

TIMESTAMP ALLOCATION METHOD AND APPARATUS FOR DISTRIBUTED DATABASE

TECHNICAL FIELD

[0001] One or more embodiments of this specification relate to the field of database technologies, and in particular, to a timestamp allocation method and apparatus for a distributed database.

BACKGROUND

[0002] As an important application part of a distributed system, a distributed database not only inherits core features of a conventional single-node database, but also has a processing capability of the distributed system, and features low hardware costs, high scalability, high availability, etc. The distributed database includes a plurality of nodes, and the nodes need to exchange messages to communicate with each other. A database instance on each node has an independent clock, and the clock on each node cannot reflect a global order. Therefore, a clock offset may occur, and consistency of the distributed database cannot be ensured.

[0003] To avoid the clock offset, globally consistent version numbers can be provided based on a clock service. The clock service is to establish an order relationship for transactions in the database by using timestamps (ts). A larger timestamp indicates a larger version number of a transaction, and the transaction being later in an order. In the related art, a single timing node provides the clock service for all transactions in the database. The single timing node can ensure that timestamps are allocated to transactions of the nodes incrementally, thereby ensuring consistency between the transactions in the database.

[0004] For such a distributed database that uses a single-point-timing clock service, each node needs to send a timestamp obtaining request to a timing node for each transaction, which easily overloads the timing node. When an upper processing limit of the timing node is reached, time to respond to the timestamp obtaining request is significantly prolonged. In addition, each time a transaction is committed, a request needs to be sent to the timing node, and the transaction needs

to wait for the timing node to return an allocated timestamp, which significantly prolongs processing of each transaction, and reduces database performance.

SUMMARY

[0005] This specification provides a timestamp allocation method and apparatus for a distributed database, to resolve problems in the related art.

[0006] According to a first aspect of one or more embodiments of this specification, a timestamp allocation method for a distributed database is provided. Nodes of the distributed database include a timing node and an ordinary node, the method is applied to any node, and the method includes: committing a corresponding to-be-timed transaction in response to a received transaction commission request, and recording a start-to-commit moment of the to-be-timed transaction; querying for a timing request sending moment at which the any node sends a timing request to the timing node for another transaction; and when a timing request sending moment of any other transaction is not earlier than the start-to-commit moment of the to-be-timed transaction, allocating, to the to-be-timed transaction, a timestamp fed back by the timing node for the any other transaction.

[0007] According to a second aspect of one or more embodiments of this specification, a timestamp allocation apparatus for a distributed database is provided. Nodes of the distributed database include a timing node and an ordinary node, the apparatus is applied to any node, and the apparatus includes: a commission module, configured to commit a corresponding to-be-timed transaction in response to a received transaction commission request, and record a start-to-commit moment of the to-be-timed transaction; a query module, configured to query for a timing request sending moment at which the any node sends a timing request to the timing node for another transaction; and an allocation module, configured to: when a timing request sending moment of any other transaction is not earlier than the start-to-commit moment of the to-be-timed transaction, allocate, to the to-be-timed transaction, a timestamp fed back by the timing node for the any other transaction.

[0008] According to a third aspect of one or more embodiments of this specification, an electronic device is provided, including a processor, and a memory configured to store a processor-executable instruction. The processor runs the executable instruction to implement the method in

the embodiment of the first aspect.

[0009] According to a fourth aspect of one or more embodiments of this specification, a computer-readable storage medium is provided, storing a computer instruction. The instruction is executed by a processor to implement steps of the method in the embodiment of the first aspect.

5 **[0010]** It can be learned from the above-mentioned technical solutions that, in one or more embodiments of this specification, a timestamp cache is added to each node in the distributed database to store a timestamp previously obtained from the timing node, so that a plurality of transactions can share a same timestamp on a basis of ensuring external consistency of the database. In this way, a frequency at which each node sends a timestamp obtaining request to the timing
10 node is effectively reduced. Accordingly, load caused to the timing node by processing the request is reduced, and a problem that the timing node takes a long time to return a timing response due to overload of the timing node is avoided. In addition, because it is free of waiting for the timing node to allocate a timestamp, waiting duration is shortened, so that processing efficiency of each transaction is significantly improved, and database performance is ensured.

