be combined and the M pieces of block data to be fed back in the storage processing pool 332, so that data obtained after duplication detection is written into the distributed database of the distributed storage 400 shown in FIG. 2. By configuring the space size of the storage processing pool 332, parallel storage of multiple blocks can be performed without excessively occupying the memory.

**[0045]** A specific process of the block chain node writing the first block data into the storage buffer pool and storing multiple blocks in parallel may refer to the embodiments corresponding to FIGS. 3 to 9 below.

**[0046]** Further, referring to FIG.3, FIG. 3 is a schematic flowchart of a data processing method based on distributed storage according to this application. As shown in FIG. 3, the method may be executed by the block chain node in the above-mentioned block chain network. For example, the block chain node may be any core node in the consensus network 100a shown in FIG. 1. The method may specifically include the following steps S101-S104.

**[0047]** In step S101, after writing first block data of a first block into a local cache and a local storage, a first block processing space of a storage buffer pool associated with the local cache is detected to obtain a first space detection result.

**[0048]** The storage buffer pool includes N pieces of block data to be stored, where N is a non-negative integer and is used to represent the number of blocks to be stored in the first block processing space.

**[0049]** The block chain node may acquire K1 transaction services in the first block on reception of a first block transmitted by a consensus layer (e.g., the consensus layer 100 shown in FIG. 2), and call a service contract of each transaction service to execute a corresponding transaction service. The first block may be a block packaged at the current time (such as a time T1) based on a block packaging function of the consensus layer, and the block packaged at the time T1 may be the above-mentioned block 1.

**[0050]** The block chain node may, after executing the block 1 (i.e., the first block) through an executive layer (e.g., the executive layer 200 shown in FIG. 2), write the first block data of the block 1 into a local cache (e.g., the local cache 320 shown in FIG. 2) and a local storage (e.g., the local storage 312 shown in FIG. 2). The block chain node may immediately notify the consensus layer to process the next block after writing the first block data into the local cache and the local storage. Further, the consensus layer may acquire K2 transaction services from a transaction pool for packaging based on the block packaging function, to generate a new block (i.e., the block 2) for which consensus is to be reached. In this embodiment of this application, blocks packaged by the consensus layer at a time T2 may be referred to as a second block. The number (i.e., K2) of transaction services in the second block and the number (i.e., K1) of transaction services in the first block may be the same or different. The specific values of K1 and K2 will not be limited here. The time T1 and the time T2 are two different packaging times, and the time T2 follows the time T1.

**[0051]** For ease of understanding, further referring to FIG.4, FIG. 4 is a schematic diagram of a scenario in which the above operations are performed on the first block and the second block according to an embodiment of this application. The first block shown in FIG. 4 may be the block 1 to be verified that is packaged by the consensus layer at the time T1, and the block 1 to be verified may include K1 transaction services, specifically including a transaction 11, a transaction 12, a transaction 13, ..., a transaction 1k shown in FIG. 4. It is to be understood that an executive layer H of the block chain node may perform, after acquiring the K1 transaction services, transaction verification on each of the K1 transaction services, and obtain a transaction execution result of each transaction service if the transaction verification is successful. As shown in FIG. 4, the transaction execution result of each transaction service in the block 1 may be a first transaction execution result shown in FIG. 4. In the first transaction execution result, a result 21 may be a transaction execution result corresponding to the transaction 11, the result 22 may be a transaction execution result corresponding to the transaction 12, the result 23 may be a transaction execution result corresponding to the transaction 13, and by analogy, the result 2k may be a transaction execution result corresponding to the transaction 1k.

**[0052]** The block chain node may generate, after performing the transaction verification, verification success indication information. The consensus layer of the block chain node may determine that a block consensus on the first block has been reached after receiving the verification success indication information transmitted by more than half of the other consensus nodes, such that the first block data of the first block on which consensus has been reached (i.e., the block 1 shown in FIG. 4) can be written to the local cache and local storage. The block chain node may asynchronously detect, after writing the first block data of the first block into the local cache and the local storage, a

block processing space (i.e., the first block processing space) of a storage buffer pool (e.g., the storage buffer pool 331 shown in FIG. 2) through the first refreshing thread, to obtain a first space detection result. The first space detection result may be used to represent whether the storage buffer pool has space to process the current block at the current time (e.g., time T1', where the time T1' may be the time after the above-mentioned time T1, such as the above-mentioned time T2).

**[0053]** In a case that the block chain node counts, through the first refreshing thread, that the number of block data to be stored currently stored in the storage buffer pool is N, it means that the number of blocks to be stored that may be buffered by the first block processing space of the storage buffer pool (i.e., the number of blocks to be stored) is N. For example, if the first space detection result indicates that the number of block data to be stored in the first block processing space (e.g., N=6) does not reach a buffer block processing threshold of the first block processing space (e.g., N<sub>max</sub>= 10), it means that the storage buffer pool has space for processing the current block at the current time (e.g., the above-mentioned T1' time), and the following step S102 may be performed.

**[0054]** In an embodiment, if the first space detection result indicates that the number of block data to be stored in the first block processing space (e.g., N=10) has reached the buffer block processing threshold (e.g., N<sub>max</sub>= 10), the executive layer may be notified to wait for the first block of data to be stored, which means that the storage buffer pool has no space to process the current block at this time. Therefore, the executive layer needs to wait until the storage buffer pool has space to process the current block, and the following step S102 may be performed.

**[0055]** In step S102, if the first space detection result indicates that the number of block data to be stored in the first block processing space does not reach the buffer block processing threshold of the first block processing space, the first block data is written into the storage buffer pool based on the N pieces of block data to be stored.

**[0056]** Specifically, if the first space detection result indicates that the number of block data to be stored in the first block processing space does not reach the buffer block processing threshold of the first block processing space, the block chain node may acquire N pieces of block data to be stored from the storage buffer pool, and search the N pieces of block data to be stored for the block data to be stored that duplicates with the first block data. If the block data to be stored that duplicates with the first block data is found from the N pieces of block data to be stored, the block chain node may use the found block data to be stored as duplicated block data, and the block chain node may replace the duplicated block data with the first block data, use the N pieces of block data to be stored that is obtained after replacement as first combined data associated with the first block data, and write the first combined data into the storage buffer pool.

**[0057]** It is to be noted that in the process of writing the first block data into the storage buffer pool, the block chain node needs to combine N pieces of block data to be stored that is currently stored in the storage buffer pool, so that repeated storage of the same data can be reduced.

**[0058]** For example, in the process of writing the first block data into the storage buffer pool, in a case that the system fails before the block chain node has time to write the block data of the entire block into the storage buffer pool, the storage buffer pool may store part of the block data of the first block. In this way, after the block chain node restarts, it can re-receive the block 1 (i.e., the first block) transmitted by the executive layer. In this case, the block chain node may read target data for executing the K1 transaction services in the block 1 from the local storage of the storage layer, and obtain transaction execution results of the K1 transaction services. In a case of writing the regenerated first block data into the storage buffer pool, the current N pieces of block data to be stored may be searched for block data to be stored that duplicates with the first block data. Further, the block chain node may use the found block data to be stored (i.e., the part of block data of the first block) as duplicated block data, and replace the duplicated block data with the first block data in the N pieces of block data to be stored, such that the replaced N pieces of block data to be stored may be used as first combined data, and the first combined data may be written into the storage buffer pool. It can be seen that, in the event of a downtime or a system failure, the block chain node may replace part of block data of the first block buffered during the system failure or downtime with the regenerated first block data in the N pieces of block data to be stored, thereby avoiding repeated writing of the same data.

**[0059]** In an embodiment, if no block data to be stored that duplicates with the first block data is found from the N pieces of block data to be stored, the block chain node may use the first block data and the N pieces of block data to be stored as second combined data associated with the first block data, write the second combined data into the storage buffer pool, and perform incremental processing on the number of block data to be stored in the first block processing space, where the number of block data to be stored in the first block processing space after incremental processing is N+1.

