

METHOD AND APPARATUS FOR UPDATING DATABASE BY USING TWO-PHASE COMMIT DISTRIBUTED TRANSACTION

TECHNICAL FIELD

[0001] The present specification relates to the field of network communication
5 technologies, and in particular, to a method and an apparatus for updating a database
by using a two-phase commit distributed transaction.

BACKGROUND

[0002] Transactions provide a mechanism, in which all operations involved in an
activity can be incorporated into an indivisible execution unit. All operations that make
10 up a transaction can only be committed if all operations can be executed normally. A
failure of any of these operations causes the entire transaction to abort. In distributed
transactions, all parties participating in transaction processing are usually located on
different network nodes. Two-phase commit (2PC) protocols are widely used to meet
the atomicity, consistency, isolation, and durability (ACID) properties of the distributed
15 transactions.

[0003] Large-scale websites often use distributed transactions to achieve high
concurrent access to databases. In some application scenarios, such as in a "seckill"
activity, intensive updates to hot rows occur in a very short period of time. The
performance of "seckill" is closely related to user experience. How to improve the
20 concurrent performance of distributed transactions used to update hot rows has become
particularly important.

SUMMARY

[0004] In view of this, the present specification provides a method for updating a
database by using a two-phase commit distributed transaction. The method is applied
25 to a participant, and includes: according to an instruction on updating a target database
record in a current transaction, performing an update operation according to the
instruction after adding a write lock to the target database record; using another
transaction, in a transaction information record, whose state is to release the write lock

in advance and whose update object involves the target database record as a predecessor transaction of the current transaction, and recording the predecessor transaction of the current transaction in the transaction information record; writing a log of the update operation of the current transaction to a log buffer after receiving a phase 1 prepare request message sent by a coordinator, and sending a write lock advance release request to the coordinator after the writing succeeds; releasing the write lock after receiving a write lock advance release response returned by the coordinator, and setting a state of the current transaction in the transaction information record to releasing the write lock in advance; after persisting the log of the update operation of the current transaction in the log buffer succeeds, sending a phase 1 prepare response message of preparation success to the coordinator after all predecessor transactions of the current transaction are in a commit state or sending a phase 1 prepare response message of preparation failure to the coordinator after any predecessor transaction of the current transaction is in an abort state; and setting the current transaction in the transaction information record to a commit state or an abort state based on an execution result of the current transaction in phase 2.

[0005] The present specification provides a method for updating a database by using a two-phase commit distributed transaction. The method is applied to a coordinator, and includes: sending a phase 1 prepare request message to each participant to end a current transaction, the current transaction being used to update a target database record; and after receiving write lock advance release requests sent by all participants, returning a write lock advance release response to each participant to instruct each participant to unlock a target database record of the participant, the write lock advance release requests being sent by the participants after the participants successfully write a log of an update operation of the current transaction to a log buffer.

[0006] The present specification further provides an apparatus for updating a database by using a two-phase commit distributed transaction. The apparatus is applied to a participant, and includes: an update operation execution unit, configured to: according to an instruction on updating a target database record in a current transaction, perform an update operation according to the instruction after adding a write lock to the target database record; a predecessor transaction discovery unit, configured to: use another transaction, in a transaction information record, whose state is to release the write lock in advance and whose update object involves the target database record as a predecessor transaction of the current transaction, and record the predecessor

transaction of the current transaction in the transaction information record; a write lock release request unit, configured to: write a log of the update operation of the current transaction to a log buffer after receiving a phase 1 prepare request message sent by a coordinator, and send a write lock advance release request to the coordinator after the writing succeeds; a write lock advance release unit, configured to: release the write lock after receiving a write lock advance release response returned by the coordinator, and set a state of the current transaction in the transaction information record to releasing the write lock in advance; a first prepare response sending unit, configured to: after persisting the log of the update operation of the current transaction in the log buffer succeeds, send a phase 1 prepare response message of preparation success to the coordinator after all predecessor transactions of the current transaction are in a commit state or send a phase 1 prepare response message of preparation failure to the coordinator after any predecessor transaction of the current transaction is in an abort state; and a commit-abort state unit, configured to set the current transaction in the transaction information record to a commit state or an abort state based on an execution result of the current transaction in phase 2.

[0007] The present specification provides an apparatus for updating a database by using a two-phase commit distributed transaction. The apparatus is applied to a coordinator, and includes: a prepare request sending unit, configured to send a phase 1 prepare request message to each participant to end a current transaction, the current transaction being used to update a target database record; and a write lock release response unit, configured to: after receiving write lock advance release requests sent by all participants, return a write lock advance release response to each participant to instruct each participant to unlock a target database record of the participant, the write lock advance release requests being sent by the participants after the participants successfully write a log of an update operation of the current transaction to a log buffer.

[0008] The present specification provides a computer device, including a memory and a processor; the memory storing a computer program that can be run by the processor; and when the processor runs the computer program, the steps of the method, applied to the participant, for updating a database by using a two-phase commit distributed transaction being performed.

[0009] The present specification provides a computer device, including a memory and a processor; the memory storing a computer program that can be run by the processor; and when the processor runs the computer program, the steps of the method,

applied to the coordinator, for updating a database by using a two-phase commit distributed transaction being performed.

[0010] The present specification provides a computer readable storage medium, storing a computer program; when the computer program is run by a processor, the steps of the method, applied to the participant, for updating a database by using a two-phase commit distributed transaction being performed.

[0011] The present specification further provides a computer readable storage medium, storing a computer program; when the computer program is run by a processor, the steps of the method, applied to the coordinator, for updating a database by using a two-phase commit distributed transaction being performed.

[0012] As can be learned from the above technical solutions, in implementations of the present specification, in the prepare phase in phase 2 commitment, the participants send the write lock advance release requests to the coordinator after the log of the update operation is successfully written to the log buffer. After receiving the write lock advance release request from each participant of the distributed transaction, the coordinator returns the write lock advance release response, so that each participant releases the write lock in advance. In this way, a locking time of the target database record is greatly reduced, so that another transaction used to update the same target database record can be performed in advance, thereby improving the performance of concurrent processing. In addition, each participant maintains the state of the current transaction and the predecessor transaction of the current transaction in the transaction information record, and returns a phase 1 preparation success or failure message to the coordinator after the predecessor transaction is in the commit state or the abort state, thereby implementing the ACID properties of the distributed transaction.