15 **[0011]** It should be understood that the general descriptions above and the detailed descriptions below are merely examples for an explanatory purpose, and are not intended to limit this specification.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] The accompanying drawings herein, which are incorporated in this specification as a
20 part of this specification, show embodiments consistent with this specification, and together with this specification, are used to explain the principles of this specification.

[0013] FIG. 1 is an architectural diagram of a distributed database according to an example embodiment;

[0014] FIG. 2 is a flowchart of a timestamp allocation method for a distributed database
25 according to an example embodiment;

[0015] FIG. 3a is a schematic diagram of ensuring external consistency of a distributed database according to an example embodiment;

[0016] FIG. 3b is a schematic diagram of ensuring external consistency of a distributed database according to another example embodiment;

[0017] FIG. 4 is a schematic structural diagram of an electronic device according to an example embodiment; and

[0018] FIG. 5 is a block diagram of a timestamp allocation apparatus for a distributed database according to an example embodiment.

5

DESCRIPTION OF EMBODIMENTS

[0019] Example embodiments will be described herein in detail, and examples thereof are shown in the accompanying drawings. In the following descriptions involving the accompanying drawings, the same numerals in different accompanying drawings denote the same or similar elements, unless specified otherwise. The implementations set forth in the following descriptions
10 of example embodiments do not represent all implementations consistent with this specification. Instead, they are merely examples of apparatuses and methods consistent with some aspects of this specification as detailed in the appended claims.

[0020] It should be noted that, in another embodiment, steps of a corresponding method are not necessarily performed in an order shown and described in this specification. In some other
15 embodiments, the method may include more or less steps than those described in this specification. In addition, individual steps described in this specification may be split into a plurality of steps in another embodiment for description, and a plurality of steps described in this specification may be combined into a single step in another embodiment for description.

[0021] Distributed databases are widely used for data storage and management due to their
20 high scalability and reliability. FIG. 1 is an architectural diagram of a distributed database according to an example embodiment. As shown in FIG. 1, the distributed database includes a plurality of nodes, including a timing node and an ordinary node. All ordinary nodes send a timing request to the timing node to obtain a timestamp allocated by the timing node. In the related art, a single timing node in such an architecture provides a clock service for all transactions in the
25 database to ensure consistency between the transactions in the database. However, in such a single-point timing manner, each ordinary node needs to send a timing request to the timing node for each transaction on the ordinary node one by one, which easily overloads the timing node in processing the timing request. When an upper processing limit of the timing node is reached, time to respond to the timing request is significantly prolonged. In addition, each time a transaction is committed,

a timing request needs to be sent to the timing node, and the transaction needs to wait for the timing node to return an allocated timestamp, which significantly prolongs execution of each transaction, and reduces database performance.

[0022] This specification provides a timestamp allocation method for a distributed database, to resolve the problems in the related art.

[0023] FIG. 2 is a flowchart of a timestamp allocation method for a distributed database according to an example embodiment. The method may be applied to any node of the distributed database in FIG. 1. In addition to the ordinary node, when the timing node not only has a timing function but also can implement a transaction processing function of the ordinary node, the method is also applicable to the timing node. As shown in FIG. 2, the method may include the following steps S201 to S203.

[0024] S201. Commit a corresponding to-be-timed transaction in response to a received transaction commission request, and record a start-to-commit moment of the to-be-timed transaction.

[0025] A database transaction may be understood as a logical unit in a database execution process, and includes a limited database operation sequence. Generally, one transaction includes a plurality of read/write operations. The transaction has four basic characteristics: atomicity, consistency, isolation, and durability. The durability of the transaction determines that the transaction is valid only when committed. After the transaction is committed, a related modification of the transaction to the database can be permanently stored in the database and read by other users. If the transaction is not committed, content of the transaction is stored in a cache and can be rolled back at any time. There are three database transaction commission manners: manual explicit commission, implicit commission, and automatic commission. A person skilled in the art can set a manner according to actual requirements, which is not limited in this specification.