**[0060]** In an embodiment, the block chain node may further receive block data of the next block transmitted by the executive layer when writing the first block data into the storage buffer pool, the block data of the next block here mainly referring to the second block data of the second block (e.g., the block 2 shown in FIG. 4), and perform buffer refreshing on the first block processing space of the storage buffer pool through the first refreshing thread to obtain a buffered refreshing result (for example, at this time, the first refreshing thread may count that the number of the stored block data to be stored in the storage buffer pool at a next time (for example, a time T2', that is, the time T2' is

the next time of the time  $T1'$ ) is  $N+1$ ). In this case, the block chain node may update the first space detection result by using the buffer refreshing result and determine whether the storage buffer pool has space to process the next block at the next time (for example, the time  $T2'$ ) based on the updated first space detection result. If the updated first space detection result indicates that the number of block data to be stored in the first block processing space has not reached the buffer block processing threshold, the second block data may be added to the storage buffer pool where the first block data is located at the next time.

**[0061]** The second block shown in FIG. 4 may be a block 2 to be verified that is packaged by the consensus layer at the time  $T2$ , and the block 2 to be verified may include  $K2$  transaction services, specifically including a transaction 31, a transaction 32, a transaction 33, ..., a transaction  $3n$  shown in FIG. 4. It is to be noted that an executive layer of the block chain node may perform, after acquiring the  $K2$  transaction services, transaction verification on each of the  $K2$  transaction service, and obtain a transaction execution result of each transaction service if the transaction verification is successful. The transaction execution result of each transaction service may be a second transaction execution result shown in FIG. 4. In the second transaction execution result, a result 41 may be a transaction execution result corresponding to the transaction 31, the result 42 may be a transaction execution result corresponding to the transaction 32, the result 43 may be a transaction execution result corresponding to the transaction 33, and by analogy, the result  $4n$  may be a transaction execution result corresponding to the transaction  $3n$ .

**[0062]** The block chain node may generate, after performing the transaction verification of the  $K2$  transaction services, verification success indication information. In this way, the consensus layer of the block chain node may determine, on reception of new verification success indication information transmitted by more than half of the other consensus nodes, that a block consensus on the second block has been reached, such that the second block data of the second block that has reached consensus (i.e., the block 2 shown in FIG. 4) may be written to the local cache and the local storage. Therefore, the second block data of the second block can be quickly written to the storage buffer pool in a case that the storage buffer pool has space to process the next block upon subsequent detection.

**[0063]** In step S103, a second block processing space of the storage processing pool associated with the storage buffer pool is detected to obtain a second space detection result.

**[0064]** The storage processing pool is configured to store  $M$  pieces of block data to be fed back associated with a distributed database, where  $M$  is a non-negative integer and is used to represent the number of blocks to be fed back in the second block processing space.

**[0065]** After performing step S103, the first combined data or the second combined data in the storage buffer pool may be referred to as the data to be stored, and the block chain node may detect the second block processing space of the storage processing pool associated with the storage buffer pool through a second refreshing thread to obtain the second space detection result. If the second space detection result indicates that the number of block data to be fed back in the second block processing space (e.g.,  $M=10$ ) reaches a feedback block processing threshold (e.g.,  $M_{\max}=10$ ), the block chain node may notify the storage buffer pool to wait for the storage of the data to be stored until the second space detection result detected through the second refreshing thread indicates that the number of data to be fed back in the second block processing space has not reached the feedback block processing threshold. In this way, the following step S104 may be performed in a case that the block chain node determines that the number of data to be fed back has not reached the feedback block processing threshold of the storage processing pool according to the second space detection result.

**[0066]** The following step S104 may be performed directly after the block chain node performs step S103, that is, the block chain node determines the number (e.g.,  $M=7$ ) of data to be fed back in the second block processing space as indicated by the second space detection result has not reached the feedback block processing threshold (e.g.,  $M_{\max}=10$ ) after detecting the second block processing space of the storage processing pool associated with the storage buffer pool through the second refreshing thread.

**[0067]** In step S104, if the second space detection result indicates that the number of block data to be fed back in the second block processing space does not reach a feedback block processing threshold of the second block processing space, block data to be combined in the storage buffer pool is determined based on the number of the block data to be fed back and the feedback block processing threshold, duplication detection is performed on the block data to be combined and the  $M$  pieces of block data to be fed back, and the data retained after duplication detection is written into the distributed database.

**[0068]** Specifically, in a case that the data to be stored in the storage buffer pool includes the  $N$  pieces of block data to be stored and the first block data, and the first block data has no data duplication with the  $N$  pieces of block data to be stored, the block chain node may determine, if the second space detection result indicates that the number of block data to be fed back in the second block processing space does not reach the feedback block processing threshold of the second block processing space, a number difference between the number of the block data to be fed back and the feedback block processing threshold, determine block data to be combined from the  $N$  pieces of block

data to be stored and the first block data based on the number difference and a block height of the data to be stored in the storage buffer pool, and perform duplication detection on the block data to be combined and the M pieces of block data to be fed back to obtain an duplication detection result. Further, the block chain node may write, if the duplication detection result indicates that the M pieces of block data to be fed back has no block data to be fed back that duplicates with the block data to be combined, the block data to be combined into the storage processing pool, use the block data to be combined in the storage processing pool and the M pieces of block data to be fed back as data retained after duplication detection, and write the data retained after duplication detection into the distributed database. The process of writing multiple blocks data into the distributed database by the block chain node is executed in parallel, which can eliminate the limitations of the existing block chain structure of block storage, thereby improving the data storage efficiency.

**[0069]** In an embodiment, in the process of writing the data retained after duplication detection into the distributed database, the block chain node may continuously receive storage completion feedback information of the corresponding data in the block data returned by the distributed database. Based on this, the block chain node receiving the storage completion feedback information of the corresponding data is indirectly reflected in that data successfully written into the distributed database exists in the data retained after the duplication detection. In this case, the block chain node may refer to the block data to which the data successfully written into the distributed database belongs to target block data, and receive a data completion amount of the target block data fed back by the distributed database. Then, the block chain node may perform decremental processing, based on the data completion amount, on the amount of data to be stored corresponding to the target block data in a block storage state mapping table until the amount of data to be stored reaches a decrement threshold (e.g., 0), and determine that a complete block corresponding to the target block data has been successfully written into the distributed database.

**[0070]** The M pieces of block data to be fed back stored in the storage processing pool are block data that are currently being stored in the distributed database and for which the stored feedback information returned by the distributed database have not been received. The data retained after the duplication detection may include multiple pieces of block data. In this way, the block chain node can realize parallel storage of multiple blocks through the storage buffer pool and the storage processing pool. Based on this, in the process of parallel storage of the block data in the storage processing pool to the distributed database, the block chain node may receive the storage completion feedback information of the corresponding data in each block data returned by the distributed database. In this case, the block chain node may perform decremental processing, based on the received storage completion feedback information of the corresponding data (i.e., the data completion amount of the

target block data), on the amount of data to be stored at a block height (i.e., a height of the block) of the block corresponding to the target block data in a block storage state mapping table in the storage layer, until the amount of data to be stored in the block height of the corresponding block reaches a decrement threshold (e.g., 0). In this case, the block chain node may determine that the complete  
5 block with the block height corresponding to the target block data has been successfully written to the distributed database.

**[0071]** It can be seen that, in the process of data storage in this embodiment of this application, the previous local storage is replaced with the distributed storage in order to increase the data storage capacity. In addition, in order to improve the efficiency of reading and writing data in the process of  
10 using the distributed storage for data storage, the storage layer is expanded in this embodiment of the application. For example, the local cache, the local storage, the storage buffer pool, the storage processing pool and the like are expanded in the storage layer and to reconstruct a new storage layer. In this way, the process of using this storage layer to obtain the data stored in the distributed database can not only realize parallel storage of multiple blocks, but also break through the limitation of a  
15 block chain structure maintained by the entire block chain system from the root. In other words, the block chain node (i.e., a consensus node participating in the consensus) in this embodiment of this application can increase the data storage capacity and improve the data storage efficiency in the process of storing multiple blocks in parallel in the distributed database.

**[0072]** Further, FIG. 5 is a schematic flowchart of a data processing method based on  
20 distributed storage according to an embodiment of this application. As shown in FIG. 5, the method may be executed by a block chain node, which may be any core node (e.g., the node 10c) in the consensus network 100a shown in FIG. 1. The method may specifically include the following steps S201-S210.

