

Method, Device, Transaction Node and Storage Medium for Processing Transaction Request

FIELD

The present application relates to the field of computer, and in particular to a method, device, transaction node and storage medium for processing transaction request.

BACKGROUND

The typical use scenario of raft algorithm is, a leader node replicates the request from a client to other nodes, and then each node processes the request independently. However, in the case of a transaction system, its input includes not only client requests but also market data. The latter is volatile, i.e., frequently updated. Therefore, when applying this approach, there is a high probability that the market data will be outdated after the request is replicated. For example, the leader node receives a request from the client, assuming that it is a conditional buy order with time point t_0 , and the time point at which raft replicates the request and commits it successfully is t_1 , then the market data obtained during processing is the market data at moment t_1 , while what the user wants is the market data at moment t_0 when the request is received by the system. In a market with relatively large volatility, the market prices at t_0 and t_1 will differ significantly from each other. That is, if each node processes this conditional buy order at the same time, the processing results of each node may be different due to the inconsistency of asset prices during processing, resulting in inconsistent states among nodes, such as inconsistent states of order books, funds, etc.

SUMMARY

In view of the above deficiencies or shortcomings, the present application provides a method and device for processing transaction request, computer equipment and storage medium. The embodiments of the present application can ensure that the state of other transaction nodes in the transaction system is only related to the processing result of the leader node without being affected by the high frequent market data updates, thus ensure that the state is always consistent among the transaction nodes in the transaction system.

According to a first aspect, the present application provides a method for processing a transaction request. In an embodiment, the method is applied to any one of the transaction nodes in the transaction system, the transaction system comprises a plurality of the transaction nodes; the method comprises:

upon being elected as a leader node in the plurality of transaction nodes, receiving a first transaction request from a client, obtaining latest market data from a market data server based on the first transaction request, processing the first transaction request to obtain a transaction processing result based on the market data;

replicating the transaction processing result to other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

In an embodiment, wherein each of the transaction nodes in the transaction system is configured with a service layer and a data replication layer;

wherein the receiving the first transaction request from the client, obtaining latest market data from the market data server based on the first transaction request, processing the first transaction request to obtain the transaction processing result based on the market data comprises:

receiving the first transaction request from the client via the service layer of current node, obtaining the latest market data from the market data server based on the first transaction request, and processing the first transaction request to obtain the transaction processing result based on the market data;

wherein the replicating the transaction processing result to the other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines comprises:

replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

In an embodiment, wherein the receiving the first transaction request from the client via the service layer of the current node is followed by:

obtaining a first node role and a first term from the data replication layer of the current node via the service layer of the current node;

obtaining a first historical term last obtained from the data replication layer of the current node;

detecting whether the first node role is a leader node and whether the first historical term is equal to the first term;

wherein the obtaining the latest market data from the market data server based on the first transaction request comprises:

upon detecting the first node role as a leader and the first historical term equal to the first term, obtaining the latest market data from the market data server based on the first transaction request via the service layer of the current node.

In an embodiment, the method further comprises:

upon detecting the first node role not as a leader and the first historical term not equal to the first term, rejecting the first transaction request from the client via the service layer of the current node.

In an embodiment, wherein the transaction processing result carries the first term;

the replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node is preceded by:

extracting the first term carried in the transaction processing result via the data replication layer of the current node;

obtaining a second node role and a second term;

detecting whether the second node role is the leader node and whether the first term carried in the transaction processing result is equal to the second term;

wherein the replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of current node comprises:

upon detecting the second node role as a leader and the first term equal to the second term, replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node.

In an embodiment, the method further comprises:

upon detecting the second node role not as a leader and the first term not equal to the second term, discarding the transaction processing result from the service layer of the current node, and notifying the client that processing of the first transaction request has failed.

In an embodiment, the method further comprises:

obtaining node information of current leader node via the data replication layer of the current node;

sending the node information of the current leader node to the client via the service layer of the current node, causing the client to send a transaction request to the current leader node based on the node information of the current leader node.

In an embodiment, wherein the replicating the transaction processing result to the other transaction nodes in the transaction system comprises:

appending the transaction processing result as a new log entry to local raft log;

replicating the transaction processing result to the other transaction nodes in the transaction system based on log replication mechanism of raft algorithm.

In an embodiment, the method further comprises:

obtaining latest snapshot locally after a reboot, or from a remote storage system after a boot from a new machine;

using the transaction processing result corresponding to the latest snapshot as a starting position and current latest generated transaction processing result as an ending position, and applying all transaction processing results from the starting position to the ending position in sequence;

joining a raft cluster consisting of the plurality of the transaction nodes;

upon receiving the transaction processing result sent by the current leader node, applying the received transaction processing result.

In an embodiment, wherein the receiving the first transaction request from the client is preceded by:

obtaining a plurality of unprocessed transaction records from a cross system, the unprocessed transaction records referring to transaction records that have been closed and not yet processed;

processing each unprocessed transaction record, updating account asset information of a user corresponding to each unprocessed transaction record based on processing result of each unprocessed transaction record, recovering a state of the replicated state machine of the current node to a state at the time of a first ex-leader node stepped down.

In an embodiment, the method further comprises: when the current node is a non-leader node, in response to receiving a second transaction request from the client via the service layer of the current node, obtaining a third node role and a third term from the data replication layer of the current node via the service layer of the current node;

obtaining a second historical term last obtained from the data replication layer of the current node;

detecting whether the third node role is a leader node and whether the second historical term is equal to the third term;

upon detecting the third node role as a leader node and the second historical term equal to the third term, recovering a state of the replicated state machine of the current node to a state at the time of a second ex-leader node stepped down and processing the second transaction request;

upon detecting the third node role not as a leader and the second historical term not equal to the third term, rejecting the second transaction request.

In an embodiment, the method further comprises: initiating a connection to a cross system after the current node becomes a leader node to cause the cross system to send latest transaction records to the current node via the connection; and disconnecting the connection upon detecting the current node not as the leader node.

According to a second aspect, the present application provides a device for processing a transaction request. The device is applied to any one of the

transaction nodes in the transaction system, the transaction system comprises a plurality of the transaction nodes; the device comprises:

- a request processing module configured for, upon being elected as a leader node in the plurality of transaction nodes, receiving a first transaction request from a client, obtaining latest market data from a market data server based on the first transaction request, processing the first transaction request to obtain a transaction processing result based on the market data;

- a processing result replication module configured for replicating the transaction processing result to other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

In an embodiment, wherein the request processing module is configured with a service layer and the processing result replication module is configured with a data replication layer;

- the request processing module is specifically configured for receiving the first transaction request from the client via the service layer of current node, obtaining the latest market data from the market data server based on the first transaction request, and processing the first transaction request to obtain the transaction processing result based on the market data;

- the processing result replication module is specifically configured for replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

In an embodiment, wherein the request processing module, after being configured for receiving the first transaction request from the client via the service layer of the current node, is further configured for obtaining a first

node role and a first term from the data replication layer of the current node via the service layer of the current node; obtaining a first historical term last obtained from the data replication layer of the current node; detecting whether the first node role is a leader node and whether the first historical term is equal to the first term;

the request processing module, upon being configured for obtaining the latest market data from the market data server based on the first transaction request, is specifically configured for, upon detecting the first node role as a leader and the first historical term equal to the first term, obtaining the latest market data from the market data server based on the first transaction request via the service layer of the current node.