[0026] Generally, an entire transaction execution process may be divided into three stages: the transaction starts to be committed, a timestamp is obtained for the transaction, and commission of the transaction is completed. A timestamp obtaining process may include as follows: an ordinary node sends a timing request to the timing node for a committed transaction, the timing node receives the timing request and returns a timing response to the ordinary node that sends the timing request, and the ordinary node receives the timing response and obtains a timestamp fed back for the transaction. However, this timestamp obtaining process is time-consuming. In addition, the

timing request needs to be sent to the timing node separately for each transaction, which easily overloads the timing node in processing the request. Based on this, this specification proposes another timestamp obtaining manner, that is, a plurality of transactions share a same timestamp. Specifically, when receiving a transaction commission request, any node in the distributed database commits a corresponding to-be-timed transaction, and records a start-to-commit moment of the to-be-timed transaction for use in a subsequent process. In this specification, the start-to-commit moment of the to-be-timed transaction may be denoted as Absolute Time (Start To Commit).

[0027] S202. Query for a timing request sending moment at which the any node sends a timing request to the timing node for another transaction.

[0028] Generally, after the to-be-timed transaction starts to be committed, the node that receives the transaction commission request needs to send a timing request to the timing node, so that the timing node allocates a timestamp to the to-be-timed transaction. Therefore, there is a timing request sending moment at which the timing request is sent to the timing node. Generally, the start-to-commit moment of the transaction and the timing request sending moment are not a same moment, and there is a time difference between the two. In this specification, information exchange between the ordinary node and the timing node depends on a Remote Procedure Call (RPC) protocol. Therefore, the timing request sending moment may be denoted as Absolute Time (Send RPC Request). Certainly, the ordinary node and the timing node may alternatively exchange information with each other based on another protocol, which is not limited in this specification.

[0029] An ordinary node N1 is used as an example. In an embodiment, the ordinary node N1 receives a commission request of a transaction T0, commits the transaction T0 in response to the request, and records a start-to-commit moment, denoted as Absolute Time (T0. Start To Commit), of the transaction T0. In this case, the transaction T0 is a to-be-timed transaction, and needs to be allocated a timestamp. After the node N1 records the start-to-commit moment of the to-be-timed transaction T0, the node N1 may query for a timing request sending moment at which the node sends a timing request to the timing node for another transaction. The another transaction herein may be understood as all transactions on the node N1 for which a timing request is sent to the timing node except the current to-be-timed transaction T0, including a timed transaction and an untimed transaction. The timed transaction is a transaction for which a timing request is sent to the timing node and a timing response returned by the timing node is received when the to-be-timed

transaction T0 starts to be committed. The untimed transaction is a transaction for which a timing request is sent to the timing node but no timing response returned by the timing node is received when the to-be-timed transaction T0 starts to be committed.

[0030] S203. When a timing request sending moment of any other transaction is not earlier than the start-to-commit moment of the to-be-timed transaction, allocate, to the to-be-timed transaction, a timestamp fed back by the timing node for the any other transaction.

[0031] An entire execution process of a transaction chronologically includes as follows: the transaction starts to be committed, a timing request is sent to the timing node, the timing node receives the timing request, and commission of the transaction is completed. That is, a moment at which commission of the transaction is completed is later than a moment at which the timing node receives the timing request, the moment at which the timing node receives the timing request is later than a moment at which the timing request is sent to the timing node, and the moment at which the timing request is sent to the timing node is later than a moment at which the transaction starts to be committed. External consistency of the database means that an execution sequence of transactions inside the database cannot be contrary to an order observed outside. If an order of two transactions can be observed outside the database, the transactions are definitely committed one after the other. Otherwise, the two transactions are concurrent, and the two transactions can be executed in any order inside the database.

[0032] For example, it is observed outside the database that a first transaction writes data A into one node, and after completion, a second transaction starts to write data B into another node. In this process, a third transaction running concurrently attempts to read A and B. If the data B is read but the data A fails to be read, it means that the second transaction is executed before the first transaction inside the database, which is contrary to the order observed outside the database, that is, external consistency is not met. Therefore, to ensure external consistency of the database, a timestamp needs to be allocated to each transaction. An objective of external consistency is that when a transaction Tb starts to be committed after commission of a transaction Ta is completed, a timestamp allocated to the transaction Tb needs to be greater than a timestamp allocated to the transaction Ta, to ensure that the transaction Ta is executed before the transaction Tb inside the database based on magnitudes of the timestamps.