**[0073]** In step S201, on reception of a first block transmitted by a consensus layer, the  
25 executive layer acquires K1 transaction services in the first block.

**[0074]** The first block includes first block header information and a transaction list associated with the K1 transaction services, K1 is a positive integer.

**[0075]** In step S202, for each of the K1 transaction services, a service contract for executing the transaction service is acquired, and the transaction service is executed through the service contract  
30 of the transaction service to obtain a transaction execution result of the transaction service.

**[0076]** Specifically, for each of the K1 transaction services, the block chain node may acquire, based on a contract calling address of the transaction service, a service contract for executing the

transaction service. Further, the block chain node may search, through the service contract of the transaction service, target data associated with the transaction service from the local cache. The block chain node may use, if the target data associated with the transaction service is found from the local cache, the found target data in the local cache as target read data of the transaction service, and execute the transaction service based on the target read data of this transaction service to obtain a transaction execution result of this transaction service.

**[0077]** For ease of understanding, further referring to FIG.6, FIG. 6 is a schematic diagram of a scenario in which target data is read according to an embodiment of this application. The block chain node may acquire, when executing a certain transaction service (e.g., transaction service Tx1) in the first block, target data for executing this transaction service Tx1 from a local database (e.g., a database 1 corresponding to the local cache, the database 1 being a system database) as shown in FIG. 6 through a contract data reading method of a smart contract (i.e., a service contract of the transaction service Tx1) that runs on a virtual machine. The target data here may be configured adaptively according to the above-mentioned service scenarios, which is not limited here. For example, in the process of asset transfer, the target data here may be an account balance of a target user who requests the asset transfer.

**[0078]** In a case that the block chain node is down or a system failure occurs in the process of executing the next transaction service (e.g., transaction service Tx2) of the transaction service Tx1 in the first block, the block chain node may re-receive the aforementioned first block through a data recovery thread when being restarted, so as to continually execute the transaction service in the first block based on data (i.e., backup data) stored in the local storage. In this case, the block chain node may preferentially read data that needs to execute the corresponding transaction service from a local database in FIG. 6 (e.g., a database 2 corresponding to local storage, the database 2 being a hard disk database). Under normal circumstances, the system database may be regarded as a primary storage database in the node, and the hard disk database may be regarded as a standby storage database in the node. It can be seen that, in embodiment of this application, the disaster recovery capability can be improved in the event of a failure through primary-standby switching in the local database of the node during the execution of the transaction service. In addition, in this embodiment of this application, by using the local cache locally in the node, the network overhead caused by the distributed database can be effectively reduced during the transaction execution process.

**[0079]** It can be seen that, if the block chain node finds the target data associated with each transaction service from the local cache, the block chain node may use the found target data in the local cache as the target read data of each transaction service. Further, the block chain node may

execute, based on the target read data of each transaction service, the transaction service to obtain a transaction execution result of the transaction service.

**[0080]** In an embodiment, as shown in FIG. 6, the block chain node may also search, through the service contract, the distributed database shown in FIG. 6 for the target data associated with the transaction service if the block chain node does not find the target data associated with the transaction service from the local cache, and further use the found target data in the distributed database as target read data of the transaction service.

**[0081]** In this embodiment of this application, a data reading level of the local database corresponding to the local cache is higher than a data reading level of the distributed database shown in FIG. 6. The distributed database as shown in FIG. 6 may include a primary database and a standby database. By configuring a one-primary and multiple-standby mode for the distributed database, more storage devices may be used to support service scenarios with massive data, which can greatly enhance the availability of the distributed database while increasing the data storage capacity.

**[0082]** The block chain node may create a local cache in a local storage layer of the node. In this way, the created local cache may be used to cache block data of the executed block transmitted by the executive layer (e.g., the block data of the executed block may be block data of an executed block A).

**[0083]** Further, in a case that the consensus layer of the block chain node submits the next block (e.g., the next block of the block A is a block B), the block B to be executed may be referred to as the above-mentioned first block, and K1 transaction services in the first block may be executed in the executive layer. For example, the executive layer may preferentially read, when executing the first block, the target data for executing the corresponding transaction service from the block data of the executed block cached in the local cache, and use the target data read in the local cache as target read data of the corresponding transaction service, and execute the corresponding transaction service according to the target read data of the corresponding transaction service, to obtain a transaction execution result of the corresponding transaction service. In an embodiment, if the executive layer of the block chain node reads (i.e., hitting) no target data for executing the transaction service at the current time in the local cache when executing a transaction service, the executive layer of the block chain node may search target data associated with this transaction service from the distributed database through a network, and use the found target data in the distributed database as the target read data of this transaction service.

**[0084]** Then, the storage layer of the block chain node may refer to block header information (i.e., first block header information) of the first block, the transaction list of the first block and the

transaction execution result of each transaction service in the first block, which are transmitted by the executive layer, as first block data of the first block, and use the first block data of the first block as the block data of the block currently executed by the executive layer. Then, the storage layer of the block chain node may write the block data of the currently executed block (i.e., the first block data of the first block) to the local cache and the local storage.

**[0085]** For ease of understanding, further referring to FIG.7, FIG. 7 is a schematic diagram of a scenario of a storage layer according to an embodiment of this application. A storage layer 700 shown in FIG. 7 may be the storage layer 300 in the embodiment corresponding to FIG. 2. In the implementation of replacing the local storage in the related art with a distributed database, in order to solve the problem of poor read and write efficiency in the distributed storage, in this embodiment of this application, it is proposed to create multiple modules in the storage layer 700 shown in FIG. 7. The multiple modules here may specifically include a disaster recovery module 710, a local cache module 720, and a persistence module 730 shown in FIG. 7. The disaster recovery module 710 may have the same function as the disaster recovery module 310 in the embodiment corresponding to FIG. 2, the local cache module 720 may have the same function as the local cache module 320 in the embodiment corresponding to FIG. 2, and the persistence module 730 may have the same function as the persistence module 330 in the embodiment corresponding to FIG. 2.

**[0086]** The disaster recovery module 710 may specifically include a data packaging unit 711, a local storage unit 712, a block storage state mapping table unit 713, a version refreshing thread 714, a data recovery thread 716 and a persistence completion block height unit 715 as shown in FIG. 7.

**[0087]** A disaster recovery processing flow of the disaster recovery module 710 may be described as follows.

**[0088]** The data packaging unit 711 may serialize all data to be stored (i.e., n1 pieces of data) in the current block (i.e., the above-mentioned first block) into one piece of data (e.g., the currently serialized data may be data V1), use a block height (e.g., H1) of the current block (i.e., the first block) as a key in a KV key-value pair and use a processing result (i.e., the serialized data) of the current block by the data packaging unit 731 as a Value, and store the KV key-value pair (i.e., block height of the current block, currently serialized data) into the local storage unit 712 shown in FIG. 7. For example, (block height of the current block, currently serialized data) = (H1, V1). It is to be noted that, in this embodiment of this application, all the n1 pieces of data to be stored in the current block (i.e., the above-mentioned first block) may be referred to as an amount of data to be stored in the first block having a block height H1. For example, the n1 pieces of data may include, but is not limited to, the transaction execution results of the K1 transaction services.

**[0089]** The block storage state mapping table unit 713 may be configured to store the amount of data to be stored in the corresponding block, and may be configured to indicate that the block chain node has completed the distributed storage of this block in a case that the amount of data to be stored in the corresponding block is 0. For example, in a case that the storage processing pool (e.g., the storage processing pool 732 shown in FIG. 7) has block data (i.e., the target block data) to which the data successfully written into the distributed database belongs in the data retained after the duplication detection, the data storage capacity corresponding to the target block data can be decremented in a storage state mapping table stored in the block storage state mapping table unit 713, to update the amount of data to be stored in the storage state mapping table in real time.