BRIEF DESCRIPTION OF DRAWINGS

[0013] FIG. 1 is a schematic diagram illustrating an interactive procedure between a coordinator and a participant in the 2PC protocol;

[0014] FIG. 2 is an example sequence diagram illustrating that participants concurrently update a database record in an existing technology;

[0015] FIG. 3 is a flowchart illustrating a method, applied to a participant, for updating a database by using a two-phase commit distributed transaction according to an implementation of the present specification;

[0016] FIG. 4 is a flowchart illustrating a method, applied to a coordinator, for updating a database by using a two-phase commit distributed transaction according to an implementation of the present specification;

5 [0017] FIG. 5 is a schematic diagram illustrating an interactive procedure between a coordinator and a participant after ending of a transaction is initiated in an application example of the present specification;

[0018] FIG. 6 is an example sequence diagram illustrating that participants concurrently update a database record in an application example of the present specification;

10 [0019] FIG. 7 is a hardware structural diagram illustrating a device that implements an implementation of the present specification;

[0020] FIG. 8 is a logical structural diagram illustrating an apparatus, applied to a participant, for updating a database by using a two-phase commit distributed transaction according to an implementation of the present specification; and

15 [0021] FIG. 9 is a logical structural diagram illustrating an apparatus, applied to a coordinator, for updating a database by using a two-phase commit distributed transaction according to an implementation of the present specification.

DESCRIPTION OF IMPLEMENTATIONS

20 [0022] For a distributed transaction used to update a database record (where the updated database record is referred to as a target database record), a coordinator initiates execution of a current transaction after determining several participants executing the current transaction. The coordinator sends an instruction on updating the target database record to each participant. For the participants, an execution process includes the following four steps:

25 [0023] Step A1: The participants parse an operation command of updating the database record, to generate an execution plan used to execute the operation command.

[0024] Step A2: Add a write lock to the target database record, and perform an update operation on the target database record according to the execution plan, where the update operation on the database record can be at least one of modification, deletion, 30 or addition of a data item in the target database record, may be at least one of addition or deletion of an entry of the database record, or may be another operation that needs to be performed on the target database record in an exclusive way.

[0025] When an initiator of the current transaction instructs the coordinator to end the transaction (in other words, when the initiator of the current transaction commits the current transaction, where the commitment herein is referred to as ending the transaction to distinguish from the phase 2 commitment in 2PC), the coordinator starts
5 a procedure of 2PC, and sends a phase 1 prepare request message to each participant. 2PC is a type of distributed consensus protocol used to commit a transaction. The protocol includes two phases of operations: the first phase is the prepare phase, and the second phase is a commit/abort phase. An interactive procedure between a coordinator and a participant in the 2PC protocol is shown in FIG. 1.

10 **[0026]** For the participants, step A3: After receiving the phase 1 prepare request message, the participants write a log of the update operation to a log buffer.

[0027] Step A4: Persist the log of the update operation in the log buffer, that is, permanently store the log written to the log buffer on a medium. For example, a locally stored log file is to be written to a log file on a local hard drive. For another example,
15 in a scenario in which a log file has a plurality of backup files, the log of the update operation in the log buffer is synchronized to the plurality of backup files. After the log persistence succeeds, the participants return phase 1 prepare response messages of preparation success ("Prepare OK") to the coordinator. If not all the operations in steps A1 to A3 are successfully performed or the log persistence fails, the participants return
20 phase 1 prepare response messages of preparation failure ("Prepare No") to the coordinator. The participants perform two-phase commit or abort based on a commit or abort message sent by the coordinator in phase 2, to write a commit or abort log of the current transaction.

[0028] After completing step A4, the participants unlock the target data record.

25 **[0029]** In the above four steps, a locking time of each participant for the target database record is from step A2 to step A4. The operations consuming the longest time are the log persistence in step A4 and a synchronization process performed when the transaction is ended according to the 2PC protocol. Especially for a large-scale website, a log file thereof usually has a plurality of backups, which are often located in different
30 cities. An inevitable transmission delay makes log persistence take a very long time. In a case of centralized and concurrent transactions used to update the same target database record, because steps A2 to A4 on a subsequent transaction can be performed only after the target database record is unlocked for a previous transaction, the performance is severely affected. Assuming that transactions T1, T2, and T3 are to use the same row

Row1 in the database as an object of the update operation, an execution sequence at the maximum possible concurrency of T1, T2, and T3 is shown in FIG. 2.

[0030] An implementation of the present specification provides a method for updating a database by using a two-phase commit distributed transaction. When

5 participants execute phase 1 prepare request messages in the prepare phase, the participants send write lock advance release requests to a coordinator after writing a log of an update operation to a log buffer in step A3. After receiving the write lock advance release requests from all the participants, the coordinator returns a write lock advance release response to each participant. The participants unlock a target database record
10 based on the write lock advance release responses, and do not wait for log persistence in step A4 and the synchronization process in the 2PC protocol. Therefore, a locking time for the target database record is greatly shortened, and the performance of concurrent processing is improved. In addition, based on a state of a transaction of each participant that is maintained in a transaction information record, and dependency
15 between a predecessor transaction and a successor transaction, the ACID properties of a distributed transaction is not affected when a transaction is aborted.

[0031] In the implementation of the present specification, in a distributed system, a coordinator can communicate with one or more participants via a network. The coordinator or the participant can run on any device having a computing capability and
20 a storage capability, such as a mobile phone, a tablet computer, a personal computer (PC), a notebook computer, or a server; or various functions of the coordinator or the participant in the implementation of the present specification can be implemented by logic nodes running on two or more devices. In addition, a coordinator and one or more participants can be located on the same physical device. This is not limited herein.