[0033] In an embodiment, a transaction T3 starts to be committed after commission of a transaction T1 is completed. To ensure external consistency, a timestamp of the transaction T3

needs to be greater than a timestamp of the transaction T1. If the node has a transaction whose timing request sending moment is not earlier than a start-to-commit moment of the transaction T3 when the transaction T3 starts to be committed, when this transaction receives a timestamp returned by the timing node, the transaction T3 may reuse the timestamp of this transaction.

5 Detailed descriptions are provided below with reference to FIG. 3. As shown in FIG. 3a, a timing request sending moment 306 of a transaction T2 is later than the start-to-commit moment 309 of the transaction T3. In this case, it can be inferred that a moment 307 at which the timing node receives a timing request of the transaction T2 is later than the start-to-commit moment 309 of the transaction T3. In addition, since the transaction T3 starts to be committed after commission of the transaction T1 is completed, that is, the start-to-commit moment 309 of the transaction T3 is later than a commission completion moment 304 of the transaction T1, it can be further inferred that the moment 307 at which the timing node receives the timing request of the transaction T2 is later than the commission completion moment 304 of the transaction T1. Therefore, a timestamp of the transaction T2 is definitely greater than the timestamp of the transaction T1. In this case, the transaction T3 can reuse the timestamp of the transaction T2, so that the timestamp of the transaction T3 is greater than the timestamp of the transaction T1, thereby ensuring external consistency.

[0034] Certainly, if the timing request sending moment 306 of the transaction T2 and the start-to-commit moment 309 of the transaction T3 are a same moment, it can also be inferred that the moment 307 at which the timing node receives the timing request of the transaction T2 is later than the start-to-commit moment 309 of the transaction T3, and it can be further inferred that the moment 307 at which the timing node receives the timing request of the transaction T2 is later than the commission completion moment 304 of the transaction T1, that is, the timestamp of the transaction T2 is greater than the timestamp of the transaction T1. In this case, the transaction T3 can also reuse the timestamp of the transaction T2 to ensure external consistency. In conclusion, when the timing request sending moment 306 of the transaction T2 is not earlier than the start-to-commit moment 309 of the transaction T3, the transaction T3 can reuse the timestamp of the transaction T2 to ensure external consistency between the transaction T3 and the transaction T1.

[0035] In the above-mentioned embodiment, a start-to-commit moment 305 of the transaction T2 may be earlier than the commission completion moment 304 of the transaction T1, or may be later than the commission completion moment 304 of the transaction T1, that is, the start-to-

commit moment 305 of the transaction T2 is unrelated to a plurality of transaction sharing a same timestamp in this specification.

[0036] In addition, in the above-mentioned embodiment, the timestamp of the transaction T1 may be obtained from the timing node, or may be a reused timestamp of another transaction, provided that a timing request sending moment of the another transaction is not earlier than a start-to-commit moment of the transaction T1. Referring to FIG. 3b, a timing request sending moment 311 of a transaction T0 is later than the start-to-commit moment 301 of the transaction T1. According to the above-mentioned inference, the transaction T1 may reuse a timestamp of the transaction T0. In this case, the commission completion moment 304 of the transaction T1 is definitely later than a moment 312 at which the timing node receives a timing request of the transaction T0. Further, according to the inference in FIG. 3a, because the moment 307 at which the timing node receives the timing request of the transaction T2 is later than the commission completion moment 304 of the transaction T1, the moment 307 at which the timing node receives the timing request of the transaction T2 is definitely later than the moment 312 at which the timing node receives the timing request of the transaction T0, that is, the timestamp of the transaction T2 is greater than the timestamp of the transaction T0. Therefore, when the transaction T3 reuses the timestamp of the transaction T2 and the transaction T1 reuses the timestamp of the transaction T0, since the timestamp of the transaction T3 is greater than the timestamp of the transaction T1, external consistency between the transaction T1 and the transaction T3 is implemented.