**[0090]** The persistence completion block height unit 715 may be configured to store block data of continuous and complete blocks in the current distributed database. For example, if the blocks that have been stored in the distributed database include: blocks 0 to 100, a block 111, a block 112, and a block 113, the persistence completion block height unit 715 may store the block 100 (in this case, a block height of the block 100 may be H). Since the data between the block 100 and the block 111 is not continuous, the persistence completion block height unit 715 may periodically query whether a block height H+1 in the block storage state mapping table maintained by the block storage state mapping table unit 713 shown in FIG. 7 has been stored. If the storage has been completed, the persistence completion block height unit 715 may update the persistence completion block height maintained by itself as  $H=H+1$  (that is, at this time, H may be updated as  $100+1=101$ ). In addition, the block chain node may call the version refresh threading 714 shown in FIG. 7 to store the block height H (e.g., the block height of the previously stored block 100) into the distributed database.

**[0091]** The version refreshing thread 714 here may store the block height H in the persistence completion block height unit 715 shown in FIG. 7 into the above-mentioned distributed database. The reason why the block height H is to be stored in the distributed database is that the local node is likely to damage, such as system failure, or hard disk damage. Therefore, in a case that the block chain node is used subsequently, data in a block corresponding to the corresponding block height may be acquired from other consensus nodes for completion.

**[0092]** The data recovery thread 716 may compare the latest block height in the local storage unit 712 shown in FIG. 7 with the latest block height in the distributed database (i.e., the block height stored by the version refreshing thread) as the node restarts. If it is found that a remotely distributed database has block missing (that is, the block heights are inconsistent, for example, the latest block height of the local storage unit 712 is higher than the latest block height in the distributed database), the block chain node may acquire the missing block height data from the local storage unit shown in

FIG. 7, and rewrite the missing block height data to the distributed database in an order of block height from high to low. In this way, after the data recovery is completed, the corresponding data in the local storage unit 712 shown in FIG. 7 may be deleted to prevent excessive consumption of a local storage space corresponding to the local storage unit 712.

5     **[0093]**         The local storage unit 712 has the same function as the local storage 312 in the above-mentioned embodiment shown in FIG. 2. For example, the local storage unit 712 may be configured to store complete disaster recovery data (i.e., to store complete backup data). Specifically, the disaster recovery data in the local storage unit 712 may be the block data of the current executed block transmitted by the executive layer.

10    **[0094]**         For ease of understanding, further referring to FIG.8, FIG. 8 is a schematic diagram of a scenario of a data packaging unit according to an embodiment of this application. A data packaging unit 811 shown in FIG. 8 may be the data packaging unit 711 for performing data packaging in the embodiment shown in FIG. 7.

**[0095]**         In a case that the current block executed by the executive layer is a block 1  
15    corresponding to the block height H1 shown in FIG. 8, the block 1 may be referred to as a first block. In this case, as shown in FIG. 8, after the executive layer executes the block 1, the data packaging unit 811 may serialize all pieces of data to be stored (e.g., a block header 11, a transaction set 12 and a block write data set 13 shown in FIG. 8) in the current block (i.e., the above-mentioned first block) into one piece of data (e.g., the currently serialized data may be a block data set 1 shown in FIG. 8),  
20    use a block height (e.g., a block height H1 shown in FIG. 8) of the current block (i.e., the first block) as a key in a KV key-value pair and use a processing result (i.e., the serialized block data set 1) of the current block by the data packaging unit 831 as a Value, and store the KV key-value pair (i.e., the block height of the current block, the currently serialized data) into a local storage unit 812 shown in FIG. 8. For example, in this case, (the block height of the current block, the currently serialized data)  
25    = (block height H1, block data set 1). It is to be noted that all the n1 pieces of data to be stored in the current block (i.e., the first block) may specifically include a block header 11, a transaction set 12 and a block write data set 13 shown in FIG. 8.

**[0096]**         Further, as shown in FIG. 8, in a case that the current block executed by the executive layer is a block 2 corresponding to a block height H2 shown in FIG. 8, the block 2 may be referred to  
30    as a new first block. The new first block may be a second block. In this case, as shown in FIG. 8, after the executive layer executes the block 2 shown in FIG. 8, the data packaging unit 811 may serialize all pieces of data to be stored (e.g., a block header 21, a transaction set 22 and a block write data set 23 shown in FIG. 8) in the current block (i.e., the new first block) into one data (e.g., the currently

serialized data may be a block data set 2 shown in FIG. 8), use a block height (e.g., a block height H2 shown in FIG. 8) of the current block (i.e., the new first block) as a key in a KV key-value pair and use a processing result (i.e., the serialized block data set 2) of the current block by the data packaging unit 831 as a Value, and store the KV key-value pair (i.e., the block height of the current block, the currently serialized data) into the local storage unit 812 shown in FIG. 8. For example, in this case, (the block height of the current block, the currently serialized data) = (block height H2, block data set 2). All the n1 pieces of data to be stored in the current block (i.e., the new first block) may specifically include a block header 21, a transaction set 22 and a block write data set 23 shown in FIG. 8.

**[0097]** It can be seen that the data packaging unit 811 may be responsible for packaging all the data of each block into one block data set, and use the block height (i.e., a height of the block) as a key and use a block data set as a value, and store them into the local storage unit 812 shown in FIG. 8. In this way, in a case that the system is down and restarted, the block chain node may compare the latest block height stored in the local storage unit 812 shown in FIG. 8 with the latest block height stored in the distributed database, to complete data of the block corresponding to the corresponding block height that is lacking in a remote distributed database. The block chain node may acquire the latest block height of the distributed database from the above-mentioned distributed database through a getLastBlockHeight interface (i.e., a general block chain interface).

**[0098]** The reason why the block chain node needs to write the first block data of the first block to the local storage is that the local cache here is a storage unit in a system memory of the block chain node, and this local storage may be a storage unit in a storage hard disk of the block chain node. In a case that the block chain node is down or restarts due to a system failure, a case that no target data can be directly read based on the restarted local cache due to the lost content in the local cache will be prevented. Therefore, by arranging the disaster recovery module, in a case that the block chain node fails, primary-standby switching in the node can be intelligently realized. Therefore, in the process of reading the target data, the disaster recovery capability can be improved through a backup local storage, so that the block data of the currently executed block can be quickly and accurately written to the distributed database after the first block is executed subsequently.

**[0099]** In step S203, the first block header information, the transaction list, and the transaction execution results of the respective transaction services are used as first block data of the first block.

**[0100]** In step S204, the first block data of the first block is written into the local cache, and the first block data of the first block is written into the local storage corresponding to the local cache.

**[0101]** The following step S205 may be executed after the block chain node executes step S204. In addition, the block chain node may further notify the executive layer to execute K2

transaction services in a second block after writing the first block data into the local cache and the local storage. The second block may be obtained by packaging the K2 transaction services by a consensus layer, K2 here may be a positive integer. It is to be noted that the second block here is the next block of the aforementioned first block.

5     **[0102]**         It can be seen that the block chain node in the embodiments of this application may support read-write separation, so that an access pressure on the database (herein referred to as the distributed database) can be reduced. That is, after executing the first block, the block chain node may write the first block data of the first block into the distributed database, and search (i.e., read) the target data associated with each of the K2 transaction services from the local cache while executing  
10   the second block, thereby greatly advancing the time of processing the next block and increasing the throughput of a block chain maintained by the block chain node.

**[0103]**         The specific implementation of steps S201 to S204 may refer to the description of the specific process of writing the first block data to the local cache and the local storage in the embodiment shown in FIG. 3, which will not be repeated here.

15   **[0104]**         In step S205, after writing first block data of a first block into a local cache and a local storage, a first block processing space of a storage buffer pool associated with the local cache is detected to obtain a first space detection result.

**[0105]**         The storage buffer pool includes N pieces of block data to be stored, where N is a non-negative integer and is used to represent the number of blocks to be stored in the first block processing  
20   space.

**[0106]**         Before writing the first block data into the storage buffer pool, the block chain node needs to determine whether the storage buffer pool has space to process the next block through the first refreshing thread. That is, the block chain node may determine whether the number of blocks to be stored in the first block processing space of the storage buffer pool reaches a buffer block  
25   processing threshold by using the first refreshing thread. In a case that the block chain node determines that the number of blocks to be stored has reached the buffer block processing threshold, the block chain node may notify the executive layer to wait for the first block of data to be stored. In a case that the block chain node determines that the number of blocks to be stored has not reached the buffer block processing threshold, the block chain node may execute the following step S206.