25 **[0032]** In the implementations of the present specification, a procedure in which a method for updating a database by using a 2PC distributed transaction is applied to a participant is shown in FIG. 3, and a procedure in which the method is applied to a coordinator is shown in FIG. 4. The distributed transaction is used to update a database record.

30 **[0033]** For the participants, step S310: According to an instruction on updating a target database record in a current transaction, perform an update operation according to the instruction after adding a write lock to the target database record.

[0034] As described above, after initiating execution of the current transaction, the coordinator sends the instruction on updating the target database record to each

participant. The participants determine a to-be-updated target database record according to the instruction, add the write lock to the target database record, and after the write lock is successfully added, perform, on the target database record, the update operation specified in the instruction.

- 5 **[0035]** In the implementation of the present specification, the database record can be a table, a page, a row, or the like in the database, or an entry or a combination of entries stored in another database. This is not limited herein.

10 **[0036]** For the participants, step S320: Use another transaction, in a transaction information record, whose state is to release the write lock in advance and whose update object involves the target database record as a predecessor transaction of the current transaction, and record the predecessor transaction of the current transaction in the transaction information record.

15 **[0037]** Each participant maintains an independent transaction information record, which is used to record various information of a distributed transaction executed by the participant, including but not limited to an identifier of the transaction, a status of the transaction, and the like. The status of the transaction includes an execution result (to be specific, commit or abort) in phase 2, and can further include a prepare result (to be specific, preparation success or preparation failure) in phase 1. In the implementation of the present specification, the predecessor transaction of the current transaction is recorded in the transaction information record, and a state of releasing the write lock in advance is added to the status of the transaction.

20 **[0038]** When starting to execute a certain transaction, the participant writes information about the transaction to the transaction information record. In the implementation of the present specification, after determining the target database record according to the received instruction, the participant searches the transaction information record for another transaction whose update operation involves the target database record (that is, the database updated when the update operation is performed includes at least a part of the target database record) and whose state is to release the write lock in advance. If such a transaction is found, such a transaction is recorded in the transaction information record as a predecessor transaction of the current transaction.

25 The current transaction is a successor transaction of each predecessor transaction. In some application scenarios, for each predecessor transaction of the current transaction, the current transaction can further be recorded in the transaction information record as a successor transaction of the predecessor transaction.

30

[0039] In the implementation of the present specification, the current transaction can have one or more predecessor transactions or no predecessor transaction.

[0040] The write lock added to the target database record can be any lock that can prevent the update operation from being performed by another transaction on the target database record, such as a mutex lock and an exclusive lock, which are collectively referred to as a write lock in the implementations of the present specification.

[0041] For the coordinator, step S410: Send, to the participant, a phase 1 prepare request message to end the current transaction; the current transaction being used to update the target database record.

10 **[0042]** After receiving, from an initiator of the current transaction, an instruction on ending the transaction, the coordinator starts the 2PC protocol, and sends the phase 1 prepare request message of the current transaction to each participant.

[0043] For the participants, step S330: Write a log of the update operation of the current transaction to a log buffer after receiving the phase 1 prepare request message sent by the coordinator, and send write lock advance release requests to the coordinator after the writing succeeds.

[0044] If the participants fail to write the log of the update operation of the current transaction to the log buffer, the participants do not send the write lock advance release requests to the coordinator.

20 **[0045]** For the coordinator, step S420: After receiving write lock advance release requests sent by all participants, return a write lock advance release response to each participant to instruct each participant to unlock a target database record of the participant, the write lock advance release requests being sent by the participants after the participants successfully write the log of the update operation of the current transaction to the log buffer.

[0046] For the participants, step S340: Release the write lock after receiving a write lock advance release response returned by the coordinator, and set a state of the current transaction in the transaction information record to releasing the write lock in advance.

30 **[0047]** The coordinator receives the write lock advance release requests sent by the participants. For one transaction, if the coordinator has successfully collected write lock advance release requests sent by all participants of the transaction, the coordinator returns the write lock advance release response to each participant. The write lock advance release response is used to instruct each participant of the transaction to unlock

the target database record in advance, and the participant of the transaction does not need to unlock the target database record after step A4 is completed.

[0048] If the coordinator does not receive the write lock advance release requests sent by one or more participants, the coordinator does not perform processing.

- 5 **[0049]** If the participants receive the write lock advance release response sent by the coordinator, the participants unlock the target database record regardless of whether log persistence is completed, and record, in the transaction information record, the state of the current transaction as releasing the write lock in advance.

- 10 **[0050]** If the participants do not receive the write lock advance release request sent by the coordinator, the participants unlock the target database record after step A4 is completed.

- [0051]** For the participants, step S350: After persisting the log of the update operation of the current transaction in the log buffer succeeds, send phase 1 prepare response messages of preparation success to the coordinator after all predecessor transactions of the current transaction are in a commit state or send phase 1 prepare response messages of preparation failure to the coordinator after any predecessor transaction of the current transaction is in an abort state.

- 20 **[0052]** After the persisting the log of the update operation in the log buffer is completed, if the current transaction has one or more predecessor transactions, the participants query a state of each predecessor transaction in the transaction information record. The participants send the phase 1 prepare response messages of preparation success to the coordinator after all the predecessor transactions are in the commit state; or the participants send the phase 1 prepare response messages of preparation failure to the coordinator after any predecessor transaction is in the abort state.

- 25 **[0053]** If the current transaction has no predecessor transaction, the participants send the phase 1 prepare response messages of preparation success to the coordinator after the persisting the log of the update operation of the current transaction in the log buffer succeeds. Similar to the existing 2PC protocol, if the participants fail in any one of steps A1 to A3 or the persisting the log of the update operation of the current transaction fails, 30 the participants send the phase 1 prepare response messages of preparation failure to the coordinator.

[0054] For the participants, step S360: Set the current transaction in the transaction information record to a commit state or an abort state based on an execution result of the current transaction in phase 2.