[0037] In conclusion, when a timing request sending moment of a transaction Tn is not earlier than a start-to-commit moment of a to-be-timed transaction Tx, that is, $\text{Absolute Time (Tn. Send RPC Request)} \geq \text{Absolute Time (Tx. Start To Commit)}$, Tx can use a timestamp obtained by Tn, to meet external consistency.

[0038] In an embodiment, the node N1 may preferentially query, in an order of received timestamps returned by the timing node, for a timing request sending moment at which the node sends a timing request to the timing node for a latest timed transaction. The timing request sending moment of the latest timed transaction is stored in a timestamp cache. Therefore, the timing request sending moment of the latest timed transaction may be denoted as $\text{Absolute Time (ts cache. Last Send RPC Request)}$. When the timing request sending moment of the latest timed transaction is not earlier than the start-to-commit moment of the to-be-timed transaction T0, that is, $\text{Absolute Time (ts cache. Last Send RPC Request)} \geq \text{Absolute Time (T0. Start To Commit)}$, a timestamp

received by the latest timed transaction is allocated to the to-be-timed transaction T0, that is, the to-be-timed transaction T0 and the latest timed transaction share the same timestamp, and version numbers of the two transactions are consistent. In this case, the node N1 does not need to send a timing request to the timing node for the to-be-timed transaction T0. In this way, a frequency at which the node N1 sends a timing request to the timing node is reduced, and load pressure of the timing node is accordingly reduced. In addition, because the timestamp received by the latest timed transaction is stored in the cache, the to-be-timed transaction T0 can directly reuse the timestamp in the cache, thereby significantly reducing time in waiting to be timed, and improving processing efficiency of the to-be-timed transaction T0.

[0039] In another embodiment, when the timing request sending moment of the latest timed transaction does not meet the preset condition, the node N1 may query for a timing request sending moment at which a timing request is sent to the timing node for another untimed transaction on the node other than the latest timed transaction. Similarly, the timing request sending moment of the untimed transaction is denoted as Absolute Time (ts cache. Send RPC Request). When it is found that a timing request sending moment of any untimed transaction is not earlier than the start-to-commit moment of the to-be-timed transaction T0, that is, $\text{Absolute Time (ts cache. Send RPC Request)} \geq \text{Absolute Time (T0. Start To Commit)}$, it indicates that a timestamp in a timing response corresponding to the any untimed transaction may be allocated to the to-be-timed transaction T0, so that the to-be-timed transaction T0 and the any untimed transaction share the same timestamp, and version numbers of the two transactions are consistent. For example, it is found that a timing request sending moment of an untimed transaction T4 meets the preset condition. In this case, the to-be-timed transaction T0 only needs to be added to a to-be-timed queue cached by the node N1, and wait for the timing node to return a timing response for the untimed transaction T4. When the untimed transaction T4 receives the timing response returned by the timing node, a timestamp in the timing response is allocated to the to-be-timed transaction T0, so that the to-be-timed transaction T0 and the transaction T4 share the same timestamp. In this case, the node N1 does not need to send a timing request to the timing node for the to-be-timed transaction T0, either, which reduces a frequency at which the node N1 sends a timing request to the timing node, and reduces load pressure of the timing node to a certain extent. In this embodiment, the adding the to-be-timed transaction T0 to the to-be-timed queue to wait for the timing response of the to-be-timed transaction T4 to be returned may be used as an asynchronous task, to avoid affecting a

commission process of another transaction.

[0040] In an embodiment, when timing request sending moments of all other transactions than the to-be-timed transaction T0 on the node N1 do not meet the preset condition, that is, the timing request sending moments of all the other transactions are earlier than the start-to-commit moment of the to-be-timed transaction T0, the node N1 needs to send a timing request to the timing node for the to-be-timed transaction T0. The timing node allocates a timestamp in response to the received timing request, and assigns a value to the timestamp in a timing response. In addition, the timing response includes related information about the timing request, so that when receiving the timing response, the node N1 can determine the timing request corresponding to the timing response. Because timestamps of the timing node do not fall back, that is, the timing node allocates a timestamp to a transaction of each node incrementally, it can be ensured that a timestamp of a latest timed transaction is definitely the largest, thereby ensuring external consistency of the database.