30   **[0107]**         In step S206, if the first space detection result indicates that the number of block data to be stored in the first block processing space does not reach the buffer block processing threshold of the first block processing space, the first block data is written into the storage buffer pool based on

the N pieces of block data to be stored.

**[0108]** Specifically, if the block chain node determines, after executing step S205, that a first space detection result indicates that the number of block data to be stored in the first block processing space has not reached the buffer block processing threshold of the first block processing space, the block chain node may acquire N pieces of block data to be stored from the storage buffer pool, and search block data to be stored that duplicates with the first block data from the N pieces of block data to be stored. Further, if the block data to be stored that duplicates with the first block data is found from the N pieces of block data to be stored, the block chain node may use the found block data to be stored as duplicated block data. Further, the block chain node may replace the duplicated block data with the first block data in the N pieces of block data to be stored, use the replaced N pieces of block data to be stored as first combined data associated with the first block data, and write the first combined data into the storage buffer pool. The block chain node may use, if no block data to be stored that duplicates with the first block data is found from the N pieces of block data to be stored, the first block data and the N pieces of block data to be stored as second combined data associated with the first block data, write the second combined data into the storage buffer pool, and perform incremental processing on the number of block data to be stored in the first block processing space in the storage buffer pool. The number of block data to be stored in the first block processing space after incremental processing is N+1.

**[0109]** It can be seen that, in this embodiment of this application, the block chain node needs to perform combination processing in the process of writing the first block data into the storage buffer pool, that is, determine whether the first block data has data duplication with the N pieces of block data to be stored in the storage buffer pool. In the presence of data duplication, data needs to be combined. That is, the block data to be stored with data duplication in the N pieces of block data to be stored may be replaced with the first block data, and the first combined data may be written into the storage buffer pool, thereby avoiding the phenomenon of wasting storage space due to repeated storage of the same data (i.e., two data having data duplication) in the subsequent distributed database. In the absence of data duplication, the first block data and the N pieces of block data to be stored may be combined, so as to write the combined second combined data into the storage buffer pool. In this embodiment of this application, the repeated block data is combined by using a combination processing method, which can reduce the writing amount of data.

**[0110]** In an embodiment, after the block chain node executes step S206, that is, when the block chain node writes the first block data into the storage buffer pool, the block chain node may receive second block data of the second block transmitted by the executive layer. The second block

data is obtained by the executive layer through executing the K2 transaction services. Further, the block chain node may perform, through the first refreshing thread, buffer refreshing on the first block processing space of the storage buffer pool to obtain a buffer refreshing result, and update the first space detection result by using the buffer refreshing result. Further, the block chain node may add, if  
5 the updated first space detection result indicates that the number of block data to be stored in the first block processing space does not reach a buffer block processing threshold of the storage buffer pool, the second block data to the storage buffer pool where the first block data is located.

**[0111]** It is to be noted that, in this embodiment of this application, the block can be prevented from executing too fast as much as possible by arranging the storage buffer pool in the persistence  
10 module of the storage layer. That is, the storage buffer pool in this storage layer may notify the local cache to eliminate some data that has been cached for a long time after the data is successfully stored in the distributed database. In this way, the problem that the local cache eliminates data that has not been stored in the distributed database due to the fast execution of the block can be solved fundamentally, thereby avoiding the phenomenon that the data read by the executive layer is  
15 erroneous.

**[0112]** In step S207, a second block processing space of the storage processing pool associated with the storage buffer pool is detected to obtain a second space detection result.

**[0113]** The storage processing pool is used to store M pieces of block data to be fed back associated with a distributed database, where M is a non-negative integer and is used to represent the  
20 number of blocks to be fed back in the second block processing space.

**[0114]** In this embodiment of this application, in the process of performing the above steps S205 to S206, the block chain node may also perform steps S207 to S208 asynchronously. For example, in the process of writing the first block data to the storage buffer pool, the block chain node may asynchronously perform space detection on the second block processing space of the storage  
25 processing pool to obtain a second space detection result.

**[0115]** In a case that the second space detection result indicates that the number of block data to be fed back in the second block processing space reaches a feedback block processing threshold of the second block processing space, it is determined that the storage processing pool currently has no space to hold data. In this case, in order to prevent writing too much data to the storage processing  
30 pool at one time, the block chain node needs to notify the storage buffer pool to store and wait for the data to be stored. That is, the block chain node does not need to instruct the storage processing pool to acquire the block data to be combined from the storage buffer pool.

**[0116]** In addition, in a case that the second space detection result indicates that the number of block data to be fed back in the second block processing space has not reached a feedback block processing threshold of the second block processing space, it is determined that the storage processing pool currently has space to hold data. In this case, the block chain node may execute the following step S208 to acquire the block data to be combined from the storage buffer pool.

**[0117]** In step S208, if the second space detection result indicates that a number of block data to be fed back in the second block processing space does not reach a feedback block processing threshold of the second block processing space, block data to be combined in the storage buffer pool is determined based on the number of the block data to be fed back and the feedback block processing threshold, duplication detection is performed on the block data to be combined and the M pieces of block data to be fed back, and the data retained after duplication detection is written into the distributed database.

**[0118]** Specifically, the data in the storage buffer pool includes N pieces of block data to be stored and the first block data, and the block chain node may determine, if the second space detection result indicates that the number of block data to be fed back in the second block processing space does not reach the feedback block processing threshold of the second block processing space, a number difference between the number of the block data to be fed back in the second block processing space and the feedback block processing threshold in a case that the first block data has no data duplication with the N pieces of block data to be stored, determine block data to be combined from the N pieces of block data to be stored and the first block data based on the number difference and a block height of the data to be stored in the storage buffer pool, and perform duplication detection on the block data to be combined and the M pieces of block data to be fed back to obtain an duplication detection result. In this case, if the block chain node determines that the duplication detection result indicates that the M pieces of block data to be fed back has no block data to be fed back that duplicates with the block data to be combined, the block chain node may write the block data to be combined into the storage processing pool, use the block data to be combined in the storage processing pool and the M pieces of block data to be fed back as data retained after duplication detection, and write the data retained after duplication detection into the distributed database. If the block chain node determines that the duplication detection result indicates that the M pieces of block data to be fed back has no block data to be fed back that duplicates with the block data to be combined, the block chain node needs to overwrite, with the data to be combined acquired from the storage buffer pool, the block data to be fed back that has an duplication with the data to be combined, so as to prevent multiple writing of repeated data.

**[0119]** In step S209, a data completion amount of the target block data fed back by the distributed database is received.

**[0120]** The target block data is block data to which data successfully written into the distributed database belongs in the data retained after the duplication detection.

5 **[0121]** In step S210, based on the data completion amount, decremental processing is performed on the amount of data to be stored corresponding to the target block data in a block storage state mapping table until the amount of data to be stored reaches a decrement threshold, it is determined that a complete block corresponding to the target block data has been successfully written into the distributed database.

10 **[0122]** The specific implementation of steps S205 to S210 may refer to the description of the specific process of storing multiple block data into the distributed database in the embodiment shown in FIG. 3, which will not be repeated here.