[0055] In phase 2 (the commit/abort phase), if messages received by the coordinator from all the participants are phase 1 prepare response messages of preparation success, the coordinator sends a commit message to each participant. If any participant returns preparation failure, the coordinator sends an abort message to all the participants. The participants commit or abort the current transaction in phase 2 based on the commit message or the abort message received from the coordinator, and write the state of the current transaction in the transaction information record as a commit or abort state.

[0056] In the implementation of the present specification, after transaction TA releases the write lock in advance, another transaction (denoted as transaction TS) can add the write lock to the same target database record to perform an update operation. When transaction TS adds the write lock, if a state of transaction TA that is recorded in the transaction information record is to release the write lock in advance instead of the commit or abort state, it indicates that whether the update operation of transaction TA on the target database record is finally effective is still pending. If transaction TA is finally aborted, the target database record before being updated by using transaction TS is very likely to be incorrect (in other words, dirty data has been updated), and transaction TS also needs to be aborted. A success or failure of transaction TS depends on a final execution result of transaction TA. Therefore, in the implementation of the present specification, dependency between transaction TS and transaction TA is indicated by using transaction TA as a predecessor transaction of transaction TS and using transaction TS as a successor transaction of transaction TA, and the dependency is recorded in the transaction information record. Whether preparation of a successor transaction succeeds or fails is determined after a final state of a predecessor transaction of the successor transaction is determined, and then a phase 1 prepare response message is sent to a coordinator of the successor transaction.

[0057] The predecessor transaction is found by the successor transaction. To be specific, before performing the update operation on the target database record, the successor transaction searches the transaction information record for the predecessor transaction whose object of the update operation involves the target database record and whose state is to release the write lock in advance. The dependency between the successor transaction and the predecessor transaction is recorded in the transaction information record. After two-phase commitment/abortion on each transaction in 2PC is completed, the state of the current transaction is updated in the transaction information record. In this way, the successor transaction of the current transaction can

determine to send the phase 1 prepare response message of preparation success or preparation failure to the coordinator in time.

[0058] In some application scenarios, various information about a transaction can be recorded by using an entry (for example, a row in a database table) of the transaction information record. In these application scenarios, after the successor transaction discovers the predecessor transaction of the current transaction, when the predecessor transaction of the current transaction is recorded in the transaction information record, each predecessor transaction and a status of each predecessor transaction can be recorded in an entry of the current transaction, and it is written in the entry of each predecessor transaction that the current transaction is the successor transaction of the predecessor transaction. When used to update the state of the current transaction in the transaction information record based on an execution result in the commit/abort phase, each transaction not only updates the state of the current transaction in the entry of the current transaction, but also updates the state of the current transaction in an entry of each successor transaction to a commit or abort state if the current transaction has a successor transaction. In this way, when waiting for a final state of the predecessor transaction, the successor transaction only needs to query the entry of the current transaction in the transaction information record, and does not need to repeatedly retrieve the entry of each predecessor transaction, thereby reducing resource occupation and accelerating a response speed.

[0059] In the running process of the implementation of the present specification, if the predecessor transaction consumes a relatively long time in step A4, there can be a plurality of layers of cascading successor transactions (where cascading means that the predecessor transaction of the current transaction is a successor transaction of another transaction). In practice, an operation that takes more time than a normal operation is more likely to fail. In this way, once a predecessor transaction having a plurality of layers of cascading successor transactions is aborted, all layers of successor transactions depending on the predecessor transaction need to be aborted. This not only hinders improvement of the concurrent processing performance, but also affects the stability of the distributed system.

[0060] To reduce the possibility of the occurrence of the above situation, after receiving the write lock advance release requests sent by all the participants, the coordinator can generate a commit timestamp of the current transaction, and add the generated commit timestamp to the write lock advance release response for sending to

each participant. The commit timestamp is used to measure a start time of log persistence on the participants of the current transaction. The participants record the commit timestamp of the current transaction in the transaction information record. When the phase 1 prepare request message is executed, commit timestamps of all predecessor transactions of the successor transaction can be found in the transaction information record, and the probability of occurrence of a plurality of layers of cascading abortion is reduced by using one or two of the following implementations:

[0061] In a first implementation, before the participants perform the update operation on the target database record according to the instruction, a difference between a timestamp of the current update operation and a commit timestamp of each predecessor transaction of the current transaction is calculated. If the difference between the timestamp of the current update operation and a commit timestamp of a certain predecessor transaction or commit timestamps of some predecessor transactions reaches or exceeds a first threshold, the participants perform the update operation according to the instruction after the predecessor transaction or these predecessor transactions are in the commit state or the abort state.

[0062] In a second implementation, before the participants write the log of the update operation of the current transaction to the log buffer, a difference between a timestamp of writing the log of the update operation of the current transaction to the log buffer and a commit timestamp of each predecessor transaction of the current transaction is calculated. If the difference between the timestamp and a commit timestamp of a certain predecessor transaction or commit timestamps of some predecessor transactions reaches or exceeds a second threshold, the participants write the log of the update operation of the current transaction to the log buffer after the predecessor transaction or these predecessor transactions are in the commit state or the abort state.

[0063] In the above two implementations, for a start time, measured by the commit timestamp, at which the participants perform log persistence, because an exact start time of each participant is different, the coordinator can use an approximate time as the commit timestamp, for example, can use a moment at which the coordinator receives the write lock advance release requests sent by all the participants as the commit timestamp, can use a moment at which the coordinator sends the write lock advance release response as the commit timestamp, or can use another moment as the commit timestamp. This is not limited herein. In an example, each participant can record the timestamp of writing the log of the update operation of the current transaction to the

log buffer, and encapsulate the timestamp into the write lock advance release request for sending to the coordinator. After receiving the write lock advance release requests sent by all the participants, the coordinator uses the latest timestamp in timestamps, in all the write lock advance release requests, of writing the log of the update operation to the log buffer as the commit timestamp of the current transaction.

[0064] In addition, the first threshold and the second threshold can be determined with reference to factors such as duration required by log persistence in an actual application scenario and tolerance for cascading abortion. This is not limited herein. For example, in a scenario in which a log file has backups in different places, several times (for example, two to four times) a normal transmission delay between two places can be used as the first threshold or the second threshold.

