FRAGMENTATION PROCESSING METHOD FOR USE WITH MULTI-BLOCKCHAINS, DEVICE, SERVER AND STORAGE MEDIUM

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims priority to Chinese Patent Application No. 2017101392776, entitled "Fragmentation Processing Method for Use with Multi-Blockchains and Device" filed March 9, 2017, the content of which is expressly incorporated by reference herein in its entirety.

TECHNICAL FIELD

[0002] The present disclosure relates to a field of Internet technologies, particularly to a multi-blockchains fragmentation processing method, a multi-blockchains fragmentation processing device, a server and a storage medium.

BACKGROUND

[0003] An application-level code of a blockchain refer to a code in an application level called in an application field of the blockchain. The application field of the blockchain includes smart contracts, securities transactions, and e-commerce, etc. The application-level code is stored in the blockchain after initialization, and a transaction can be performed on the blockchain through the application-level code, and a transaction result is saved in the blockchain. In the conventional blockchain platform, usually only one blockchain is provided to cover a plurality of service types, and electronic transactions of a plurality of service types can be performed therethrough. If a large number of transaction requests are initiated to the blockchain at the same time, it will undoubtedly have a great impact on the performance of the blockchain and even lead to shutdown of the blockchain platform.

SUMMARY

[0004] According to various embodiments of the present disclosure, a multi-blockchains fragmentation processing method, a multi-blockchains fragmentation processing device, a server and a storage medium are provided.

[0005] A multi-blockchains fragmentation processing method includes:

[0006] receiving a transaction request sent by an application client, wherein the transaction request carries an application client parameter, and the application client parameter includes a service type;

[0007] acquiring blockchain identifiers of a plurality of blockchains, wherein the blockchain has a corresponding service type and has an application-level code deployed thereon;

[0008] parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request; and

[0009] assigning the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

[0010] A multi-blockchains fragmentation processing device includes:

[0011] an application-level code fragmentation server configured to receive a transaction request from an application client, wherein the transaction request carries an application client parameter, and the application client parameter includes a service type; to acquire blockchain identifiers of a plurality of blockchains, wherein the blockchain has a corresponding service type and has an application-level code deployed thereon; and to parse the application client parameter and select the blockchain identifier corresponding to the service type carried by the transaction request; and

[0012] an application-level code fragmentation client configured to assign the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

[0013] A server includes a memory and a processor, the memory having computer-readable instructions stored therein which, when executed by the processor, cause the processor to perform the steps of:

[0014] receiving a transaction request sent by an application client, wherein the transaction request carries an application client parameter, and the application client parameter includes a service type;

[0015] acquiring blockchain identifiers of a plurality of blockchains, wherein the blockchain has a corresponding service type and has an application-level code deployed thereon;

[0016] parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request; and

[0017] assigning the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

[0018] One or more non-transitory computer-readable storage medium has computer-readable instructions stored thereon which, when executed by one or more processors, cause the one or more processors to perform the steps of:

[0019] receiving a transaction request sent by an application client, wherein the transaction request carries an application client parameter, and the application client parameter includes a service type;

[0020] acquiring blockchain identifiers of a plurality of blockchains, wherein the blockchain has a corresponding service type and has an application-level code deployed thereon;

[0021] parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request; and

[0022] assigning the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

[0023] The details of at least one embodiment of the present disclosure will be presented with reference to the following drawings and description. Other

characteristic and advantages of the present disclosure will be more apparent from the specification, drawings and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] To illustrate the technical schemes according to the embodiments of the present disclosure or in the prior art more clearly, the accompanying drawings for describing the embodiments or the prior art are introduced briefly in the following. Apparently, the accompanying drawings in the following description are only some embodiments of the present disclosure, and those skilled in the art can derive other drawings from the accompanying drawings without creative efforts.

[0025] FIG. 1 is a schematic diagram of an application environment of a multi-blockchains fragmentation processing method according to an embodiment.

[0026] FIG. 2 is a flowchart of a multi-blockchains fragmentation processing method according to an embodiment.