[0041] In an embodiment, each node in the distributed database is further responsible for receiving any timing response returned by the timing node, and comparing a timing request sending moment corresponding to the received timing response with a timing request sending moment of a latest timed transaction in a timestamp cache. If the timing request sending moment corresponding to the timing response is later than the timing request sending moment of the latest timed transaction in the timestamp cache, the timing request sending moment of the latest timed transaction in the cache is updated to the timing request sending moment corresponding to the timing response. In addition, a timestamp stored in the timestamp cache is updated to a timestamp fed back in the timing response. If the timing response corresponding to the timing request sending moment is not later than the timing request sending moment of the latest timed transaction in the timestamp cache, the timing response is discarded. This update operation can ensure that the timing request sending moment, that is, Absolute Time (ts cache. Last Send RPC Request), of the latest timed transaction in the cache is always the largest value in a set of timing request sending moments of timed transactions, so that any to-be-timed transaction that meets the condition that a start-to-commit moment of the transaction is not later than the timing request sending moment of the latest timed transaction can use the timestamp of the latest timed transaction.

[0042] In an embodiment, each time a timestamp stored in the timestamp cache is updated, all to-be-timed transactions of a to-be-timed queue on the node are traversed. After the update, a

timing request sending moment of a latest timed transaction in the cache is the largest value in a current set of timing request sending moments of timed transactions in the cache. Therefore, the updated timestamp can be allocated to all to-be-timed transactions whose start-to-commit moments are not later than the current largest timing request sending moment.

5 **[0043]** In an embodiment, a current state of the timing node may be added to a timing response returned by the timing node, so that when receiving the timing response, each node adjusts, based on the current state of the timing node, a frequency at which the node sends a timing request to the timing node. For example, the state of the timing node may be classified into three types: normal, idle, and busy. When any node learns that the current state of the timing node is the idle state, the
10 any node may send a timing request to the timing node for all current to-be-timed transactions one by one, thereby increasing a frequency of sending a timing request to the timing node, fully utilizing the timing node, and avoiding a waste of resources. In addition, each to-be-timed transaction can obtain a timestamp allocated by the timing node, thereby fully ensuring external consistency of the database. When any node learns that the current state of the timing node is the
15 busy state, the any node may first query whether the node has another transaction that complies with the above-mentioned embodiment. If yes, a timestamp of the another transaction is allocated to a to-be-timed transaction, thereby reducing a frequency of sending a timestamp obtaining request to the timing node, and reducing load pressure of the timing node. When any node learns that the current state of the timing node is the normal state, it indicates that a frequency at which
20 the node sends a timing request to the timing node is in a balanced state. Provided that the current frequency is maintained, the timing node does not have excessive load pressure, and efficiency of processing the timing request by the timing node can be ensured, thereby avoiding a problem that a long time is taken to return a timing response.

25 **[0044]** Corresponding to the above-mentioned method embodiment, this specification further provides an embodiment of an apparatus.

30 **[0045]** FIG. 4 is a schematic structural diagram of an electronic device according to an example embodiment of this specification. Referring to FIG. 4, at a hardware level, the electronic device includes a processor 402, an internal bus 404, a network interface 406, an internal memory 408, and a non-volatile memory 410. Certainly, the electronic device may further include hardware required by another service. The processor 402 reads a corresponding computer program from the non-volatile memory 410 to the internal memory 408 and then runs the computer program.

Certainly, in addition to a software implementation, this specification may include another implementation, such as a logic component or a combination of software and hardware, that is, an execution body of the following processing procedure is not limited to a logical unit, and may be hardware or a logical component.

5 **[0046]** FIG. 5 is a block diagram of a timestamp allocation apparatus for a distributed database according to an example embodiment of this specification. Nodes of the distributed database include a timing node and an ordinary node, and the apparatus is applied to any node. Referring to FIG. 5, the apparatus includes a commission module 502, a query module 504, and an allocation module 506.