[0065] When a device on which the participant is located is restarted due to breakdown, in a recovery process, a status of a committed or aborted transaction can be recovered by using the log. For a transaction that has not been committed or aborted yet, the procedure in 2PC needs to be continued until the transaction is committed or aborted. In the recovery process, if the transaction that has not been committed or aborted yet has a predecessor transaction, an execution result of a successor transaction can only be determined with reference to a commit/abort state of the predecessor transaction. For a primary-secondary backup application scenario, the situation is similar after primary-secondary switching. To resolve a problem of data correctness in the recovery process, each participant can persist a state of the predecessor transaction of the current transaction in the transaction information record (for example, writing all predecessor transactions of the current transaction to the redo log) and a state of the current transaction. In this way, after the restart due to breakdown or the primary-secondary switching, the recovery process can be performed based on dependency between transactions and the state of the predecessor transaction. If the predecessor transaction is aborted, in the recovery process, the successor transaction can learn, from persisted information, that the predecessor transaction is aborted, and the current transaction is not committed based on dirty data.

[0066] In addition, for a read-only transaction used to access the target database record, the read-only transaction can be enabled to read an update result of a committed transaction for the target database record, and update performed on the target database record by using the transaction that has not been committed or aborted yet is ignored.

In this way, abortion of the read-only transaction can be avoided, facilitating the stability of the distributed system.

[0067] As can be learned, in implementations of the present specification, in the prepare phase, the participants send the write lock advance release requests to the coordinator after the log of the update operation is successfully written to the log buffer. After receiving the write lock advance release request from each participant of the distributed transaction, the coordinator returns the write lock advance release response, so that each participant releases the write lock in advance, and does not wait for the log persistence process. In this way, a locking time of the target database record is reduced, so that another transaction used to update the same target database record can be performed in advance, thereby improving the performance of concurrent processing. In addition, a state of a transaction of each participant and dependency between a predecessor transaction and a successor transaction are maintained in the transaction information record, so that the ACID properties of the distributed transaction are not affected when the transaction is aborted.

[0068] The above describes the specific implementations of the present specification. Other implementations are within the scope of the appended claims. In some cases, the actions or steps described in the claims can be performed in an order different from that in the implementations and still can achieve desired results. In addition, the processes described in the accompanying drawings do not necessarily require the specific shown order or the sequential order to achieve the desired results. In some implementations, multitasking and parallel processing are also possible or can be advantageous.

[0069] In an application example of the present specification, a large-scale business website uses a distributed system to update a plurality of associated databases when deals occur. The "seckill" activity poses a challenge to the concurrent update performance of a hot row in the database. The business website implements, based on the 2PC protocol by using the following method, a distributed transaction used to update the hot row. After ending of a transaction is initiated, an interactive procedure between a coordinator and a participant can be shown in FIG. 5.

[0070] Each participant maintains a transaction information registration table (a transaction information record). Each transaction in the transaction information registration table corresponds to a row of record. Data items of this row of record include: an identifier of the current transaction, a commit timestamp of the current transaction, an identifier of a predecessor transaction of the current transaction, a state

of the predecessor transaction of the current transaction, an identifier of a successor transaction of the current transaction, and the like.

5 **[0071]** A log file of the distributed transaction of the business website has three backups, which are located in three different cities. The longest transmission delay between the three cities is 100 milliseconds (ms). In this application example, the first threshold and the second threshold are both set to 200 ms.

10 **[0072]** After the distributed transaction used to update the hot row is started, the coordinator sends an instruction on updating a target database record to each participant, where the instruction includes the identifier of the current transaction, and a database row to be updated by each participant and a specific update operation on this row are specified.

15 **[0073]** The participant receives the instruction on updating the target database record, and parses the to-be-updated target from the instruction. The participant queries the transaction information registration table maintained by the participant, and uses another transaction, in the transaction information registration table, whose state is to release the write lock in advance and whose update object is the target row as a predecessor transaction of the current transaction. The participant writes an entry (a row of record) of the current transaction to the transaction information registration table, where the entry includes the identifier of the current transaction, an identifier of each predecessor transaction of the current transaction, and a state (which currently is to release the write lock in advance) of each predecessor transaction; and writes the identifier of the current transaction to the entry of each predecessor transaction in the transaction information registration table as a successor transaction of the predecessor transaction.

20 **[0074]** The participant adds the write lock to the target row. The participant queries a commit timestamp of each predecessor transaction from the transaction information registration table, obtains a timestamp of the current update operation on the target row, and calculates a difference between the two timestamps. If the difference exceeds 200 ms, the participant pauses or waits. The participant performs the specific update operation on the target row after the predecessor transaction, in the transaction information registration table, with a difference exceeding 200 ms enters the commit or abort state. If none of the differences reaches 200 ms, the participant performs the specific update operation on the target row.

[0075] After the instruction of the current transaction is ended, the coordinator sends a phase 1 prepare request message of the current transaction to each participant.

[0076] The participants receive the phase 1 prepare request message of the current transaction, obtain the timestamp of writing the log of the current update operation to the log buffer, and calculate a difference between the timestamp of writing the log of the current update operation to the log buffer and the commit timestamp of each predecessor transaction. If the difference exceeds 200 ms, the participants pause or wait. The participants write the log of the current update operation to the log buffer after a predecessor transaction, in the transaction information registration table, with a difference exceeding 200 ms enters the commit or abort state. If none of the differences reaches 200 ms, the participants write the log of the current update operation to the log buffer.

[0077] After the log of the current update operation is successfully written to the log buffer, the participants encapsulate the timestamp of writing the log of the current update operation to the log buffer into the write lock advance release requests for sending to the coordinator.

[0078] After receiving the write lock advance release requests sent by all the participants, the coordinator uses the latest timestamp in timestamps, in all the write lock advance release requests, of writing the log of the update operation to the log buffer as the commit timestamp of the current transaction. The coordinator adds the commit timestamp of the current transaction to the write lock advance release response for sending to each participant.