In an embodiment, wherein the request processing module is further configured for, upon detecting the first node role not as a leader and the first historical term not equal to the first term, rejecting the first transaction request from the client via the service layer of the current node.

In an embodiment, wherein the transaction processing result carries the first term;

the processing result replication module, before being configured for replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, is further configured for extracting the first term carried in the transaction processing result via the data replication layer of the current node; obtaining a second node role and a second term; detecting whether the second node role is the leader node and whether the first term carried in the transaction processing result is equal to the second term;

the processing result replication module, upon being configured for replicating the transaction processing result to the other transaction nodes in

the transaction system via the data replication layer of the current node, is specifically configured for, upon detecting the second node role as a leader and the first term equal to the second term, replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node.

In an embodiment, wherein the processing result replication module is further configured for, upon detecting the second node role not as a leader and the first term not equal to the second term, discarding the transaction processing result from the service layer of the current node, and notifying the client that processing of the first transaction request has failed.

In an embodiment, wherein the processing result replication module is further configured for obtaining node information of current leader node via the data replication layer of the current node;

the request processing module is further configured for sending the node information of the current leader node to the client via the service layer of the current node, causing the client to send a transaction request to the current leader node based on the node information of the current leader node.

In an embodiment, wherein the processing result replication module, upon being configured for replicating the transaction processing result to the other transaction nodes in the transaction system, is specifically configured for appending the transaction processing result as a new log entry to local raft logs; replicating the transaction processing result to the other transaction nodes in the transaction system based on log replication mechanism of raft algorithm.

In an embodiment, the device further comprises the following modules:

a snapshot acquisition module configured for obtaining latest snapshot locally after a reboot, or from a remote storage system after a boot from a new machine;

a first result application module configured for using the transaction processing result corresponding to the latest snapshot as a starting position and current latest generated transaction processing result as an ending position, and applying all transaction processing results from the starting position to the ending position in sequence;

a cluster joining module configured for joining a raft cluster consisting of the plurality of the transaction nodes;

a second result application module configured for applying transaction processing result received from the current leader node.

In an embodiment, the device further comprises the following modules:

a transaction record acquisition module configured for obtaining a plurality of unprocessed transaction records from a cross system, the unprocessed transaction records referring to transaction records that have been closed and not yet processed;

a transaction record processing module configured for processing each unprocessed transaction record, updating account asset information of a user corresponding to each unprocessed transaction record based on processing result of each unprocessed transaction record, recovering a state of the replicated state machine of the current node to a state at the time of a first ex-leader node stepped down.

In an embodiment, wherein the request processing module is further configured for performing the following steps:

when the current node is a non-leader node, in response to receiving a second transaction request from the client via the service layer of the current node, obtaining a third node role and a third term from the data replication layer of the current node via the service layer of the current node;

obtaining a second historical term last obtained from the data replication

layer of the current node;

detecting whether the third node role is a leader node and whether the second historical term is equal to the third term;

upon detecting the third node role as a leader node and the second historical term equal to the third term, recovering a state of the replicated state machine of the current node to a state at the time of a second ex-leader node stepped down and processing the second transaction request;

upon detecting the third node role not as a leader and the second historical term not equal to the third term, rejecting the second transaction request.

In an embodiment, the device further comprising a connection management module;

wherein the connection management module is configured for initiating a connection to a cross system after the current node becomes a leader node to cause the cross system to send latest transaction records to the current node via the connection; and disconnecting the connection upon detecting the current node not as the leader node.

According to a third aspect, the present application provides a transaction node comprising a memory, a processor, and a computer program stored on the memory and executable on the processor, wherein the processor performs the steps of any embodiment of the above methods when executing the computer program.

According to a fourth aspect, the present application provides a computer readable storage medium on which a computer program is stored, wherein the computer program when executed by a processor performs the steps of any embodiment of the above methods.

In the above embodiments of the present application, upon any transaction node included by the transaction system is elected as a leader node, it receives

a first transaction request from a client, obtains latest market data from a market data server based on the first transaction request, processes the first transaction request to obtain a transaction processing result based on the market data, and replicates the transaction processing result to other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines. In the embodiments of the present application, by first allowing the leader node to process the received transaction request and then replicating the transaction processing result resulting from processing the transaction request to other transaction nodes, it can be ensured that the state of other transaction nodes in the transaction system is only related to the processing result of the leader node without being affected by the high frequency of market data updates, thus ensuring that the state is always consistent among the transaction nodes in the transaction system.

The embodiments of the present application further provide an automatic leader election function and a state recovery function for the transaction system. On the basis of ensuring consistency among the states of each transaction node in the transaction system, the restarted or newly started transaction node can quickly recover its state, and then join the transaction system to ensure that the majority or more than half of the transaction nodes in the transaction system will be in normal operation. And when the current leader node steps down, for example, when the current leader node is shut down for upgrade, a new leader node is automatically elected. Compared with other existing systems, this system can improve the availability of the transaction system, can run 7x24 uninterruptedly, and can always provide services to clients online.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of an application environment for a method for processing transaction request in an embodiment;

FIG. 2 is a schematic flowchart for a method for processing transaction request in an embodiment;

FIG. 3 is a schematic diagram of the processing of a transaction processing request in an embodiment;

FIG. 4 is a schematic diagram of redirecting a client to the current leader node in an embodiment;

FIG. 5 is a schematic diagram of a rolling upgrade in an embodiment;

FIG. 6 is a schematic flowchart of recovering the state of a transaction node after reboot in an embodiment;

FIG. 7 is a schematic diagram of a transaction node obtaining the latest snapshot in an embodiment;

FIG. 8 is a schematic diagram of the application of the latest snapshot by a transaction node in an embodiment;

FIG. 9 is a schematic flowchart of recovering the state of a leader node in an embodiment;

FIG. 10 is a schematic diagram of the interaction between a transaction node and a cross system in an embodiment;

FIG. 11 is a structural block diagram of a device for processing transaction request in an embodiment; and

FIG. 12 is a diagram of the internal structure of a computer equipment in an embodiment.

DETAILED DESCRIPTION

To make the objectives, technical solutions and advantages of the present

application clearer, the present application will be further described in detail below with reference to the accompanying drawings and embodiments. It should be understood that the specific embodiments described herein are merely intended to explain the present application and are not intended to limit the present application.