[0027] FIG. 3 is a block diagram of a multi-blockchains fragmentation processing system according to an embodiment.

[0028] FIG. 4 is a block diagram of a server according to an embodiment.

[0029] FIG. 5 is a block diagram of a multi-blockchains fragmentation processing device according to an embodiment.

[0030] FIG. 6 is a block diagram of a multi-blockchains fragmentation processing device according to another embodiment.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0031] The present disclosure will be further described below with reference to the accompanying drawings for more clearly and fully discussing the technical solutions of the present disclosure. It should be appreciated that specific embodiments described herein are merely illustrative of the present disclosure and are not intended to limit the present disclosure.

[0032] The multi-blockchains fragmentation processing method provided in the embodiment of the present disclosure can be applied to the application environment as shown in FIG. 1. A client terminal 102 is connected to a server 104 through a network. The client terminal 102 has an application client installed thereon . Each application client is configured to conduct a transaction of one business type. The server 104 has an application-level code fragmentation server and a plurality of application-level code fragmentation clients running thereon. The plurality of application-level code fragmentation clients correspond to a plurality of blockchains in a one-to-one manner. The application-level code fragmentation server receives a transaction request sent by the application client. The transaction request carries an application client parameter. The application-level code fragmentation server parses the application client parameter to obtain a service type, a service identifier and a transaction data involved therein. The application-level code fragmentation server acquires blockchain identifiers of the plurality of blockchains, and selects the blockchain identifier corresponding to the service type carried by the transaction request among the plurality of blockchain identifiers. The blockchains correspond to the application-level code fragmentation clients in a one-to-one manner. The application-level code fragmentation server selects the corresponding application-level code fragmentation client according to the selected blockchain identifier, and forwards the transaction request to the selected application-level code fragmentation client. The application-level code fragmentation client calls an external interface of a corresponding blockchain according to the transaction request, whereby an application-level code deployed on the blockchain is called to perform the transaction according to the application client parameter. Since the blockchain has the corresponding service type, the transaction request is assigned to the blockchain corresponding to the service type. When facing a large number of transaction requests, the transactions of different service types can be performed in different blockchains, the performance of the blockchains is improved effectively.

[0033] In one embodiment, as shown in FIG. 2, a multi-blockchains fragmentation processing method is provided. It should be appreciated that even though steps in the

flowchart of FIG. 2 are shown sequentially as indicated by arrows, these steps are not necessarily performed in the sequence indicated by the arrows. Unless expressly stated herein, these steps is not strictly limited to a certain sequence when performed, and may be performed in other sequences. Moreover, at least a part of the steps in FIG. 2 may include a plurality of sub-steps or stages that are not necessarily performed at the same time, but may be performed at different times, and that are not necessarily performed in sequence, but may be performed in turn or alternately with at least a part of other steps, or of sub-steps or stages of other steps. The method applied to the server in FIG. 1 will be described by way of example, specifically including:

[0034] step 202, a transaction request sent by an application client is received, the transaction request carrying an application client parameter, and the application client parameter including a service type.

[0035] step 204, blockchain identifiers of a plurality of blockchains are acquired, the blockchain having a corresponding service type, and having an application-level code deployed thereon.

[0036] step 206, the application client parameter is parsed, and the blockchain identifier corresponding to the service type carried by the transaction request is selected.

[0037] step 208, the transaction request is assigned to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

[0038] The client terminal has an application client installed thereon. Each application client is configured to conduct a transaction of one business type. The client terminal is able to conduct transactions of different service types through different application clients. One client terminal may has one application client installed thereon, or has a plurality of application clients installed thereon.

[0039] In the blockchain platform, a plurality of blockchains are generated according to the service types. Each blockchain has a corresponding service type, and the same service type may also correspond to a plurality of blockchains. Each blockchain may have a corresponding blockchain identifier. For example, the

blockchain identifier may be a serial number of the blockchain. Each blockchain may has an application-level code deployed thereon according to a preset rule, and the blockchains provide various external interfaces for the application-level codes. The external interfaces include an initialization interface, a transaction call interface, a transaction result inquiry interface, and a blockchain state inquiry interface, etc. The user of the blockchain is able to call the application-level code deployed on the specified blockchain when calling these external interfaces through the application client and the server, so that each blockchain can perform the transaction corresponding to its service type. This may also be referred to as a fragmentation of the application level code.