[0079] After receiving the write lock advance release response, the participants write the log commit time of the current transaction in the write lock advance release response to the transaction information registration table, and unlock the write lock of the target row.

[0080] After successfully synchronizing the log of the update operation of the current transaction to three backup log files in the log buffer, the participants query a state of each predecessor transaction in the entry of the current transaction in the transaction information registration table, and send phase 1 prepare response messages of preparation success to the coordinator after all predecessor transactions are in the commit state, or send phase 1 prepare response messages of preparation failure to the coordinator after any predecessor transaction is in the abort state.

[0081] In the commit/abort phase, if messages received by the coordinator from all the participants are phase 1 prepare response messages of preparation success, the coordinator sends a commit message to each participant. If any participant returns preparation failure, the coordinator sends an abort message to all the participants. The participants commit or abort the current transaction in phase 2 based on the commit message or the abort message received from the coordinator, write the state of the current transaction in the transaction information record as a commit or abort state, and also write the state of the current transaction in an entry of the successor transaction of the current transaction as the commit or abort state.

[0082] The participants persist the state of the current transaction and the identifier of the predecessor transaction of the current transaction in the transaction information registration table into the log file, so that the data correctness in the recovery process after restart due to breakdown or primary-secondary switching is ensured.

[0083] In this application example, assuming that transactions T1, T2, and T3 of a certain participant are to use the same row Row1 in the database as an object of the update operation, an execution sequence at the maximum possible concurrency of T1, T2, and T3 is shown in FIG. 6. It can be seen that the concurrency performance has been greatly improved compared with FIG. 2.

[0084] Corresponding to implementation of the above procedure, the implementations of the present specification further provide an apparatus, applied to a participant, for updating a database by using a two-phase commit distributed transaction and an apparatus, applied to a coordinator, for updating a database by using a two-phase commit distributed transaction. The two apparatuses each can be implemented by software or a combination of software and hardware. An example in which the apparatus is implemented by software is used. As a logical apparatus, the apparatus is formed in a way in which a central processing unit (CPU) of the device on which the apparatus is located reads a corresponding computer program instruction and runs the instruction in a storage. From a hardware perspective, in addition to a CPU, a storage, and a memory shown in FIG. 7, the device on which the apparatus for updating a database by using a two-phase commit distributed transaction is located usually further includes at least one of other hardware such as a chip for wireless signal transmission and reception, or other hardware such as a board used to implement a network communication function.

[0085] FIG. 8 shows an apparatus for updating a database by using a two-phase commit distributed transaction provided in an implementation of the present specification. The apparatus is applied to a participant, and includes: an update operation execution unit, a predecessor transaction discovery unit, a write lock release request unit, a write lock advance release unit, a first prepare response sending unit, and a commit-abort state unit. The update operation execution unit is configured to: according to an instruction on updating a target database record in a current transaction, perform an update operation according to the instruction after adding a write lock to the target database record. The predecessor transaction discovery unit is configured to: use another transaction, in a transaction information record, whose state is to release the write lock in advance and whose update object involves the target database record as a predecessor transaction of the current transaction, and record the predecessor transaction of the current transaction in the transaction information record. The write lock release request unit is configured to: write a log of the update operation of the current transaction to a log buffer after receiving a phase 1 prepare request message sent by a coordinator, and send a write lock advance release request to the coordinator after the writing succeeds. The write lock advance release unit is configured to: release the write lock after receiving a write lock advance release response returned by the coordinator, and set a state of the current transaction in the transaction information record to releasing the write lock in advance. The first prepare response sending unit is configured to: after persisting the log of the update operation of the current transaction in the log buffer succeeds, send a phase 1 prepare response message of preparation success to the coordinator after all predecessor transactions of the current transaction are in a commit state or send a phase 1 prepare response message of preparation failure to the coordinator after any predecessor transaction of the current transaction is in an abort state. The commit-abort state unit is configured to set the current transaction in the transaction information record to a commit state or an abort state based on an execution result of the current transaction in phase 2.

[0086] Optionally, the apparatus further includes: a second prepare response sending unit, configured to: when the current transaction has no predecessor transaction, send the phase 1 prepare response message of preparation success to the coordinator after the persisting the log of the update operation of the current transaction in the log buffer succeeds.

[0087] Optionally, the write lock advance release response carries a commit timestamp generated by the coordinator. The apparatus further includes: a commit timestamp writing unit, configured to record a commit timestamp of the current transaction in the transaction information record. The performing, by the update operation execution unit, the update operation according to the instruction includes: when a difference between a timestamp of the current update operation and a commit timestamp of the predecessor transaction of the current transaction reaches or exceeds a first threshold, performing the update operation according to the instruction after the predecessor transaction is in the commit state or the abort state.

10 [0088] Optionally, the write lock advance release response carries a commit timestamp generated by the coordinator. The apparatus further includes: a commit timestamp writing unit, configured to record a commit timestamp of the current transaction in the transaction information record. The writing, by the write lock advance release unit, the log of the update operation of the current transaction in the log buffer includes: when a difference between a timestamp of writing the update operation of the current transaction to the log buffer and a commit timestamp of the predecessor transaction of the current transaction reaches or exceeds a second threshold, writing the log of the update operation of the current transaction to the log buffer after the predecessor transaction is in the commit state or the abort state.

15 [0089] Optionally, the write lock advance release request carries a timestamp of writing the log of the update operation of the current transaction to the log buffer, for the coordinator to generate a commit timestamp of the current transaction.

20 [0090] Optionally, the recording, by the predecessor transaction discovery unit, the predecessor transaction of the current transaction in the transaction information record includes: writing each predecessor transaction of the current transaction and a status of the predecessor transaction to an entry of the current transaction in the transaction information record, and recording, in an entry of each predecessor transaction of the current transaction, the current transaction as a successor transaction of the predecessor transaction. The commit-abort state unit is specifically configured to: based on the execution result of the current transaction in a commit/abort phase, update a state in the entry of the current transaction in the transaction information record into the commit state or the abort state, and update a state of the current transaction in an entry of a successor transaction of the current transaction to the commit state or the abort state.