The present application provides a method for processing transaction request. In this embodiment, the method for processing transaction request may be applied in an application environment as shown in FIG. 1, where reference number 10 refers to a client, reference number 20 refers to a transaction system, and reference number 30 refers to a market data server. The transaction system includes a number of transaction nodes, such as transaction nodes 21, 22 and 2Nd as shown in FIG. 1, which form a raft cluster. The transaction system elects a leader node from the number of transaction nodes based on the raft algorithm. The leader node refers to the transaction node in the number of transaction nodes whose node role is leader. Among them, in order to improve disaster tolerance to further ensure uninterrupted operation of the transaction system, the transaction nodes in the transaction system can be distributed in different data centers; in addition, the number of transaction nodes in the transaction system is usually an odd number, i.e., the above N is a positive integer and an odd number.

A user can send a transaction request to the transaction system through a client, and a leader node in the transaction system is responsible for processing the transaction request from the client. Specifically, the leader node, after receiving the transaction request from the client, obtains the latest market data from the market data server based on the transaction request, and processes the transaction request to obtain the transaction processing result based on the market data, and then replicates the transaction processing result to other

transaction nodes in the transaction system, so that the other transaction nodes apply the transaction processing result to their own replicated state machines. Where, the other transaction nodes refer to the transaction nodes other than the leader node in the transaction system.

The client may include, but are not limited to, various personal computers, laptops, smartphones, tablets, and desktop computers, etc. Each transaction node and market data server may be implemented as an independent server or a server cluster consisting of multiple servers.

The method for processing transaction request provided in this embodiment may be applied to any one of the transaction nodes in the transaction system in FIG. 1, and when any one of the transaction nodes is successfully elected as a leader node among a number of transaction nodes included in the transaction system by the election mechanism of the raft algorithm, the following steps shown in FIG. 2 are performed:

S110: receiving a first transaction request from a client, obtaining latest market data from a market data server based on the first transaction request, processing the first transaction request to obtain a transaction processing result based on the market data;

S120: replicating the transaction processing result to other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

Wherein, the first transaction request is a transaction request received by a transaction node after becoming the leader node.

The steps shown in FIG. 2 are explained in detail below.

After any transaction node becomes the leader node, it is responsible for processing transaction requests from the client, and upon receiving a transaction request from the client, it obtains the corresponding market data

from the market data server based on the transaction request (the market data is the latest market data pushed by the market data server; possibly, the market data may also be the latest market data requested by the transaction node from the market data server), and processes the transaction request based on the market data to obtain the transaction processing result. Exemplarily, the process of processing a transaction request by the leader node will be described below in conjunction with FIG. 3. Take the user creating a conditional buy order as an example. Suppose the user sends a transaction request to the leader node via his client (e.g., Web Client or FIX Client in FIG. 3), i.e., he wants to buy a certain amount of asset A at a predetermined price P. After receiving the transaction request, the leader node obtains the latest price of the asset A pushed by the market data server (e.g., Market Data Service in FIG. 3) and determines whether to generate a conditional order and place it in the user's order book or to generate a cross request and send it to the cross system (e.g., Cross Node Cluster in FIG. 3), based on the latest price of the asset A. The result of this judgment is the transaction processing result. Where if it is finally determined as the former, then the user's order book in the state of the leader node will contain an additional Conditional Order (such order will only be triggered after reaching preset conditions, the trader needs to specify a Trigger Price as a condition to activate the order), and part of the funds will be freezed (or locked down). And if it is finally determined as the latter, the order book will contain an additional Market Order (which refers to the immediate purchase or sale of a certain amount of the underlying asset at the best price or market price at the time), and the leader node will send a cross order (or cross request) to the cross system. The cross system will send the corresponding transaction record (Result) to the leader node after the cross order is closed, so that the leader node will process the transaction record, update the user's

account balance, assets and other information, as well as instruct the Wallet Order Service to update the user's account balance and assets.

Further, the leader node replicates the generated transaction processing result to other transaction nodes in the transaction system, and the other transaction nodes, upon receiving the transaction processing result, apply them to their own replicated state machines. When the leader node replicates the transaction processing result to other transaction nodes in the transaction system, the transaction processing result is appended to the local raft log as a new log entry; then the transaction processing result is replicated to other transaction nodes in the transaction system based on the log replication mechanism of the raft algorithm. Please refer to existing raft algorithm for the log replication mechanism of the raft algorithm, which will not be described in detail in this embodiment.

In this embodiment, since the function of the apply is deterministic, this embodiment provides a way to let the leader node process the received transaction request first and then replicate the transaction processing result generated by processing the transaction request to other transaction nodes, compared to the existing way of letting the leader node send the received transaction request to other transaction nodes first and then each transaction node processes the transaction request respectively, it can ensure that the state of other transaction nodes in the transaction system is only related to the processing result of the leader node, and is not affected by the high update frequency of market data, thus ensuring that the state of each transaction node in the transaction system is always consistent.

On the other hand, the present embodiment achieves faster and more accurate response to the user's transaction request while ensuring consistent state among the transaction nodes. Exemplarily, assuming that the leader node

receives the transaction request from the client at time point t_0 , the leader node in this embodiment can directly respond to the transaction request, whereas if the existing raft algorithm is employed, then the leader node needs to first replicate the transaction request to other transaction nodes and wait for the successful commit of the transaction request before responding to the transaction request. In addition, if the transaction request is successfully committed at time point t_1 , then the leader node will acquire the market data at moment t_1 when processing the transaction request. In a volatile market, the market price at moment t_1 that the leader node acquires may not be the market data at t_0 that the user wants. For example, the market price of the asset that the user wants to buy at moment t_0 may be 10, but the market price of the asset at moment t_1 may be updated to 20.

In an embodiment, DDD (Domain Driven Design) and Event sourcing are utilized to model the replicated state machine of each transaction node in the transaction system.

Specifically, one can discuss with domain experts to get a ubiquitous language, and then extract the key objects in the domain based on that language, and the main behaviors of each object, and then implement it with computer programming language, such as using C++ classes to represent the objects, class variables to represent the properties of the objects, and class functions to represent the behaviors of the objects. There are two special classes of objects in the domain object, wherein the first class is called Command object, i.e. Command, which refers to an intention, for example TradeAssetCommand, which indicates that it wants the system to trade assets; the second class is called Event object, i.e. Event, which represents the result of the command processing, for example AssetTradedEvent, which is a fact that has happened and cannot be changed. Further, the function that processes

the Command is called Process, which is a non-deterministic function, which may return a successful result, namely AssetTradedEvent, or may return a failed result, namely AssetTradeFailedEvent. The function that applies the Event is called Apply, which is a deterministic function, i.e., given the current state and Event, the output state is deterministic, regardless of how many times it runs. The system designed for Event sourcing will process the request in two steps. The first step, Process Command, which generates Event, does not change the system state; the second step, Apply Event, which changes the system state. In other words, the leader node receives the transaction request, processes the transaction request to generate Event, i.e., the transaction processing result, and then replicates the transaction processing result to other transaction nodes through the log replication mechanism of raft algorithm, so that other transaction nodes can apply the transaction processing result through the Apply function and update the state of their own replicated state machine.