10 **[0047]** The commission module 502 is configured to commit a corresponding to-be-timed transaction in response to a received transaction commission request, and record a start-to-commit moment of the to-be-timed transaction.

[0048] The query module 504 is configured to query for a timing request sending moment at which the any node sends a timing request to the timing node for another transaction.

15 **[0049]** The allocation module 506 is configured to: when a timing request sending moment of any other transaction is not earlier than the start-to-commit moment of the to-be-timed transaction, allocate, to the to-be-timed transaction, a timestamp fed back by the timing node for the any other transaction.

[0050] Optionally, the query module 504 is specifically configured to query for a timing request sending moment at which the any node sends a timing request to the timing node for a latest timed transaction. When the found timing request sending moment is not earlier than the start-to-commit moment of the to-be-timed transaction, the latest timed transaction is determined as the any other transaction.

[0051] Optionally, the query module 504 is further configured to: when the timing request sending moment of the latest timed transaction is earlier than the start-to-commit moment of the to-be-timed transaction, query for a timing request sending moment at which the any node sends a timing request to the timing node for another untimed transaction. When a timing request sending moment of any other untimed transaction is not earlier than the start-to-commit moment of the to-be-timed transaction, the any other untimed transaction is used as the any other transaction.

30 **[0052]** Optionally, the apparatus further includes: a sending module 508, configured to: when timing request sending moments of all other transactions are earlier than the start-to-commit

moment of the to-be-timed transaction, send a timing request to the timing node for the to-be-timed transaction, so that the timing node allocates a timestamp to the to-be-timed transaction in response to the timing request;

5 a receiving module 510, configured to receive any timing response returned by the timing node;

an update module 512, configured to: when a timing request sending moment corresponding to the any timing response is later than a timing request sending moment of a latest timed transaction in a cache, update the timing request sending moment of the latest timed transaction in the cache to the timing request sending moment corresponding to the timing response;

10

a traversal module 514, configured to traverse all to-be-timed transactions in a to-be-timed queue in the cache, and allocate a timestamp of the latest timed transaction in the cache to all to-be-timed transactions that meet a condition, where the condition includes that the timing request sending moment of the latest timed transaction is not earlier than start-to-commit moments of the to-be-timed transactions; and

15

an adjustment module 516, configured to adjust, based on a current running state of the timing node included in the any timing response, a frequency at which the any node sends a timing request to the timing node, where the current running state of the timing node indicates a busy degree of the timing node.

20 **[0053]** For details about implementation processes of functions and effects of the modules in the above-mentioned apparatus, refer to implementation processes of corresponding steps in the above-mentioned method. Details are not described herein again.

[0054] The apparatus or module illustrated in the above-mentioned embodiment can be specifically implemented by a computer chip or an entity, or implemented by a product having a certain function. A typical implementation device is a computer, and a specific form of the computer may be a personal computer, a laptop computer, a cellular phone, a camera phone, a smart phone, a personal digital assistant, a media player, a navigation device, an e-mail transceiver device, a game console, a tablet computer, a wearable device, or a combination of any of these devices.

25

30 **[0055]** In a typical configuration, the computer includes one or more processors (CPUs), an input/output interface, a network interface, and an internal memory.

[0056] The internal memory may include a computer-readable medium such as a volatile memory, a random access memory (RAM), and/or a non-volatile memory, for example, a read-

only memory (ROM) or a flash RAM. The internal memory is an example of the computer-readable medium.

5 **[0057]** The computer-readable medium includes non-volatile and volatile media as well as removable and non-removable media, and can store information by using any method or technology. The information may be a computer-readable instruction, a data structure, and a module of a program, or other data. The computer-readable medium includes, for example but not limited to, a phase-change RAM (PRAM), a static RAM (SRAM), a dynamic RAM (DRAM), or other types of RAMs, a ROM, an electrically erasable programmable ROM (EEPROM), a flash memory or other memory technologies, a compact disk ROM (CD-ROM), a digital versatile disc
10 (DVD) or other optical storages, a cassette tape, a magnetic disk storage, a quantum memory, a graphene-based storage medium or other magnetic storage devices, or any other non-transmission medium, and can be used to store information accessed by a computing device. As defined herein, the computer-readable medium does not include transitory media, such as modulated data signals and carriers.