[0040] The server has an application-level code fragmentation server and a plurality of application-level code fragmentation clients running thereon. The application-level code fragmentation clients correspond to the blockchains in a one-to-one manner. When the application-level code fragmentation server and the plurality of application-level code fragmentation clients are started, the application-level code fragmentation server establishes connections with the plurality of application clients, and the application-level code fragmentation clients establish connections with the application-level codes deployed on the one-to-one corresponding blockchains.

[0041] The application-level code fragmentation server running on the server receives a transaction request sent by the application client. The transaction request carries an application client parameter. The application-level code fragmentation server parses the application client parameter to obtain a service type, a service identifier and a transaction data involved therein. The transaction data includes a transaction object, a transaction amount and a transaction date, etc. The application-level code fragmentation server acquires blockchain identifiers of the plurality of blockchains, and selects the blockchain identifier corresponding to the service type carried by the transaction request among the plurality of blockchain identifiers. Specifically, if there is only one blockchain identifier corresponding to the service type carried in the transaction request, the blockchain identifier is recorded as

the selected blockchain identifier. If there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request, the application-level code fragmentation server performs load balancing processing according to the plurality of blockchain identifiers, and selects one of the plurality of blockchain identifiers of the same service type, and records the selected blockchain identifier.

[0042] Since the plurality of blockchains correspond to the plurality of application-level code fragmentation clients in a one-to-one manner, the application-level code fragmentation server selects the corresponding application-level code fragmentation client according to the selected blockchain identifier, and forwards the transaction request to the selected application-level code fragmentation client. Thus, an appropriate application-level code client can be selected to process the transaction request on the basis of ensuring the load balance and data consistency of the blockchains. The application-level code fragmentation client calls an external interface of a corresponding blockchain according to the transaction request, whereby an application-level code deployed on the blockchain is called to perform the transaction according to the application client parameter. Data migration between different blockchains and problems such as additional load due to the data migration can be effectively avoided.

[0043] In the present embodiment, after the transaction request of the application client is received, the application client parameter carried in the transaction request is parsed, and the blockchain identifier corresponding to the service type carried in the transaction request is selected among a plurality of blockchain identifiers, and thus the transaction request is assigned to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes the transaction according to the application client parameter. Since the blockchain has a corresponding service type, the transaction request is assigned to the blockchain corresponding to the service type. When facing a large number of transaction requests, the transactions of different service types can be conducted in different blockchains, effectively improving the performance of the blockchains.

[0044] In one embodiment, the parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request includes: hash calculations are performed respectively according to the service type, the service identifier and each blockchain identifier to obtain a corresponding set of hash results if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request,; the blockchain identifier corresponding to the transaction request is selected according to the set of hash results.

[0045] In the present embodiment, the server establishes connections with a plurality of blockchains, and establishes connections with a plurality of application clients. When a number of the blockchains remains unchanged, regardless of the change in a number of the application clients, the server needs to ensure no data migration between the blockchains, and ensure the load balance and data consistency of the blockchains. That is, the server needs to ensure that the plurality of blockchains can be evenly called, and that the transaction requests sent by the same application client are assigned to the same blockchain.

[0046] Specifically, the application-level code fragmentation server running on the server parses the application client parameter to obtain the corresponding service type and service identifier. If there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request, the application-level code fragmentation server acquires the plurality of blockchain identifiers corresponding to the same service type. The application-level code fragmentation server performs hash calculations respectively using the service type, the service identifier and each blockchain identifier to obtain a corresponding set of hash results. For example, a hash algorithm with the highest random weight is used to perform a consistent hash calculation on the service type, the service identifier, and the blockchain identifier to obtain a corresponding hash result.