This embodiment uses DDD and Event sourcing to model the replicated state machine, so that in one aspect, it can ensure correct business; in another aspect, it can make all records auditable, which is especially important to financial systems; in yet another aspect, when the transaction system is down, it can recover 100% to the state before it is down by replaying the historical events (Event); finally, for a non-event sourcing system, the processing of Command and Event is tightly coupled. If there is a downtime, it is difficult to recover to the state before the downtime; in this embodiment, if there is a bug in the system, it can be reproduced by replaying the Event at the moment of the problem.

Further, the prior art will outsource the leader election function to a third-party middleware such as ETCD (a distributed KV storage system developed by CoreOS). Since the service layer and ETCD are separated by a

network, a temporary network outage may cause a split brain issue, resulting in multiple leader nodes providing services at the same time, which may lead to state inconsistency among transaction nodes. To address this issue, the present application combines the leader election function and the service into one, and ensures that only one leader node is providing service to the client at the same time, thereby ensuring correctness. This will be explained in detail below.

In an embodiment, the step of S110: the receiving the first transaction request from the client, obtaining latest market data from the market data server based on the first transaction request, processing the first transaction request to obtain the transaction processing result based on the market data specifically comprises: receiving the first transaction request from the client via the service layer of current node, obtaining the latest market data from the market data server based on the first transaction request, and processing the first transaction request to obtain the transaction processing result based on the market data.

The step of S120: the replicating the transaction processing result to the other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines specifically comprises: replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

In this embodiment, each transaction node in the transaction system is configured with a service layer and a data replication layer. Specifically, the service layer and the data replication layer are configured in each transaction node. Where the service layer corresponds to transaction processing logic and can be used, for example, to receive user transaction requests and decide when

to send transaction information to the cross system, etc. The data replication layer is configured to synchronize and replicate the transaction processing results from the service layer to other transaction nodes. The two layers, the service layer and the data replication layer, can be viewed as two components or C++ classes running in different threads, with the service layer calling the functions of the data replication layer.

In an embodiment, the above-mentioned receiving the first transaction request from the client via the service layer of the current node is followed by: obtaining a first node role and a first term from the data replication layer of the current node via the service layer of the current node; obtaining a first historical term last obtained from the data replication layer of the current node; detecting whether the first node role is a leader node and whether the first historical term is equal to the first term.

Accordingly, the above-mentioned obtaining the latest market data from the market data server based on the first transaction request specifically comprises: upon detecting the first node role as a leader and the first historical term equal to the first term, obtaining the latest market data from the market data server based on the first transaction request via the service layer of the current node.

Accordingly, the method further comprises: upon detecting the first node role not as a leader and the first historical term not equal to the first term, rejecting the first transaction request via the service layer of the current node.

The node role of each transaction node includes leader, follower and candidate. The first historical term is the historical term acquired by the transaction node after becoming the leader node and before receiving the transaction request from the client.

In this embodiment, the service layer of any transaction node acquires the

current latest node role and term from its data replication layer each time it receives a transaction request from a client (and the current node role and term from the data replication layer are saved after each transaction request is processed). Then the service layer detects whether the current latest node role is leader, and detects whether the historical term (historical term is the term last obtained from the data replication layer) and the current latest term are equal (or whether they are consistent). If the current latest node role is not the leader, or the historical term is not equal to the current latest term, which means the current node is not the leader node, then the transaction request is rejected and the client will retry. If the current latest node role is leader, and the historical term and the current latest term are equal, which means the current node is the leader node, then the service layer of the current node will start processing the transaction request, and then generate the transaction processing result, and then call the data replication layer to replicate the transaction processing result to other transaction nodes.

In this embodiment, each transaction node, after becoming a leader node, if receiving a transaction request, checks whether the node role of the current node and the corresponding term satisfy the preset conditions (i.e., the node role is leader, and the historical term is equal to the term obtained at the data replication layer) through the service layer of the current node, and processes the transaction request from the client after confirming that the preset conditions are satisfied, so as to ensure that each transaction node processes a transaction request only if it becomes a leader node, avoiding the situation where there is more than one leader nodes in the transaction system providing services to the outside (referring to clients external to the system) at the same moment.

Further, the second node role and the second term are the latest ones from

the current data replication layer, while the first node role and the first term are obtained from the data replication layer by the service layer when processing the transaction request. And since the service layer and the data replication layer run independently in different threads, the detection in the service layer cannot solve the situation that occurs in a few cases where two transaction nodes both consider themselves as leader nodes at the same time and process the transaction request at the same time. To further avoid the situation where there is more than one leader node providing services to the clients at the same time, the present application further provides an embodiment in which a second-layer detection in the data replication layer during replication is configured to ensure that only the transaction processing results generated by the leader node can be replicated, while the transaction processing results generated by the non-leader node will be discarded, as well as to notify the client that the transaction requests it sent have failed to be processed.

In this embodiment, the above-mentioned replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node is preceded by: extracting the first term carried in the transaction processing result via the data replication layer of the current node; obtaining a second node role and a second term; detecting whether the second node role is the leader node and whether the first term carried in the transaction processing result is equal to the second term.

Accordingly, the replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of current node comprises: upon detecting the second node role as a leader and the first term equal to the second term, replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node.

Accordingly, the method further comprises: upon detecting the second node role not as a leader or the first term not equal to the second term, rejecting the first transaction request from the client via the service layer of the current node.

In this embodiment, considering that the service layer and the data replication layer run independently in their own threads, and the latter can trigger the leader election at any time due to network burst delay or the current leader node stepping down, downtime, etc. Thus, in some specific scenarios, the following situations may occur, for one scenario, the node role that the service layer of transaction node A acquires from the data replication layer at moment t1 is the leader (that is, transaction node A is the leader node at moment t1), however, at the subsequent moment t2 (e.g., the moment when the transaction processing results are generated and replicated), transaction node A has stepped down, i.e., the node role of transaction node A is not the leader at this time, if transaction node A continues to process transaction requests and replicate transaction processing results, it may provide services to the outside with the new leader node at the same time, which will cause system state inconsistency. The transaction node A needs to stop processing the transaction requests and replicating the transaction processing results; and for another scenario, the node role that the service layer of transaction node A acquires from the data replication layer at moment t1 is the leader with the term term_1, however, at the subsequent moment t2, transaction node A has stepped down and re-elected as the leader node, i.e., the node role is still leader, but the term has changed to term_2, at this time, the transaction node A needs to recover its state before continuing to process transaction requests and copy transaction processing results.

Specifically, when generating the transaction processing result, the

service layer adds the first term to the transaction processing result, such that the transaction processing result carries the first term.