**[0123]** For ease of understanding, further referring to FIG.9, FIG. 9 is a data interaction diagram according to an embodiment of this application. An executive layer 200 shown in FIG. 9 may be the executive layer 200 in the embodiment corresponding to FIG. 2. In a case that the block chain node executes a step S11 shown in FIG. 9, the storage layer in the block chain node may receive the first block data transmitted by the executive layer 200 after the first block is executed, and store the received first block data to the local storage and the local cache in this storage layer. The block chain node may further perform, after storing the first block data to the local storage and the local cache, step S12 shown in FIG. 9 to determine whether the storage buffer pool 331 has space to process block data of the current block (i.e., the first block data), and perform step S13 shown in FIG. 9 if the storage buffer pool 331 has space to process block data of the current block. On the contrary, the block chain node turns to execute step S11 shown in FIG. 9 if the storage buffer pool 331 has no space to process block data of the current block. That is, the block chain node needs to wait for the storage buffer pool to have space to process the first block data, and then write the first block data to the storage buffer pool 331 shown in FIG. 9. Further, as shown in FIG. 9, in the process of performing the above steps S12 and S13, the block chain node may asynchronously perform steps S14 and S15 shown in FIG. 9. That is, the block chain node may asynchronously detect whether the storage processing pool 332 shown in FIG. 9 has space to save data, and perform duplication detection on the data acquired from the storage buffer pool 331 (i.e., the above-mentioned block data to be combined) and the M pieces of block data to be fed back currently stored in the storage processing pool if the storage processing pool 332 has space to save data to obtain the above-mentioned duplication detection result. In case of determining, based on the duplication detection result, that the M pieces

of block data to be fed back has no block data to be fed back that duplicates with the block data to be combined, the block chain node may write the block data to be combined into the storage processing pool, and then use the block data to be combined in the storage processing pool and the M pieces of block data to be fed back as data retained after duplication detection, for executing a step S16 shown in FIG. 9. On the contrary, the block chain node may perform step S17 shown in FIG. 9. For example, in case of determining, based on the duplication detection result, that the M pieces of block data to be fed back has block data to be fed back that duplicates with the block data to be combined, the block chain node may overwrite the data (i.e., old data) in the storage processing pool 332 with the data (i.e., new data) acquired from the storage buffer pool 331, thereby limiting the writing number of times of data, thus ensuring the accuracy of the subsequent transaction execution results of the block.

**[0124]** It can be seen that, in this embodiment of this application, parallel storage can be achieved in the storage processing pool by using the distributed storage, which can effectively break through the block chain structure during block storage, thereby greatly improving the data storage efficiency while increasing the data storage capacity. In addition, in the process of executing the transaction service in the block by the executive layer, an access pressure to the distributed database can be effectively reduced by using the local cache, and the network overhead caused by the use of the distributed database can be reduced under certain circumstances. Furthermore, by performing primary/standby switching in the storage layer, the disaster recovery capability of the block chain node can be effectively improved in a case that the node fails.

**[0125]** Further referring to FIG.10, FIG. 10 is a schematic structural diagram of a data processing apparatus based on distributed storage according to this application. As shown in FIG. 10, the data processing apparatus 1 based on distributed storage may be applied to any block chain node in the above consensus network. For example, the data processing apparatus 1 based on distributed storage may be applied to the node 10c in the embodiment shown in FIG. 1. The data processing apparatus 1 based on distributed storage may be a computer program (including a program code) running in a block chain node (e.g., the aforementioned node 10c). For example, the data processing apparatus 1 based on distributed storage may be application software. It is to be noted that the data processing apparatus 1 based on distributed storage may be configured to execute corresponding steps in the methods in the embodiments of this application. As shown in FIG. 10, the data processing apparatus 1 based on distributed storage may include: a first detection module 11, a first writing module 12, a second detection module 13 and a second writing module 14. In an embodiment, the data processing apparatus 1 based on distributed storage may further include: a transaction acquisition module 15, a transaction execution module 16, a first block determination module 17, a local storage module 18, a notification block execution module 19, a second block determination module 20, a

buffer refreshing module 21, a block data addition module 22, an execution waiting module 23, a to-be-stored data determination module 24, a buffer waiting module 25, a completion amount receiving module 26 and a data amount decrement module 27.

**[0126]** The first detection module 11 is configured to detect, after writing first block data of a first block into a local cache and a local storage, a first block processing space of a storage buffer pool associated with the local cache to obtain a first space detection result, the storage buffer pool includes N pieces of block data to be stored, where N is a non-negative integer.

**[0127]** The first writing module 12 is configured to write, if the first space detection result indicates that a number of block data to be stored in the first block processing space does not reach a buffer block processing threshold of the first block processing space, the first block data into the storage buffer pool based on the N pieces of block data to be stored.

**[0128]** The first writing module 12 includes: a to-be-stored block acquisition unit 121, a first searching and determination unit 122, an duplication replacement unit 123 and a second searching and determination unit 124.

**[0129]** The to-be-stored block acquisition unit 121 is configured to acquire, if the first space detection result indicates that a number of block data to be stored in a first block processing space does not reach a buffer block processing threshold of the first block processing space, N pieces of block data to be stored from the storage buffer pool, and search the N pieces of block data to be stored for block data to be stored that has data duplication with the first block data.

**[0130]** The first searching and determination unit 122 is configured to use, if the block data to be stored that has data duplication with the first block data is found from the N pieces of block data to be stored, the found block data to be stored as duplicated block data.

**[0131]** The duplication replacement unit 123 is configured to replace the duplicated block data with the first block data, use the replaced N pieces of block data to be stored as first combined data associated with the first block data, and write the first combined data into the storage buffer pool.

**[0132]** In an embodiment, the second searching and determination unit 124 is configured to use, if the block data to be stored that has data duplication with the first block data is not found from the N pieces of block data to be stored, the first block data and the N pieces of block data to be stored as second combined data associated with the first block data, write the second combined data into the storage buffer pool, and perform incremental processing on the number of block data to be stored in the first block processing space.

**[0133]** The specific implementations of the to-be-stored block acquisition unit 121, the first

searching and determination unit 122, the duplication replacement unit 123 and the second searching and determination unit 124 may refer to the description of step S102 in the embodiment shown in FIG. 3, which will not be repeated here.

**[0134]** The second detection module 13 is configured to detect a second block processing space of a storage processing pool associated with the storage buffer pool to obtain a second space detection result. The storage processing pool is configured to store M pieces of block data to be fed back associated with a distributed database, where M is a non-negative integer.

**[0135]** The second writing module 14 is configured to determine, if the second space detection result indicates that a number of block data to be fed back in the second block processing space does not reach a feedback block processing threshold of the second block processing space, block data to be combined in the storage buffer pool based on the number of the block data to be fed back and the feedback block processing threshold, perform duplication detection on the block data to be combined and the M pieces of block data to be fed back, and write the data retained after duplication detection into the distributed database.

**[0136]** The data to be stored in the storage buffer pool includes the N pieces of block data to be stored and the first block data, and the first block data has no data duplication with the N pieces of block data to be stored.

**[0137]** The second writing module 14 includes: a difference determination unit 141, an duplication detection unit 142, a first writing unit 143 and a second writing unit 144.

**[0138]** The difference determination unit 141 is configured to determine, if the second space detection result indicates that the number of blocks to be fed back does not reach a feedback block processing threshold of the second block processing space, a number difference between the number of block data to be fed back and the feedback block processing threshold.

**[0139]** The duplication detection unit 142 is configured to determine the block data to be combined from the N pieces of block data to be stored and the first block data based on the number difference and a block height of the data to be stored in the storage buffer pool, and perform duplication detection on the data to be combined and the M pieces of block data to be fed back to obtain an duplication detection result.

**[0140]** The first writing unit 143 is configured to write the block data to be combined into the storage processing pool if the duplication detection result indicates that the M pieces of block data to be fed back has no data to be fed back that duplicates with the block data to be combined.

**[0141]** The second writing unit 144 is configured to use the block data to be combined and

the M pieces of block data to be fed back in the storage processing pool as data retained after duplication detection, and write the data retained after duplication detection into the distributed database.

**[0142]** The specific implementations of the difference determination unit 141, the duplication detection unit 142, the first writing unit 143 and the second writing unit 144 may refer to the description of step S104 in the embodiment shown in FIG. 3, which will not be repeated here.

**[0143]** In an embodiment, the transaction acquisition module 15 is configured to acquire K1 transaction services in the first block in a case that an executive layer receives the first block transmitted by a consensus layer. The first block includes first block header information and a transaction list associated with the K1 transaction services, where K1 is a positive integer.

**[0144]** The transaction execution module 16 is configured to acquire, for each of the K1 transaction services, a service contract for executing the transaction service, and execute the transaction service through the service contract of the transaction service to obtain a transaction execution result of the transaction service.

**[0145]** The transaction execution module 16 includes: a contract acquisition unit 161, a contract calling unit 162, a local searching unit 163, a transaction execution unit 164 and a distributed searching unit 165.

**[0146]** The contract acquisition unit 161 is configured to acquire, based on a contract calling address of the transaction service, a service contract for executing the transaction service.