[0047] If the application-level code fragmentation server running on the server receives a plurality of transaction requests of the same service type, the application-level code fragmentation server parses the application client parameter

carried in each transaction request, and obtains the service type and the service identifier input by each application client. The application-level code fragmentation server performs hash calculations respectively on the service type and the service identifier input by each application client, and each blockchain identifier to obtain a corresponding set of hash results. The server selects a corresponding blockchain identifier for each transaction request in the set of hash results by way of the load balancing and the consistency of the hash results.

[0048] The application-level code fragmentation server performs the load balancing on the hash results, and selects the blockchain identifiers corresponding to the transaction requests among the plurality of blockchain identifiers corresponding to the same service type. Thus, the plurality of blockchains can be evenly used, and the load balance of the blockchains can be effectively ensured. Moreover, the transaction requests of the same application client can be assigned to the same blockchain, thereby effectively ensuring the data consistency of the blockchain. By selecting the appropriate blockchain for the transaction request sent by the application client, it is possible to ensure that no data migration between the blockchains occurs regardless of the change in a data amount of the application clients when the number of the blockchains remains unchanged, that is, there is no case where the transaction data of one blockchain is migrated to another blockchain.

[0049] In one of the embodiments, the parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request includes: the blockchains of the same service type are served as a physical blockchain if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request, and the physical blockchain is divided into a plurality of virtual blockchains; virtual blockchain identifiers corresponding to the plurality of virtual blockchains are acquired; hash calculations are performed respectively according to the service type, the service identifier and each virtual blockchain identifier to obtain a corresponding set of hash results; the virtual blockchain identifier corresponding to the transaction request is selected according to the set of hash results.

[0050] When the application-level code fragmentation server running on the server receives a plurality of transaction requests of the same service type, and a number of the transaction requests of the same service type is greater than a number of the blockchains corresponding to the same service type, the application-level code fragmentation server running on the server may serve the blockchains corresponding to the same service type as a physical blockchain. The application-level code fragmentation server divides each physical blockchain into a plurality of virtual blockchains, and generates a corresponding virtual blockchain identifier for each virtual blockchain. The application-level code fragmentation server performs hash calculations respectively according to the service type, the service identifier and each virtual blockchain identifier to obtain a corresponding set of hash results. The server selects a corresponding blockchain identifier for each transaction request in the set of hash results by way of the load balancing and the consistency of the hash results.

[0051] The application-level code fragmentation server performs load balancing on the hash results, and selects the virtual blockchain identifiers corresponding to the transaction requests among the plurality of virtual blockchain identifiers corresponding to the same service type. The virtual blockchain identifiers are recorded as the selected virtual blockchain identifiers. The application-level code fragmentation server forwards the plurality of transaction requests to the application-level code fragmentation clients corresponding to the selected virtual blockchain identifiers respectively. By the application-level code fragmentation clients assigning the transaction requests to the corresponding virtual blockchain identifiers, the physical blockchain mapped by the virtual blockchains is accessible according to the virtual blockchain identifiers. Thus, when the number of the transaction requests of the same service type is larger than the number of the blockchains corresponding to the same service type, the plurality of blockchains can be evenly used, and the load balance of the blockchains can be effectively ensured. Moreover, the transaction requests of the same application client can be assigned to the same blockchain, effectively ensuring the data consistency of the blockchain. By selecting the appropriate blockchain for the transaction request sent by the application

client, it is possible to ensure that no data migration between the blockchains occurs regardless of the change in a data amount of the application clients when a number of the blockchains remains unchanged.

[0052] In one embodiment, the parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request includes: a mapping table is acquired when a number of the blockchains changes, the mapping table including a mapping relationship between the application client parameter and the blockchain identifier, and the application client parameter including a service type and a service identifier; the corresponding blockchain identifier is retrieved in the mapping table according to the mapping relationship and the service type and the service identifier carried in the transaction request, and the retrieved blockchain identifier is recorded as the selected blockchain identifier.