When the data replication layer receives the call from the service layer, it first extracts the first term carried in the transaction processing result as well as obtains the current latest node role and term. It should be noted that in order to distinguish the node role and term currently obtained by the data replication layer from those obtained by the service layer as described above, the node role and term previously obtained by the service layer from the data replication layer are referred to as the first node role and the first term, and the node role and term currently obtained by the data replication layer are referred to as the second node role and the second term.

After that, the data replication layer detects whether the second node role is the leader, as well as detects whether the term carried in the transaction processing result, i.e., the first term, and the second term are equal. If the second node role is not the leader, or the second term and the first term are not equal, then the transaction processing result obtained this time is discarded, as well as notifying the client that the processing of the transaction request it sent failed, and letting the client to retry. If the current latest node role is leader and the term carried in the transaction processing result is equal to the current latest term, then the data replication layer replicates the transaction processing result to other transaction nodes.

Further, in an embodiment, the method further comprises: obtaining node information of current leader node via the data replication layer of the current node; sending the node information of the current leader node to the client via the service layer of the current node, causing the client to send a transaction request to the current leader node based on the node information of the current leader node.

In this embodiment, after the data replication layer of any transaction node finds out that it is not a leader node through detection, in order that the client can find the current leader node, it obtains the node information of the current leader node, such as IP address and port of the node, through the data replication layer of the current node, and then sends it to the client in order to redirect the client to the current leader node. The client can then continue to send transaction requests to meet the user's needs, and this process can be seen in FIG. 4.

In an embodiment, in order to ensure that the transaction system can provide services to the outside without interruption, this embodiment adopts a rolling upgrade method to upgrade each transaction node. The process of rolling upgrade can be referred to FIG. 5, i.e., first shutting down a transaction node in the raft cluster, then upgrading the transaction node, restarting the transaction node after upgrading, and making the transaction node perform state recovery, and the transaction node is added to the raft cluster after the state recovery, and then the rest of the transaction nodes are upgraded in the above method. Since most of the transaction nodes are still in normal operation during the whole process of rolling upgrade, the services can be provided to the outside normally. In order to enable the transaction nodes to recover their states faster, the present application provides a state recovery method, which will be described below.

In an embodiment, any transaction node performs the steps shown in FIG. 6 after a reboot, namely:

S210: obtaining latest snapshot locally after a reboot, or from a remote storage system after a boot from a new machine;

S220: using the transaction processing result corresponding to the latest snapshot as a starting position and current latest generated transaction

processing result as an ending position, and applying all transaction processing results from the starting position to the ending position in sequence;

S230: joining a raft cluster consisting of the plurality of the transaction nodes;

S240: upon receiving the transaction processing result sent by the current leader node, applying the received transaction processing result.

The process of obtaining the latest snapshot by the transaction node can be referred to FIG. 7, and the process of applying the latest snapshot by the transaction node can be referred to FIG. 8.

This embodiment provides a state recovery method for the transaction node, which can be applied in scenarios where the transaction node is started from a new machine, restarted after the transaction node is down, or restarted after the transaction node halted and upgraded during the rolling upgrade of the transaction system. Specifically, by using the snapshot technology, the transaction node does not need to apply sequentially from the first transaction processing result to the current latest generated transaction processing result, but only needs to start from the transaction processing result corresponding to the latest snapshot and apply sequentially the subsequent transaction processing results to the current latest generated transaction processing result, so that the transaction node can quickly recover its state after reboot and speed up the upgrade of the whole raft cluster. If the transaction node is restarted, the latest snapshot is obtained locally, while if the transaction node is started from a new machine, the latest snapshot is obtained from the remote storage system since there is no local snapshot.

Further, in an embodiment, the transaction node, after becoming the leader node and before receiving the first transaction request from the client, further performs the steps shown in FIG. 9, namely:

S250: obtaining a plurality of unprocessed transaction records from a cross system.

S260: processing each unprocessed transaction record, updating account asset information of a user corresponding to each unprocessed transaction record based on processing result of each unprocessed transaction record, recovering a state of the replicated state machine of the current node to a state at the time of a first ex-leader node stepped down.

Wherein the unprocessed transaction records referring to transaction records that have been closed and not yet processed. The first ex-leader refers to the previous leader node, and after it steps down, the current transaction node becomes the leader node through leader election and starts state recovery until the state when the first ex-leader node stepped down is recovered.

In this embodiment, any transaction node that becomes a leader node through election after reboot needs to recover the state (i.e., the state of its own replicated state machine) to the latest state, i.e., the state when the ex-leader node (meaning the previous leader of the current incumbent leader) stepped down, in order to be able to properly process transaction requests from clients. Specifically, referring to FIG. 10, after a transaction node becomes the leader node, it pulls the closed but unprocessed transaction records (such as the history data in FIG. 10) from the cross system, and then processes these transaction records, and updates the account asset information of the relevant user accordingly based on the processing results of these transaction records, such as the balance of the corresponding user, the number of assets purchased, and other information. Thus the state of the current node is recovered to the state of the ex-leader node when it stepped down, after which it can start operations such as processing transaction requests from clients, sending cross requests to the cross system (e.g., new order in FIG. 10), and processing the

latest transaction records pushed by the cross system (e.g., recent data in FIG. 10). This embodiment is able to accommodate the recover of cluster nodes with a balance of correctness and performance.

In an embodiment, the method further comprises: when the current node is a non-leader node, in response to receiving a second transaction request from the client via the service layer of the current node, obtaining a third node role and a third term from the data replication layer of the current node via the service layer of the current node;

obtaining a second historical term last obtained from the data replication layer of the current node;

detecting whether the third node role is a leader node and whether the second historical term is equal to the third term;

upon detecting the third node role as a leader node and the second historical term equal to the third term, recovering a state of the replicated state machine of the current node to a state at the time of a second ex-leader node stepped down and processing the second transaction request;

upon detecting the third node role not as a leader or the second historical term not equal to the third term, rejecting the second transaction request.

Where, the second transaction request is the transaction request received by a transaction node when it is a non-leader node. The non-leader node is another node type opposite to the leader node, specifically the transaction node whose node role is not the leader. The third node role and the third term are the node role and term obtained from the data replication layer after receiving a transaction request from a client, when a transaction node is a non-leader node. The second historical term is the historical term obtained after receiving a transaction request from the client when the transaction node is a non-leader node.

In this embodiment, since the service layer and the data replication layer run independently in different threads, the node role and the term of the data replication layer may change in some specific scenarios. Therefore, each transaction node, when being a non-leader node, will also check whether the node role of the current node and the corresponding term satisfy the preset conditions (i.e., the node role is leader, and the historical term and the current term from the data replication layer are equal) through the service layer of the current node if receiving a transaction request sent by the client. If the preset conditions are confirmed to be satisfied, then the current node has been transformed from a non-leader node to a leader node, then the state of the replicated state machine of the current node is first recovered to the state of the ex-leader node when it stepped down, and then process the transaction request from the client; if the preset conditions are confirmed not to be satisfied, then it means that the current node is still a non-leader node and needs to reject the received transaction requests.