**[0147]** The contract calling unit 162 is configured to search, through the service contract of the transaction service, target data associated with the transaction service from the local cache.

**[0148]** The local searching unit 163 is configured to use the target data found from the local cache as target read data of the transaction service if the target data associated with the transaction service is found from the local cache.

**[0149]** The transaction execution unit 164 is configured to execute, based on the target read data of the transaction service, the transaction service to obtain a transaction execution result of the transaction service.

**[0150]** In an embodiment, the distributed searching unit 165 is configured to search, through the service contract, the target data associated with the transaction service from the distributed database if the target data associated with the transaction service is not found from the local cache, and use the target data found from the distributed database as the target read data of the transaction service.

**[0151]** The specific implementations of the contract acquisition unit 161, the contract calling unit 162, the local searching unit 163, the transaction execution unit 164 and the distributed searching unit 165 may refer to the description of each transaction service in the embodiment shown in FIG. 3, which will not be repeated here.

5 **[0152]** The first block determination module 17 is configured to use the first block header information, the transaction list, and the respective transaction execution results of the K1 transaction services as first block data of the first block.

**[0153]** The local storage module 18 is configured to write the first block data of the first block into the local cache, and write the first block data of the first block into the local storage corresponding  
10 to the local cache.

**[0154]** In an embodiment, the notification block execution module 19 is configured to notify the executive layer to execute K2 transaction services in a second block after writing the first block data into the local cache and the local storage. The second block is a next block of the first block, and the second block is obtained by packaging the K2 transaction services by a consensus layer, where  
15 K2 is a positive integer.

**[0155]** The second block determination module 20 is configured to receive second block data of the second block transmitted by the executive layer after writing the first block data into the storage buffer pool. The second block data is obtained by the executive layer after executing the K2 transaction services.

20 **[0156]** The buffer refreshing module 21 is configured to perform, through a first refreshing thread, buffer refreshing on the first block processing space of the storage buffer pool to obtain a buffer refreshing result, and update the first space detection result by using the buffer refreshing result.

**[0157]** The block data addition module 22 is configured to write, if the updated first space detection result indicates that the number of block data to be stored in the first block processing space  
25 does not reach the buffer block processing threshold, the second block data into the storage buffer pool based on the block data to be stored in the first block processing space.

**[0158]** In an embodiment, the execution waiting module 23 is configured to notify, before writing the first block data into the storage buffer pool, the executive layer to wait for the first block data to be stored if the first space detection result indicates that the number of block data to be stored  
30 in the first block processing space has reached the buffer block processing threshold.

**[0159]** In an embodiment, the to-be-stored data determination module 24 is configured to use the combined data in the storage buffer pool as data to be stored.

**[0160]** The buffer waiting module 25 is configured to notify the storage buffer pool to wait for the storage of the data to be stored if the second space detection result indicates that the number of block data to be fed back in the second block processing space has reached a feedback block processing threshold.

5 **[0161]** In an embodiment, the completion amount receiving module 26 is configured to receive a data completion amount of the target block data fed back by the distributed database. The target block data is block data to which data successfully written into the distributed database belongs in the data retained after the duplication detection.

**[0162]** The data amount decrement module 27 is configured to perform, based on the data  
10 completion amount, decremental processing on the amount of data to be stored corresponding to the target block data in a block storage state mapping table until the amount of data to be stored reaches a decrement threshold, and determine that a complete block corresponding to the target block data has been successfully written into the distributed database.

**[0163]** The specific implementations of the first detection module 11, the first writing module  
15 12, the second detection module 13 and the second writing module 14 may refer to the description of steps S101-S104 in the embodiment shown in FIG. 3, which will not be repeated here. Further, the specific implementations of the transaction acquisition module 15, the transaction execution module 16, the first block determination module 17, the local storage module 18, the notification block execution module 19, the second block determination module 20, the buffer refreshing module 21,  
20 the block data addition module 22, the execution waiting module 23, the to-be-stored data determination module 24, the buffer waiting module 25, the completion amount receiving module 26 and the data amount decrement module 27 may refer to the description of steps S201-S208 in the embodiment shown in FIG. 5, which will not be repeated here. It may be understood that, the description of beneficial effects of the same method is not described herein again.

25 **[0164]** Further referring to FIG.11, FIG. 11 is a schematic structural diagram of a computer device according to an embodiment of this application. As shown in FIG. 11, the computer device 1000 may be applied to the block chain node in the above-mentioned embodiment corresponding to FIG. 1. The computer device 1000 may include: a processor 1001, a network interface 1004 and a memory 1005. In addition, the computer device 1000 may further include: a user interface 1003, and  
30 at least one communication bus 1002. The communications bus 1002 is configured to implement connection and communication between the components. The user interface 1003 may include a display, a keyboard. In an embodiment, the user interface 1003 may further include a standard wired interface and a standard wireless interface. In an embodiment, the network interface 1004 may include

a standard wired interface and a standard wireless interface (such as a Wi-Fi interface). The memory 1004 may be a high-speed random access memory (RAM), or may be a non-volatile memory, for example, at least one magnetic disk memory. In an embodiment, the memory 1005 may be at least one storage apparatus that is located far away from the foregoing processor 1001. As shown in FIG. 11, the memory 1005 used as a computer storage medium may include an operating system, a network communication module, a user interface module, and a device-control application program.

**[0165]** The network interface 1004 in the computer device 1000 may also provide a network communication function, and optionally, the user interface 1003 may also include a display screen and a keyboard. In the computer device 1000 shown in FIG. 11, the network interface 1004 may provide a network communication function, the user interface 1003 is mainly configured to provide an input interface for the user, and The processor 1001 may be configured to call a device control application program stored in the memory 1005 to realize the description of the data processing method based on distributed storage in the embodiment shown in FIG. 3 or FIG. 5, or the description of the data processing apparatus 1 based on distributed storage in the embodiment shown in FIG. 10, which will not be repeated here. In addition, the description of beneficial effects of the same method is not described herein again.

**[0166]** In addition, it is to be noted that a computer-readable storage medium is further provided according to an embodiment of this application. The computer-readable storage medium stores a computer program executed by the data processing apparatus 1 based on distributed storage mentioned above, and the computer program includes program instructions. When executing the program instruction, the processor can perform the data processing method based on distributed storage in the embodiment corresponding to FIG. 3 or FIG. 5. Therefore, details are not described herein again. In addition, the beneficial effect of the same method is not described herein again. For technical details that are not disclosed in the embodiments of the computer-readable storage medium of this application, refer to the method embodiments of this application. In an example, the program instructions may be deployed to be executed on a computing device, or deployed to be executed on multiple computing devices at the same location, or deployed to be executed on multiple computing devices that are distributed in multiple locations and interconnected by using a communication network, where multiple computing devices that are distributed in multiple locations and interconnected through a communication network may form a block chain system.

**[0167]** In addition, it is to be noted that a computer program product or a computer program is further provided according to an embodiment of this application. The computer program product or the computer program may include computer instructions, the computer instructions being stored

in a computer-readable storage medium. In addition, the beneficial effect of the same method is not described herein again. For technical details that are not disclosed in the embodiments of the computer-readable storage medium of this application, refer to the method embodiments of this application.

5     **[0168]**         A person of ordinary skill in the art may understand that all or some of the procedures of the methods of the foregoing embodiments may be implemented by a computer program instructing relevant hardware. The computer program may be stored in a computer readable storage medium. When the program is executed, the program may include the procedures of the embodiments of the foregoing methods. The foregoing storage medium may include a magnetic disc, an optical  
10    disc, a ROM, a RAM, or the like.

**[0169]**         The description disclosed above is merely exemplary embodiments of this application, and certainly is not intended to limit the scope of the claims of this application. Therefore, equivalent variations made according to the claims of this application shall fall within the scope of this application.