15 **[0058]** The descriptions above are merely example embodiments of this specification, and are not intended to limit this specification. Any modification, equivalent replacement, improvement, etc. made without departing from the spirit and principles of this specification shall fall within the protection scope of this specification.

CLAIMS

What is claimed is:

1. A timestamp allocation method for a distributed database, wherein nodes of the distributed database comprise a timing node and an ordinary node, the method is applied to any node, and the

5 method comprises:

committing a corresponding to-be-timed transaction in response to a received transaction commission request, and recording a start-to-commit moment of the to-be-timed transaction;

querying for a timing request sending moment at which the any node sends a timing request to the timing node for another transaction; and

10 when a timing request sending moment of any other transaction is not earlier than the start-to-commit moment of the to-be-timed transaction, allocating, to the to-be-timed transaction, a timestamp fed back by the timing node for the any other transaction.

2. The method according to claim 1, wherein the querying for a timing request sending moment at which the any node sends a timing request to the timing node for another transaction

15 comprises:

querying for a timing request sending moment at which the any node sends a timing request to the timing node for a latest timed transaction, wherein

when the found timing request sending moment is not earlier than the start-to-commit moment of the to-be-timed transaction, the latest timed transaction is determined as the any other

20 transaction.

3. The method according to claim 2, wherein the querying for a timing request sending moment at which the any node sends a timing request to the timing node for another transaction

comprises:

when the timing request sending moment of the latest timed transaction is earlier than the
25 start-to-commit moment of the to-be-timed transaction, querying for a timing request sending moment at which the any node sends a timing request to the timing node for another untimed transaction, wherein

when a timing request sending moment of any other untimed transaction is not earlier than the start-to-commit moment of the to-be-timed transaction, the any other untimed transaction is

30 used as the any other transaction.

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

when timing request sending moments of all other transactions are earlier than the start-to-commit moment of the to-be-timed transaction, sending a timing request to the timing node for the to-be-timed transaction, so that the timing node allocates a timestamp to the to-be-timed transaction in response to the timing request.

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

receiving any timing response returned by the timing node; and

when a timing request sending moment corresponding to the any timing response is later than a timing request sending moment of a latest timed transaction in a cache, updating the timing request sending moment of the latest timed transaction in the cache to the timing request sending moment corresponding to the any timing response.

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

receiving any timing response returned by the timing node, wherein the any timing response comprises a current running state of the timing node, and the current running state of the timing node indicates a busy degree of the timing node; and

adjusting, based on the current running state of the timing node, a frequency at which the any node sends a timing request to the timing node.

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

traversing all to-be-timed transactions in a to-be-timed queue in a cache, and allocating a timestamp of a latest timed transaction in the cache to all to-be-timed transactions that meet a condition, wherein the condition comprises that a timing request sending moment of the latest timed transaction is not earlier than start-to-commit moments of the to-be-timed transactions.

8. A timestamp allocation apparatus for a distributed database, wherein nodes of the distributed database comprise a timing node and an ordinary node, the apparatus is applied to any node, and the apparatus comprises:

a commission module, configured to commit a corresponding to-be-timed transaction in response to a received transaction commission request, and record a start-to-commit moment of the to-be-timed transaction;

a query module, configured to query for a timing request sending moment at which the any node sends a timing request to the timing node for another transaction; and

an allocation module, configured to: when a timing request sending moment of any other

transaction is not earlier than the start-to-commit moment of the to-be-timed transaction, allocate, to the to-be-timed transaction, a timestamp fed back by the timing node for the any other transaction.

9. An electronic device, comprising:

5 a processor; and
a memory configured to store a processor-executable instruction, wherein
the processor runs the executable instruction to implement the method according to any one of claims 1 to 7.

10 10. A computer-readable storage medium, storing a computer instruction, wherein the
instruction is executed by a processor to implement the method according to any one of claims 1 to 7.