When recovering the state of the current node to the state when the ex-leader node stepped down, the state recovery method provided in the above embodiment can be used, and the specific recovery process is described in the above embodiment, which will not be described in this embodiment. After rejecting the transaction request from the client, the node information of the current leader node can be sent to the client, so that the client can send a transaction request to the current leader node. See also the above-mentioned embodiment for the specific embodiment, which will not be described in this embodiment.

In an embodiment, the method further comprises: initiating a connection to a cross system after the current node becomes a leader node to cause the cross system to send latest transaction records to the current node via the

connection; and disconnecting the connection upon detecting the current node not as the leader node.

In this embodiment, each transaction node initiates a connection to the cross system after becoming a leader node through the election, and after the connection is established, the cross system can push the latest transaction record through that connection. Accordingly, if the transaction node steps down, for example, after receiving a transaction request, and verifies that the current node has changed from a leader node to a non-leader node (see the above embodiment for the verification process), then the cross system will be notified to disconnect. This embodiment ensures that the transaction records pushed by the cross system are processed by the leader node only.

FIGS. 2-4 show a flow diagram of a method for processing transaction request in an embodiment. It should be understood that although the steps in the flowchart of FIGs. 2-4 are shown sequentially as indicated by the arrows, the steps are not necessarily performed sequentially in the order indicated by the arrows. Except as expressly stated herein, there is no strict sequential limitation on the execution of these steps, and the steps may be executed in other orders. Moreover, at least some of the steps in FIGs. 2-4 may include multiple sub-steps or multiple stages. These sub-steps or stages can be performed at different times, other than at the same time, and the order in which these sub-steps or stages are performed is not necessarily sequential, but may be performed in turn or alternately with at least some of other steps, or the sub-steps or stages of other steps.

Based on the same inventive concept, the present application further provides a device for processing a transaction request. In this embodiment, as shown in FIG. 11, the device for processing a transaction request includes the following modules:

According to a second aspect, the present application provides a device for processing a transaction request. The device is applied to any one of the transaction nodes in the transaction system, the transaction system comprises a plurality of the transaction nodes; the device comprises:

a request processing module 110 configured for, upon being elected as a leader node in the plurality of transaction nodes, receiving a first transaction request from a client, obtaining latest market data from a market data server based on the first transaction request, processing the first transaction request to obtain a transaction processing result based on the market data;

a processing result replication module 120 configured for replicating the transaction processing result to other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

In an embodiment, the request processing module is configured with a service layer and the processing result replication module is configured with a data replication layer.

the request processing module is specifically configured for receiving the first transaction request from the client via the service layer of current node, obtaining the latest market data from the market data server based on the first transaction request, and processing the first transaction request to obtain the transaction processing result based on the market data;

the processing result replication module is specifically configured for replicating the transaction processing result to the other transaction nodes in

the transaction system via the data replication layer of the current node, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

In an embodiment, after being configured for receiving the first transaction request from the client via the service layer of the current node, the request processing module is further configured for: obtaining a first node role and a first term from the data replication layer of the current node via the service layer of the current node; obtaining a first historical term last obtained from the data replication layer of the current node; detecting whether the first node role is a leader node and whether the first historical term is equal to the first term.

The request processing module, upon being configured for obtaining the latest market data from the market data server based on the first transaction request, is specifically configured for, upon detecting the first node role as a leader and the first historical term equal to the first term, obtaining the latest market data from the market data server based on the first transaction request via the service layer of the current node.

In an embodiment, the request processing module is further configured for, upon detecting the first node role not as a leader and the first historical term not equal to the first term, rejecting the first transaction request from the client via the service layer of the current node.

In an embodiment the transaction processing result carries the first term.

The processing result replication module, before being configured for replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, is further configured for extracting the first term carried in the transaction processing result via the data replication layer of the current node; obtaining a

second node role and a second term; detecting whether the second node role is the leader node and whether the first term carried in the transaction processing result is equal to the second term.

The processing result replication module, upon being configured for replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, is specifically configured for, upon detecting the second node role as a leader and the first term equal to the second term, replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node.

In an embodiment, the processing result replication module is further configured for, upon detecting the second node role not as a leader and the first term not equal to the second term, discarding the transaction processing result from the service layer of the current node, and notifying the client that processing of the first transaction request has failed.

In an embodiment, the processing result replication module is further configured for obtaining node information of current leader node via the data replication layer of the current node.

The request processing module is further configured for sending the node information of the current leader node to the client via the service layer of the current node, causing the client to send a transaction request to the current leader node based on the node information of the current leader node.

In an embodiment, the processing result replication module, upon being configured for replicating the transaction processing result to the other transaction nodes in the transaction system, is specifically configured for appending the transaction processing result as a new log entry to local raft logs; replicating the transaction processing result to the other transaction nodes in

the transaction system based on log replication mechanism of raft algorithm.

In an embodiment, the device for processing a transaction request further includes the following modules:

- a snapshot acquisition module configured for obtaining latest snapshot locally after a reboot, or from a remote storage system after a boot from a new machine;

- a first result application module configured for using the transaction processing result corresponding to the latest snapshot as a starting position and current latest generated transaction processing result as an ending position, and applying all transaction processing results from the starting position to the ending position in sequence;

- a cluster joining module configured for joining a raft cluster consisting of the plurality of the transaction nodes;

- a second result application module configured for applying transaction processing result received from the current leader node.

In an embodiment, the device for processing a transaction request further includes the following modules:

- a transaction record acquisition module configured for obtaining a plurality of unprocessed transaction records from a cross system, the unprocessed transaction records referring to transaction records that have been closed and not yet processed;

- a transaction record processing module configured for processing each unprocessed transaction record, updating account asset information of a user corresponding to each unprocessed transaction record based on processing result of each unprocessed transaction record, recovering a state of the replicated state machine of the current node to a state at the time of a first ex-leader node stepped down.

In an embodiment, the request processing module is further configured for performing the following steps:

when the current node is a non-leader node, in response to receiving a second transaction request from the client via the service layer of the current node, obtaining a third node role and a third term from the data replication layer of the current node via the service layer of the current node;

obtaining a second historical term last obtained from the data replication layer of the current node;

detecting whether the third node role is a leader node and whether the second historical term is equal to the third term;

upon detecting the third node role as a leader node and the second historical term equal to the third term, recovering a state of the replicated state machine of the current node to a state at the time of a second ex-leader node stepped down and processing the second transaction request;

upon detecting the third node role not as a leader and the second historical term not equal to the third term, rejecting the second transaction request.