15

## CLAIMS

1. A data processing method based on distributed storage, executed by a block chain node and comprising:

detecting, after writing first block data of a first block into a local cache and a local storage, a  
 5 first block processing space of a storage buffer pool associated with the local cache to obtain a first space detection result, the storage buffer pool comprising N pieces of block data to be stored, N being a non-negative integer, wherein the storage buffer pool and the local cache are both included in a storage layer of the block chain node;

writing, if the first space detection result indicates that a number of block data to be stored in the  
 10 first block processing space does not reach a buffer block processing threshold of the first block processing space, the first block data into the storage buffer pool based on the N pieces of block data to be stored;

detecting a second block processing space of a storage processing pool associated with the storage buffer pool to obtain a second space detection result, the storage processing pool being  
 15 configured to store M pieces of block data to be fed back associated with a distributed database, M being a non-negative integer, wherein the M pieces of block data to be fed back in the storage processing pool are block data that are currently being stored in the distributed database, but have not received stored feedback information returned by the distributed database; and

determining, if the second space detection result indicates that a number of block data to be fed  
 20 back in the second block processing space does not reach a feedback block processing threshold of the second block processing space, block data to be combined in the storage buffer pool based on the number of the block data to be fed back and the feedback block processing threshold, performing duplication detection on the block data to be combined and the M pieces of block data to be fed back to determine data that are not duplicated as retained data, and writing the retained data into the  
 25 distributed database.

2. The method according to claim 1, further comprising:

acquiring K1 transaction services in the first block in a case that an executive layer receives the first block transmitted by a consensus layer, the first block comprising first block header information  
 30 and a transaction list associated with the K1 transaction services, K1 being a positive integer;

acquiring, for each of the K1 transaction services, a service contract for executing the transaction

service, and executing the transaction service through the service contract of the transaction service to obtain a transaction execution result of the transaction service;

using the first block header information, the transaction list, and the transaction execution result of each of the K1 transaction services as first block data of the first block; and

5 writing the first block data of the first block into the local cache, and writing the first block data of the first block into the local storage corresponding to the local cache.

3. The method according to claim 2, wherein the acquiring the service contract for executing the transaction service and executing the transaction service through the service contract of the transaction service to obtain the transaction execution result of the transaction service comprises:

acquiring, based on a contract calling address of the transaction service, a service contract for executing the transaction service;

searching, through the service contract of the transaction service, the local cache for target data associated with the transaction service;

15 using the target data as target read data of the transaction service if the target data associated with the transaction service is found from the local cache; and

executing the transaction service based on the target read data of the transaction service, to obtain a transaction execution result of the transaction service.

20 4. The method according to claim 3, further comprising:

searching, through the service contract, the distributed database for the target data associated with the transaction service if the target data associated with the transaction service is not found from the local cache, and using the target data as the target read data of the transaction service.

25 5. The method according to claim 2, further comprising:

notifying the executive layer to execute K2 transaction services in a second block after writing the first block data into the local cache and the local storage, the second block being a next block of the first block, and the second block being obtained by packaging the K2 transaction services by the consensus layer, K2 being a positive integer.

6. The method according to claim 5, further comprising:

receiving second block data of the second block transmitted by the executive layer after writing the first block data into the storage buffer pool, the second block data being obtained by the executive layer after executing the K2 transaction services;

5 performing, through a first refreshing thread, buffer refreshing on the first block processing space of the storage buffer pool to obtain a buffer refreshing result, and updating the first space detection result by using the buffer refreshing result; and

writing, if the updated first space detection result indicates that the number of block data to be stored in the first block processing space does not reach the buffer block processing threshold, the  
10 second block data into the storage buffer pool based on the block data to be stored in the first block processing space.

7. The method according to claim 1, before writing the first block data into the storage buffer pool, the method further comprising:

15 notifying an executive layer to wait for the first block data to be stored if the first space detection result indicates that the number of block data to be stored in the first block processing space has reached the buffer block processing threshold.

8. The method according to claim 1, wherein the writing, based on the N pieces of block data to  
20 be stored, the first block data into the storage buffer pool comprises:

acquiring the N pieces of block data to be stored from the storage buffer pool, and searching the N pieces of block data to be stored for block data to be stored that duplicates with the first block data;

using, if the block data to be stored that duplicates with the first block data is found from the N pieces of block data to be stored, the found block data to be stored as duplicated block data; and

25 replacing the duplicated block data with the first block data, using the replaced N pieces of block data to be stored as first combined data associated with the first block data, and writing the first combined data into the storage buffer pool.

9. The method according to claim 8, further comprising:

30 using, if the block data to be stored that duplicates with the first block data is not found from the

N pieces of block data to be stored, the first block data and the N pieces of block data to be stored as second combined data associated with the first block data, writing the second combined data into the storage buffer pool, and performing incremental processing on the number of block data to be stored in the first block processing space.

5

10. The method according to claim 1 or 8 or 9, further comprising:

using the combined data in the storage buffer pool as data to be stored; and

notifying the storage buffer pool to wait for the storage of the data to be stored if the second space detection result indicates that a number of block data to be fed back in the second block processing space has reached a feedback block processing threshold.

10

11. The method according to claim 1, wherein the data to be stored in the storage buffer pool comprises the N pieces of block data to be stored and the first block data, and the first block data has no data duplication with the N pieces of block data to be stored, and

the determining the block data to be combined from the storage buffer pool based on the number of the block data to be fed back and the feedback block processing threshold, performing duplication detection on the block data to be combined and the M pieces of block data to be fed back, and writing the data retained after duplication detection into the distributed database comprises:

15

determining a difference between the number of the block data to be fed back and the feedback block processing threshold;

20

determining the block data to be combined from the N pieces of block data to be stored and the first block data based on the number difference and a block height of the data to be stored in the storage buffer pool; performing duplication detection on the block data to be combined and the M pieces of block data to be fed back to obtain an duplication detection result;

writing the block data to be combined into the storage processing pool if the duplication detection result indicates that the M pieces of block data to be fed back has no data to be fed back that duplicates with the block data to be combined; and

25

using the block data to be combined and the M pieces of block data to be fed back in the storage processing pool as data retained after duplication detection, and writing the data retained after duplication detection into the distributed database.

30

12. The method according to claim 11, further comprising:

receiving a data completion amount of the target block data fed back by the distributed database, the target block data being block data to which data successfully written into the distributed database belongs in the data retained after the duplication detection; and

5 performing, based on the data completion amount, decremental processing on the amount of data to be stored corresponding to the target block data in a block storage state mapping table until the amount of data to be stored reaches a decrement threshold, and determining that a complete block corresponding to the target block data has been successfully written into the distributed database.

10 13. A data processing apparatus based on distributed storage, comprising:

a first detection module configured to detect, after writing first block data of a first block into a local cache and a local storage, a first block processing space of a storage buffer pool associated with the local cache to obtain a first space detection result, the storage buffer pool comprising N pieces of block data to be stored, N being a non-negative integer, wherein the storage buffer pool and the local  
15 cache are both included in a storage layer of the block chain node;

a first writing module configured to write, if the first space detection result indicates that a number of block data to be stored in the first block processing space does not reach a buffer block processing threshold of the first block processing space, the first block data into the storage buffer pool based on the N pieces of block data to be stored;

20 a second detection module configured to detect a second block processing space of a storage processing pool associated with the storage buffer pool to obtain a second space detection result, the storage processing pool being configured to store M pieces of block data to be fed back associated with a distributed database, M being a non-negative integer, wherein the M pieces of block data to be fed back in the storage processing pool are block data that are currently being stored in the distributed  
25 database, but have not received stored feedback information returned by the distributed database; and

a second writing module configured to determine, if the second space detection result indicates that a number of block data to be fed back in the second block processing space does not reach a feedback block processing threshold of the second block processing space, block data to be combined in the storage buffer pool based on the number of the block data to be fed back and the feedback block  
30 processing threshold, performing duplication detection on the block data to be combined and the M pieces of block data to be fed back to determine data that are not duplicated as retained data, and writing the retained data into the distributed database.

14. A computer device, comprising: a processor and a memory,

the processor being connected to the memory, the memory being configured to store a computer program, and the processor being configured to invoke the computer program to cause the computer  
5 device to perform the method according to any one of claims 1 to 12.

15. A computer-readable storage medium, storing a computer program, the computer program, applicable to be loaded and executed by a processor, causing a computer device comprising the processor to perform the method according to any one of claims 1 to 12.

10

16. A computer program product, comprising instructions, the instructions, when run on a computer, causing the computer to implement the method according to any one of claims 1 to 12.