25

30

[0091] Optionally, the apparatus further includes: a predecessor transaction persistence unit, configured to persist states of the predecessor transaction of the current transaction and the current transaction in the transaction information record.

5 [0092] FIG. 9 shows an apparatus for updating a database by using a two-phase commit distributed transaction provided in an implementation of the present specification. The apparatus is applied to a coordinator, and includes a prepare request sending unit and a write lock release response unit. The prepare request sending unit is configured to send a phase 1 prepare request message to each participant to end a current transaction, the current transaction being used to update a target database record. The
10 write lock release response unit is configured to: after receiving write lock advance release requests sent by all participants, return a write lock advance release response to each participant to instruct each participant to unlock a target database record of the participant, the write lock advance release requests being sent by the participants after the participants successfully write a log of an update operation of the current transaction
15 to a log buffer.

[0093] In an example, the apparatus further includes: a commit timestamp generation unit, configured to generate a commit timestamp of the current transaction after receiving the write lock advance release requests sent by all the participants, where the write lock advance release response carries a commit timestamp of the current
20 transaction.

[0094] In the above example, the write lock advance release requests carry timestamps at which the participants sending the advance release requests write the log of the update operation of the current transaction to the log buffer. The generating, by the commit timestamp generation unit, the commit timestamp of the current transaction
25 includes: using the latest timestamp in the timestamps, at which the log is written to the log buffer, in the advance release requests sent by all the participants as the commit timestamp of the current transaction.

[0095] An implementation of the present specification provides a computer device. The computer device includes a memory and a processor. The memory stores a
30 computer program that can be run by the processor. When the processor runs the stored computer program, the steps of the method, applied to the participant, for updating a database by using a two-phase commit distributed transaction in the implementations of the present specification are performed. For a detailed description of each step of the method, applied to the participant, for updating a database by using a two-phase commit

distributed transaction, reference can be made to the above content. Details are omitted herein for simplicity.

[0096] An implementation of the present specification provides a computer device. The computer device includes a memory and a processor. The memory stores a computer program that can be run by the processor. When the processor runs the stored computer program, the steps of the method, applied to the coordinator, for updating a database by using a two-phase commit distributed transaction in the implementations of the present specification are performed. For a detailed description of each step of the method, applied to the coordinator, for updating a database by using a two-phase commit distributed transaction, reference can be made to the above content. Details are omitted herein for simplicity.

[0097] An implementation of the present specification provides a computer readable storage medium. The storage medium stores computer programs. When the computer programs are run by a processor, the steps of the method, applied to the participant, for updating a database by using a two-phase commit distributed transaction in the implementations of the present specification are performed. For a detailed description of each step of the method, applied to the participant, for updating a database by using a two-phase commit distributed transaction, reference can be made to the above content. Details are omitted herein for simplicity.

[0098] An implementation of the present specification provides a computer readable storage medium. The storage medium stores computer programs. When the computer programs are run by a processor, the steps of the method, applied to the coordinator, for updating a database by using a two-phase commit distributed transaction in the implementations of the present specification are performed. For a detailed description of each step of the method, applied to the coordinator, for updating a database by using a two-phase commit distributed transaction, reference can be made to the above content. Details are omitted herein for simplicity.

[0099] The above shows merely preferred implementations of the present specification, and is not intended to limit other embodiments claimed for protection. Any modification, equivalent replacement, improvement, and the like made within the spirit and principle of the present specification shall fall within the protection scope.

[0100] In a typical configuration, a computing device includes one or more processors (CPU), an input/output interface, a network interface, and a memory.

[0101] The memory can include at least one of a non-permanent memory in a computer readable medium, a random access memory (RAM), or a non-volatile memory, such as a read-only memory (ROM) or a flash memory (flash RAM). The memory is an example of the computer readable medium.

5 **[0102]** The computer readable medium includes permanent and non-permanent media and removable and non-removable media, and information can be stored by using any method or technology. The information can be a computer readable instruction, a data structure, a program module, or other data. Examples of the computer storage medium include, but are not limited to, a phase change random access memory
10 (PRAM), a static random access memory (SRAM), a dynamic random access memory (DRAM), another type of random access memory (RAM), a read-only memory (ROM), an electrically erasable programmable read-only memory (EEPROM), a flash memory or another memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD) or another optical storage, a magnetic cassette, a disk storage, a
15 tape disk storage or another magnetic storage device or any another non-transmission medium, which can be configured to store information that can be accessed by the computing device. According to the definition in the present specification, the computer readable medium does not include transitory media, such as a modulated data signal and carrier.

20 **[0103]** It should further be noted that the terms "comprise", "include" or any other variants thereof are intended to cover non-exclusive inclusion, so that a process, method, commodity or device including a series of elements not only includes those elements, but also includes other elements that are not explicitly listed, or also includes elements inherent to such a process, method, commodity, or device. If there are no more
25 restrictions, the element defined by the sentence "including a..." does not exclude the existence of other identical elements in the process, method, commodity, or device that includes the element.

[0104] A person skilled in the art should understand that the implementations of the present specification can provide a method, a system, or a computer program product.
30 Therefore, the implementations of the present specification can be in a form of complete hardware implementations, complete software implementations, or software and hardware combined implementations. In addition, the implementations of the present specification can be in a form of computer program products implemented on one or more computer usable storage media (including but not limited to a magnetic disk

storage, a CD-ROM, an optical storage, and the like) that include computer usable program code.