In an embodiment, the device further includes a connection management module.

wherein the connection management module is configured for initiating a connection to a cross system after the current node becomes a leader node to cause the cross system to send latest transaction records to the current node via the connection; and disconnecting the connection upon detecting the current node not as the leader node.

The specific limitations of the device for processing a transaction request can be referred to the limitations of the device for processing a transaction request above and will not be repeated here. The modules of the above device for processing a transaction request may be implemented in whole or in part by

software, hardware and combinations thereof. Each of the above modules may be embedded in or independent of the processor in the computer device in the form of hardware, or may be stored in the memory of the computer device in the form of software so that the processor can call and execute the operations corresponding to each of the above modules.

In an embodiment, a computer equipment is provided, the internal structure of which is illustrated in FIG. 12.

The computer equipment includes a processor, a memory, a network interface, and a database connected via a system bus. The processor of the computer equipment is configured to provide computing and control capabilities. The memory of the computer equipment includes a non-volatile storage medium and an internal memory. The non-volatile storage medium stores an operating system, a computer program, and a database. The internal memory provides an environment for the operation of the operating system and the computer program in the non-volatile storage medium. The database of the computer equipment is configured to store account information of the user, such as balance, assets, transaction orders and other data, the specific stored data may also be referred to the limitation in the above method embodiments. The network interface of the computer equipment is configured to communicate with an external terminal via a network connection. The computer program, when executed by the processor, implement the above method for processing transaction request.

It should be understood by those skilled in the art that the structure shown in FIG. 12 is only a block diagram of a part of the structure related to the solution of the present application, and does not constitute a limitation to the computer device to which the solution of the present application is applied. A specific computer device may include more or fewer components than shown

in the figure, or combine certain components, or have a different component arrangement.

This embodiment further provides a computer equipment including a memory, a processor, and a computer program stored in the memory and executable on the processor, and the processor executes the computer program to perform the steps contained in the method provided in any of the method embodiments described above.

In an embodiment, the present application provides a computer readable storage medium on which a computer program is stored, wherein the computer program when executed by a processor performs the steps contained in the method provided in any of the method embodiments described above.

Those skilled in the art will appreciate that all or part of the processes in the methods of the above-mentioned embodiments can be completed by instructing relevant hardware through computer readable instructions or computer program, and the computer readable instructions or computer program may be stored in a non-volatile computer readable storage medium which, when executed, may cause a processor to perform the processes of the embodiments of the methods described above. Any reference to memory, storage, database, or other media used in the various embodiments provided in the present application may include non-volatile and/or volatile memory. The non-volatile memory may include Read-only Memory (ROM), Programmable ROM (PROM), Electrically Programmable ROM (EPROM), Electrically Erasable Programmable ROM (EEPROM), or flash memory. The volatile memory may include Random Access Memory (RAM) or external cache memory. By way of illustration and not limitation, RAM is available in a variety of forms, such as Static RAM (SRAM), Dynamic RAM (DRAM), Synchronous DRAM (SDRAM), Double data rate SDRAM (DDRSDRAM),

Enhanced SDRAM (ESDRAM), Synchlink DRAM (SLDRAM), Rambus Direct RAM (RDRAM), Direct Rambus Dynamic RAM (DRDRAM), and Rambus Dynamic RAM (RDRAM), etc.

The respective technical features of the above embodiments can be merged arbitrarily, and for the conciseness of the description, not all possible combinations of each technical feature of the above embodiments have been described; however, as long as the combinations of these technical features are not contradictory, they should be considered to be within the scope of disclosure of the present description.

The above embodiments merely express several embodiments of the present application, and the description thereof is more relatively specific and detailed, but should not be construed as limiting the scope of the present invention. It should be noted that, for a person of ordinary skill in the art, several modifications and improvements can also be made without departing from the concept of the present application, which all fall within the scope of protection of the present application. Therefore, the scope of protection of the present application shall be subject to the appended claims.

CLAIMS

What is claimed is:

1. A method for processing a transaction request, the method being applied to any one of transaction nodes in a transaction system, the transaction system comprising a plurality of the transaction nodes; the method comprising:

upon being elected as a leader node in the plurality of transaction nodes, receiving a first transaction request from a client, obtaining latest market data from a market data server based on the first transaction request, processing the first transaction request to obtain a transaction processing result based on the market data;

replicating the transaction processing result to other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

2. The method according to claim 1, wherein each of the transaction nodes in the transaction system is configured with a service layer and a data replication layer;

wherein the receiving the first transaction request from the client, obtaining latest market data from the market data server based on the first transaction request, processing the first transaction request to obtain the transaction processing result based on the market data comprises:

receiving the first transaction request from the client via the service layer of current node, obtaining the latest market data from the market data server based on the first transaction request, and processing the first transaction request to obtain the transaction processing result based on the market data;

wherein the replicating the transaction processing result to the other

transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines comprises:

replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

3. The method according to claim 2, wherein the receiving the first transaction request from the client via the service layer of the current node is followed by:

obtaining a first node role and a first term from the data replication layer of the current node via the service layer of the current node;

obtaining a first historical term last obtained from the data replication layer of the current node;

detecting whether the first node role is a leader node and whether the first historical term is equal to the first term;

wherein the obtaining the latest market data from the market data server based on the first transaction request comprises:

upon detecting the first node role as a leader and the first historical term equal to the first term, obtaining the latest market data from the market data server based on the first transaction request via the service layer of the current node.

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

upon detecting the first node role not as a leader and the first historical term not equal to the first term, rejecting the first transaction request from the client via the service layer of the current node.

5. The method according to claim 3, wherein the transaction processing

result carries the first term;

the replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node is preceded by:

extracting the first term carried in the transaction processing result via the data replication layer of the current node;

obtaining a second node role and a second term;

detecting whether the second node role is the leader node and whether the first term carried in the transaction processing result is equal to the second term;

wherein the replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of current node comprises:

upon detecting the second node role as a leader and the first term equal to the second term, replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node.

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

upon detecting the second node role not as a leader and the first term not equal to the second term, discarding the transaction processing result from the service layer of the current node, and notifying the client that processing of the first transaction request has failed.

7. The method according to claim 4 or 6, further comprising:

obtaining node information of current leader node via the data replication layer of the current node;

sending the node information of the current leader node to the client via the service layer of the current node, causing the client to send a transaction

request to the current leader node based on the node information of the current leader node.

8. The method according to claim 1, wherein the replicating the transaction processing result to the other transaction nodes in the transaction system comprises:

appending the transaction processing result as a new log entry to local raft log;

replicating the transaction processing result to the other transaction nodes in the transaction system based on log replication mechanism of raft algorithm.

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

obtaining latest snapshot locally after a reboot, or from a remote storage system after a boot from a new machine;

using the transaction processing result corresponding to the latest snapshot as a starting position and current latest generated transaction processing result as an ending position, and applying all transaction processing results from the starting position to the ending position in sequence;

joining a raft cluster consisting of the plurality of the transaction nodes;

upon receiving the transaction processing result sent by the current leader node, applying the received transaction processing result.