[0053] In the present embodiment, the number of the blockchains connecting to the server may change. For example, some of the blockchains fail to connect to the server, resulting in that the blockchains are not available, and that the number of the blockchains is reduced. Alternatively, more blockchains are provided to connect to the server, resulting in that the number of the blockchains is increased. The mapping table between the application client parameter and the blockchain identifier may be established in advance, the application client parameter including a service type and a service identifier.

[0054] When the application-level code fragmentation server running on the server receives a transaction request sent by the application client, the corresponding blockchain identifier is retrieved in the mapping table according to the mapping relationship and the service type and service identifier carried in the transaction request, and the retrieved blockchain identifier is recorded as the selected blockchain identifier. Then the transaction request is assigned to the selected blockchain identifier. Thus, the load balance and data consistency of blockchains are realized. By selecting the appropriate blockchain for the transaction request sent by the application client, it is possible to ensure that no data migration between the blockchains occurs when a

data amount of the application clients changes.

[0055] Further, the mapping table is stored in a disk memory of the server and may be cached in a RAM to facilitate fast reading during selecting of the appropriate blockchain identifier for the transaction request. If a new application client parameter and/or a new blockchain identifier are added, the application-level code fragmentation server running on the server may update the mapping table according to the newly added application client parameter and/or the newly added blockchain identifier, and the updated mapping table may be cached in memory.

[0056] In one embodiment, the method further includes: an event message flow acquisition request sent by the application client is received; the event message flow acquisition request is sent to the blockchain to which the transaction request is assigned; a corresponding event message flow is acquired when the blockchain performs the transaction through the application-level code, and the event message flow is returned to the application client.

[0057] In the present embodiment, in addition to the application-level code fragmentation server and the application-level code fragmentation clients, the server further includes an event message server and an event message client running thereon. A plurality of event message clients may be provided, which correspond to the blockchains in a one-to-one manner. The application-level code fragmentation server, the application-level code fragmentation clients, the event message server and the event message clients running on the server may be collectively referred to as a multi-blockchains fragmentation processing system. When the multi-blockchains fragmentation processing system is started, the application-level code fragmentation server establishes connections with the plurality of application clients, the event message server establishes connections with the plurality of application clients, the application-level code fragmentation clients establish connections with the application-level codes deployed on the one-to-one corresponding blockchains, and the event message clients establish connections with the application-level codes deployed on the one-to-one corresponding blockchains, as shown in FIG. 3. Each blockchain may correspond to an application-level code fragmentation client and an

event message client in a one-to-one manner, respectively. The multi-blockchains fragmentation processing system will establish the aforementioned connections immediately after being started, so as to avoid extra burden for network connection caused by establishing the connections only when the application-level code and the event processing are executed.

[0058] The multi-blockchains fragmentation processing system monitors the connections between the blockchains and the application-level code fragmentation clients and the connections between the blockchains and the event message clients. When a certain connection between the blockchain and the application-level code fragmentation client or the event message client is interrupted, the disconnected blockchain is unavailable and the transaction and the event message flow cannot be processed. After the network connection of the blockchain is recovered, the blockchain can be available again.

[0059] The event includes an event triggered when the application client communicates with the application-level code fragmentation server, an event triggered when the application client communicates with the event message server, and an event triggered when the application-level code deployed on the blockchain performs a transaction. For example, a registration, a cancellation and a rejection, etc., as well as a transaction state and a blockchain state, etc. In the conventional manner, the application-level code deployed on the blockchain may return information of a communication event such as a registration, a cancellation, and a rejection, etc. to the server in a synchronous or asynchronous manner. However, a message of the event of interest to the user, such as the transaction state, can be obtained only if the user actively inquires through the application client.

[0060] In the present embodiment, the plurality of application clients communicate with the plurality of blockchains in the manner of the event message flow. After the event message server establishes a connection with the application client, the application client may register the event of interest, such as the transaction state and the blockchain state, with the event message server. When the application client sends a transaction request to the application-level code fragmentation server, the

application client may also send an event message flow acquisition request to the event message server, and the application-level code fragmentation server selects the corresponding blockchain according to the service type and the service identifier carried in the transaction request. The application-level code fragmentation client assigns the transaction request of the application client to the selected blockchain. The event message server forwards the event message flow acquisition request sent by the application client to the event message client, and the event message client sends the event message flow acquisition request to the blockchain which processes the transaction request. That is, the transaction request and the event message flow acquisition request are assigned to the same blockchain.