CLAIMS

What is claimed is:

1. A method for updating a database by using a two-phase commit distributed transaction, the method being applied to a participant, and comprising:

performing an update operation according to an instruction on updating a target database record in a current transaction after adding a write lock to the target database record;

using another transaction, in a transaction information record, whose state is to release the write lock in advance and whose update object involves the target database record as a predecessor transaction of the current transaction, and recording the predecessor transaction of the current transaction in the transaction information record;

writing a log of the update operation of the current transaction to a log buffer after receiving a phase 1 prepare request message sent by a coordinator, and sending a write lock advance release request to the coordinator after the writing succeeds;

releasing the write lock after receiving a write lock advance release response returned by the coordinator, and setting a state of the current transaction in the transaction information record to releasing the write lock in advance;

after persisting the log of the update operation of the current transaction in the log buffer succeeds, sending a phase 1 prepare response message of preparation success to the coordinator after all predecessor transactions of the current transaction are in a commit state or sending a phase 1 prepare response message of preparation failure to the coordinator after any predecessor transaction of the current transaction is in an abort state; and

setting the current transaction in the transaction information record to a commit state or an abort state based on an execution result of the current transaction in phase 2,

wherein the recording the predecessor transaction of the current transaction in the transaction information record includes: writing each predecessor transaction of the current transaction and a status of the predecessor transaction to an entry of the current transaction in the transaction information record, and recording, in an entry of each predecessor transaction of the current transaction, the current transaction as a successor transaction of the predecessor transaction; and

the setting the current transaction in the transaction information record to the commit state or the abort state based on the execution result of the current transaction in phase 2 includes: based on the execution result of the current transaction in a commit/abort phase, updating a state in the entry of the current transaction in the transaction information record into the commit state or the abort state, and updating a state of the current transaction in an entry of a successor transaction of the current transaction to the commit state or the abort state.

2. The method according to claim 1, further comprising: when the current transaction has no predecessor transaction, sending the phase 1 prepare response message of preparation success to the coordinator after the persisting the log of the update operation of the current transaction in the log buffer succeeds.

3. The method according to claim 1, wherein the write lock advance release response carries a commit timestamp generated by the coordinator;

the method further comprises: recording a commit timestamp of the current transaction in the transaction information record; and

the performing the update operation according to the instruction includes: when a difference between a timestamp of the current update operation and a commit timestamp of the predecessor transaction of the current transaction reaches or exceeds a first threshold, performing the update operation according to the instruction after the predecessor transaction is in the commit state or the abort state.

4. The method according to claim 1, wherein the write lock advance release response carries a commit timestamp generated by the coordinator;

the method further comprises: recording a commit timestamp of the current transaction in the transaction information record; and

the writing the log of the update operation of the current transaction to the log buffer includes: when a difference between a timestamp of writing the update operation of the current transaction to the log buffer and a commit timestamp of the predecessor transaction of the current transaction reaches or exceeds a second threshold, writing the log of the update operation of the current transaction to the log buffer after the predecessor transaction is in the commit state or the abort state.

5. The method according to claim 1, wherein the write lock advance release request carries a timestamp of writing the log of the update operation of the current transaction to the log buffer, for the coordinator to generate a commit timestamp of the current transaction.

6. The method according to claim 1, further comprising: persisting states of the predecessor transaction of the current transaction and the current transaction in the transaction information record.

7. An apparatus for updating a database by using a two-phase commit distributed transaction, the apparatus being applied to a participant, and comprising:

a processor; and

a memory storing instructions executable by the processor,

wherein the processor is configured to:

perform an update operation according to an instruction on updating a target database record in a current transaction after adding a write lock to the target database record;

use another transaction, in a transaction information record, whose state is to release the write lock in advance and whose update object involves the target database record as a predecessor transaction of the current transaction, and record the predecessor transaction of the current transaction in the transaction information record;

write a log of the update operation of the current transaction to a log buffer after receiving a phase 1 prepare request message sent by a coordinator, and send a write lock advance release request to the coordinator after the writing succeeds;

release the write lock after receiving a write lock advance release response returned by the coordinator, and set a state of the current transaction in the transaction information record to releasing the write lock in advance;

after persisting the log of the update operation of the current transaction in the log buffer succeeds, send a phase 1 prepare response message of preparation success to the coordinator after all predecessor transactions of the current transaction are in a commit state or send a phase 1 prepare response message of preparation failure to the coordinator after any predecessor transaction of the current transaction is in an abort state; and

set the current transaction in the transaction information record to a commit state or an abort state based on an execution result of the current transaction in phase 2,

wherein in recording the predecessor transaction of the current transaction in the transaction information record, the processor is further configured to: write each predecessor transaction of the current transaction and a status of the predecessor transaction to an entry of the current transaction in the transaction information record,

and record, in an entry of each predecessor transaction of the current transaction, the current transaction as a successor transaction of the predecessor transaction; and

in setting the current transaction in the transaction information record, the processor is further configured to: based on the execution result of the current transaction in a commit/abort phase, update a state in the entry of the current transaction in the transaction information record into the commit state or the abort state, and update a state of the current transaction in an entry of a successor transaction of the current transaction to the commit state or the abort state.

8. The apparatus according to claim 7, wherein the processor is further configured to: when the current transaction has no predecessor transaction, send the phase 1 prepare response message of preparation success to the coordinator after the persisting the log of the update operation of the current transaction in the log buffer succeeds.

9. The apparatus according to claim 7, wherein the write lock advance release response carries a commit timestamp generated by the coordinator;

the processor is further configured to:

record a commit timestamp of the current transaction in the transaction information record; and

when a difference between a timestamp of the current update operation and a commit timestamp of the predecessor transaction of the current transaction reaches or exceeds a first threshold, perform the update operation according to the instruction after the predecessor transaction is in the commit state or the abort state.

10. The apparatus according to claim 7, wherein the write lock advance release response carries a commit timestamp generated by the coordinator;

the processor is further configured to:

record a commit timestamp of the current transaction in the transaction information record; and

when a difference between a timestamp of writing the update operation of the current transaction to the log buffer and a commit timestamp of the predecessor transaction of the current transaction reaches or exceeds a second threshold, write the log of the update operation of the current transaction to the log buffer after the predecessor transaction is in the commit state or the abort state.

11. The apparatus according to claim 7, wherein the write lock advance release request carries a timestamp of writing the log of the update operation of the current

transaction to the log buffer, for the coordinator to generate a commit timestamp of the current transaction by using the timestamp.

12. The apparatus according to claim 7, wherein the processor is further configured to persist states of the predecessor transaction of the current transaction and the current transaction in the transaction information record.