10. The method according to claim 9, wherein the receiving the first transaction request from the client is preceded by:

obtaining a plurality of unprocessed transaction records from a cross system, the unprocessed transaction records referring to transaction records that have been closed and not yet processed;

processing each unprocessed transaction record, updating account asset information of a user corresponding to each unprocessed transaction record based on processing result of each unprocessed transaction record, recovering a

state of the replicated state machine of the current node to a state at the time of a first ex-leader node stepped down.

11. The method according to claim 2, further comprising: when the current node is a non-leader node, in response to receiving a second transaction request from the client via the service layer of the current node, obtaining a third node role and a third term from the data replication layer of the current node via the service layer of the current node;

obtaining a second historical term last obtained from the data replication layer of the current node;

detecting whether the third node role is a leader node and whether the second historical term is equal to the third term;

upon detecting the third node role as a leader node and the second historical term equal to the third term, recovering a state of the replicated state machine of the current node to a state at the time of a second ex-leader node stepped down and processing the second transaction request;

upon detecting the third node role not as a leader and the second historical term not equal to the third term, rejecting the second transaction request.

12. The method according to claim 1, further comprising: initiating a connection to a cross system after the current node becomes a leader node to cause the cross system to send latest transaction records to the current node via the connection; and disconnecting the connection upon detecting the current node not as the leader node.

13. A device for processing a transaction request, the device being applied to any one of transaction nodes in a transaction system, the transaction system comprising a plurality of the transaction nodes; the device comprising:

a request processing module configured for, upon being elected as a leader node in the plurality of transaction nodes, receiving a first transaction

request from a client, obtaining latest market data from a market data server based on the first transaction request, processing the first transaction request to obtain a transaction processing result based on the market data;

a processing result replication module configured for replicating the transaction processing result to other transaction nodes in the transaction system, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

14. The device according to claim 13, wherein the request processing module is configured with a service layer and the processing result replication module is configured with a data replication layer;

the request processing module is specifically configured for receiving the first transaction request from the client via the service layer of current node, obtaining the latest market data from the market data server based on the first transaction request, and processing the first transaction request to obtain the transaction processing result based on the market data;

the processing result replication module is specifically configured for replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, causing the other transaction nodes to apply the transaction processing result to their own replicated state machines.

15. The device according to claim 14, wherein the request processing module, after being configured for receiving the first transaction request from the client via the service layer of the current node, is further configured for obtaining a first node role and a first term from the data replication layer of the current node via the service layer of the current node; obtaining a first historical term last obtained from the data replication layer of the current node; detecting whether the first node role is a leader node and whether the first

historical term is equal to the first term;

the request processing module, upon being configured for obtaining the latest market data from the market data server based on the first transaction request, is specifically configured for, upon detecting the first node role as a leader and the first historical term equal to the first term, obtaining the latest market data from the market data server based on the first transaction request via the service layer of the current node.

16. The device according to claim 15, wherein the request processing module is further configured for, upon detecting the first node role not as a leader and the first historical term not equal to the first term, rejecting the first transaction request from the client via the service layer of the current node.

17. The device according to claim 15, wherein the transaction processing result carries the first term;

the processing result replication module, before being configured for replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, is further configured for extracting the first term carried in the transaction processing result via the data replication layer of the current node; obtaining a second node role and a second term; detecting whether the second node role is the leader node and whether the first term carried in the transaction processing result is equal to the second term;

the processing result replication module, upon being configured for replicating the transaction processing result to the other transaction nodes in the transaction system via the data replication layer of the current node, is specifically configured for, upon detecting the second node role as a leader and the first term equal to the second term, replicating the transaction processing result to the other transaction nodes in the transaction system via the data

replication layer of the current node.

18. The device according to claim 17, wherein the processing result replication module is further configured for, upon detecting the second node role not as a leader and the first term not equal to the second term, discarding the transaction processing result from the service layer of the current node, and notifying the client that processing of the first transaction request has failed.

19. The device according to claim 16 or 18, wherein the processing result replication module is further configured for obtaining node information of current leader node via the data replication layer of the current node;

the request processing module is further configured for sending the node information of the current leader node to the client via the service layer of the current node, causing the client to send a transaction request to the current leader node based on the node information of the current leader node.

20. The device according to claim 13, wherein the processing result replication module, upon being configured for replicating the transaction processing result to the other transaction nodes in the transaction system, is specifically configured for appending the transaction processing result as a new log entry to local raft logs; replicating the transaction processing result to the other transaction nodes in the transaction system based on log replication mechanism of raft algorithm.

21. The device according to claim 13, further comprising the following modules:

a snapshot acquisition module configured for obtaining latest snapshot locally after a reboot, or from a remote storage system after a boot from a new machine;

a first result application module configured for using the transaction processing result corresponding to the latest snapshot as a starting position and

current latest generated transaction processing result as an ending position, and applying all transaction processing results from the starting position to the ending position in sequence;

a cluster joining module configured for joining a raft cluster consisting of the plurality of the transaction nodes;

a second result application module configured for applying transaction processing result received from the current leader node.

22. The device according to claim 21, further comprising the following modules:

a transaction record acquisition module configured for obtaining a plurality of unprocessed transaction records from a cross system, the unprocessed transaction records referring to transaction records that have been closed and not yet processed;

a transaction record processing module configured for processing each unprocessed transaction record, updating account asset information of a user corresponding to each unprocessed transaction record based on processing result of each unprocessed transaction record, recovering a state of the replicated state machine of the current node to a state at the time of a first ex-leader node stepped down.

23. The device according to claim 14, wherein the request processing module is further configured for performing the following steps:

when the current node is a non-leader node, in response to receiving a second transaction request from the client via the service layer of the current node, obtaining a third node role and a third term from the data replication layer of the current node via the service layer of the current node;

obtaining a second historical term last obtained from the data replication layer of the current node;

detecting whether the third node role is a leader node and whether the second historical term is equal to the third term;

upon detecting the third node role as a leader node and the second historical term equal to the third term, recovering a state of the replicated state machine of the current node to a state at the time of a second ex-leader node stepped down and processing the second transaction request;

upon detecting the third node role not as a leader and the second historical term not equal to the third term, rejecting the second transaction request.

24. The device according to claim 13, further comprising a connection management module;

wherein the connection management module is configured for initiating a connection to a cross system after the current node becomes a leader node to cause the cross system to send latest transaction records to the current node via the connection; and disconnecting the connection upon detecting the current node not as the leader node.

25. A transaction node comprising a memory, a processor and a computer program stored on the memory and executable by the processor, wherein the processor performs the steps of the method according to any one of claims 1 to 12 when executing the computer program.

26. A computer readable storage medium on which a computer program is stored, wherein the computer program when executed by a processor performs the steps of the method according to any one of claims 1 to 12.