[0061] When the blockchain performs the transaction through the application-level code, the event client calls the external interface of the application-level code in a synchronous or asynchronous manner to acquire the event message flow. The event client returns the acquired event message flow to the event server which sends the event message flow to the corresponding application client. Thus, the user can directly learn the message of the event of interest without needs to query the event of interest through the application client, which provides convenience for the user.

[0062] The plurality of application clients may communicate with the plurality of blockchains in the manner of a bi-directional event message flow. The event server may concurrently read the event message flow of the plurality of application clients, and send the read event message flow to the corresponding blockchain through the event client. The event client may also concurrently read the event message flow of the blockchain, and concurrently send the event message flow to the application client through the event server. Concurrency includes concurrent processing of processes, concurrent processing of threads and concurrent processing of coprocesses, etc. By concurrently reading and writing the event message flow between the application client and the blockchain, the communication efficiency between the application client and the blockchain can be effectively improved.

[0063] In one embodiment, the server may have a plurality of multi-blockchains fragmentation processing system running thereon which form a multi-blockchains

fragmentation processing cluster. The application client and the blockchain may establish connections with the multi-blockchains fragmentation processing system in the multi-blockchains fragmentation processing cluster. When the connected multi-blockchains fragmentation processing system fails, other multi-blockchains fragmentation processing systems in the multi-blockchains fragmentation processing cluster may be switched to continue processing of the transaction and the event message flow. Therefore, a single point failure of the multi-blockchains fragmentation processing system is eliminated through the cluster mode, and the performance for multi-blockchains fragmentation processing is effectively improved.

[0064] In one embodiment, as shown in FIG. 4, a server 400 is provided that includes a processor 401, a Random Access Memory (RAM) 402, a non-transitory storage medium 403, and a network interface 404 which are connected through a system bus. The processor 401 is configured to provide capabilities of computing and controlling. The non-transitory storage medium 403 stores therein an operating system 4031 and computer readable instructions 4032. The RAM 402 provides an environment for execution of the computer readable instructions 4032 in the non-transitory storage medium. The processor 401 may perform the aforementioned multi-blockchains fragmentation processing method when the computer readable instructions 4032 are executed by the processor 401. The computer readable instructions 4032 are configured to effectively reduce time-consuming for masking process for the client data and improve efficiency of the masking process. The network interface 404 is configured to communicate with an external client terminal through a network connection, e.g. to receive a transaction request uploaded by the client terminal, to send an event message flow to the client terminal, and the like. The server 400 may be a separate server or a server cluster including a plurality of servers. It should be understood by those skilled in the art that FIG. 4 is exemplary to show the parts of the structure relevant to the embodiments of the present disclosure and does not limit the server where the present disclosure is applied; the particular server may include more or fewer components, be combined with other components, or have a different component arrangement.

[0065] In one embodiment, a multi-blockchains fragmentation processing device 500 is provided, including an application-level code fragmentation server 502 and an application-level code fragmentation client.

[0066] The application-level code fragmentation server 502 is configured to receive a transaction request from an application client, the transaction request carrying an application client parameter, and the application client parameter including a service type; to acquire blockchain identifiers of a plurality of blockchains, the blockchain having a corresponding service type and having an application-level code deployed thereon; and to parse the application client parameter and select the blockchain identifier corresponding to the service type carried by the transaction request.

[0067] The application-level code fragmentation client 504 is configured to assign the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

[0068] In one embodiment, the application client parameter further includes a service identifier, and the application-level code fragmentation server 502 is further configured to perform hash calculations respectively according to the service type, the service identifier and each blockchain identifier to obtain a corresponding set of hash results if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request; and to select the blockchain identifier corresponding to the transaction request according to the set of hash results.

[0069] In one embodiment, the application client parameter further includes a service identifier, and the application-level code fragmentation server 502 is further configured to serve the blockchains corresponding to the same service type as a physical blockchain if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request, and divide the physical blockchain into a plurality of virtual blockchains; to acquire virtual blockchain identifiers corresponding to the plurality of virtual blockchains; to perform hash calculations respectively according to the service type, the service identifier and each virtual

blockchain identifier to obtain a corresponding set of hash results; and to select the virtual blockchain identifier corresponding to the transaction request according to the set of hash results.

[0070] In one embodiment, the application-level code fragmentation server 502 is further configured to acquire a mapping table when a number of the blockchains changes, the mapping table including a mapping relationship between the application client parameter and the blockchain identifier, and the application client parameter including a service type and a service identifier; to retrieve the corresponding blockchain identifier in the mapping table according to the mapping relationship and the service type and service identifier carried in the transaction request, and record the retrieved blockchain identifier as the selected blockchain identifier.

[0071] In one embodiment, as shown in FIG. 6, the device further includes an event message server 506 and an event message client 508.

[0072] The event message server 506 is configured to receive an event message flow acquisition request sent by the application client; and to send the event message flow acquisition request to the blockchain to which the transaction request is assigned.

[0073] The event message client 508 is configured to acquire a corresponding event message flow when the blockchain performs the transaction through the application-level code and return the event message flow to the application client.

[0074] The respective modules in the aforementioned multi-blockchains fragmentation processing device may be implemented in whole or in part by software, hardware, and combinations thereof. For example, in the hardware, the aforementioned application-level code fragmentation server 502 is configured to receive a transaction request sent by the application client via a network interface. The network interface may be an Ethernet card or a wireless network card. The above-mentioned modules may be embedded in the form of hardware in or independent from the processor of the server, or may be stored in the memory of the server in the form of software so as to be called by the processor to perform the operations corresponding to the above respective modules. The processor may be a

central processing unit (CPU), a microprocessor, or the like.

[0075] Those skilled in the art will appreciate that all or part of the flows of the method of implementing the aforementioned embodiments may be accomplished by instructing related hardware by a computer program that may be stored in a nonvolatile computer-readable storage medium, the program, when executed, may include the flows of embodiments of the methods as described. The storage medium may be a disk, an optical disk, a read-only memory (ROM), or the like.

[0076] The foregoing respective technical features involved in the respective embodiments can be combined arbitrarily, for brevity, not all possible combinations of the respective technical features in the foregoing embodiments are described, however, to the extent they have no collision with each other, the combination of the respective technical features shall be considered to be within the scope of the description.

[0077] The foregoing implementations are merely specific embodiments of the present disclosure, and are not intended to limit the protection scope of the present disclosure. It should be noted that any variation or replacement readily figured out by persons skilled in the art within the technical scope disclosed in the present disclosure shall all fall into the protection scope of the present disclosure. Therefore, the protection scope of the present disclosure shall be subject to the protection scope of the claims.

What is claimed is:

1. A multi-blockchains fragmentation processing method, comprising:

receiving a transaction request sent by an application client, wherein the transaction request carries an application client parameter, and the application client parameter comprises a service type;

acquiring blockchain identifiers of a plurality of blockchains, wherein the blockchain has a corresponding service type and has an application-level code deployed thereon;

parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request; and

assigning the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

2. The method according to claim 1, wherein the application client parameter further comprises a service identifier, and the parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request comprises:

performing hash calculations respectively according to the service type, the service identifier and each blockchain identifier to obtain a corresponding set of hash results if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request; and

selecting the blockchain identifier corresponding to the transaction request according to the set of hash results.

3. The method according to claim 1, wherein the application client parameter further comprises a service identifier, and the parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request comprises:

serving the blockchains of the same service type as a physical blockchain if there are a plurality of blockchain identifiers corresponding to the service type carried in the

transaction request, and dividing the physical blockchain into a plurality of virtual blockchains;
acquiring virtual blockchain identifiers corresponding to the plurality of virtual blockchains;
performing hash calculations respectively according to the service type, the service identifier and each virtual blockchain identifier to obtain a corresponding set of hash results; and
selecting the virtual blockchain identifier corresponding to the transaction request according to the set of hash results.

4. The method according to claim 1, wherein the parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request comprises:
acquiring a mapping table when a number of the blockchains changes, wherein the mapping table comprises a mapping relationship between the application client parameter and the blockchain identifier, and the application client parameter comprises a service type and a service identifier; and
retrieving the corresponding blockchain identifier in the mapping table according to the mapping relationship and the service type and the service identifier carried in the transaction request, and recording the retrieved blockchain identifier as the selected blockchain identifier.

5. The method according to claim 1, further comprising:
receiving an event message flow acquisition request sent by the application client;
sending the event message flow acquisition request to the blockchain to which the transaction request is assigned; and
acquiring a corresponding event message flow when the blockchain performs the transaction through the application-level code, and returning the event message flow to the application client.

6. A multi-blockchains fragmentation processing device, comprising:
an application-level code fragmentation server configured to receive a transaction request from an application client, wherein the transaction request carries an application client parameter, and the application client parameter comprises a service type; to acquire blockchain identifiers of a plurality of blockchains, wherein the blockchain has a corresponding service type and has an application-level code deployed thereon; and to parse the application client parameter and select the blockchain identifier corresponding to the service type carried by the transaction request; and
an application-level code fragmentation client configured to assign the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

7. The device according to claim 6, wherein the application client parameter further comprises a service identifier, and the application-level code fragmentation server is further configured to perform hash calculations respectively according to the service type, the service identifier and each blockchain identifier to obtain a corresponding set of hash results if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request; and to select the blockchain identifier corresponding to the transaction request according to the set of hash results.

8. The device according to claim 6, wherein the application client parameter further comprises a service identifier, and the application-level code fragmentation server is further configured to serve the blockchains of the same service type as a physical blockchain if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request, and divide the physical blockchain into a plurality of virtual blockchains; to acquire virtual blockchain identifiers corresponding to the plurality of virtual blockchains; to perform hash calculations respectively according to the service type, the service identifier and each virtual

blockchain identifier to obtain a corresponding set of hash results; and to select the virtual blockchain identifier corresponding to the transaction request according to the set of hash results.

9. The device according to claim 6, wherein the application-level code fragmentation server is further configured to acquire a mapping table when a number of the blockchains changes, wherein the mapping table comprises a mapping relationship between the application client parameter and the blockchain identifier, and the application client parameter comprises a service type and a service identifier; and to retrieve the corresponding blockchain identifier in the mapping table according to the mapping relationship and the service type and service identifier carried in the transaction request, and record the retrieved blockchain identifier as the selected blockchain identifier.

10. The device according to claim 6, further comprising:
an event message server configured to receive an event message flow acquisition request sent by the application client; and to send the event message flow acquisition request to the blockchain to which the transaction request is assigned; and
an event message client configured to acquire a corresponding event message flow when the blockchain performs the transaction through the application-level code, and return the event message flow to the application client.

11. A server, comprising a memory and a processor, the memory having computer-readable instructions stored therein which, when executed by the processor, cause the processor to perform the steps of:
receiving a transaction request sent by an application client, wherein the transaction request carries an application client parameter, and the application client parameter comprises a service type;
acquiring blockchain identifiers of a plurality of blockchains, wherein the blockchain has a corresponding service type and has an application-level code deployed thereon;

parsing the application client parameter and selecting the blockchain identifier corresponding to the service type carried by the transaction request; and
assigning the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

12. The server according to claim 11, wherein the application client parameter further comprises a service identifier, and in the parsing the application client parameter, the processor further performs the steps of:
performing hash calculations respectively according to the service type, the service identifier and each blockchain identifier to obtain a corresponding set of hash results if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request; and
selecting the blockchain identifier corresponding to the transaction request according to the set of hash results.

13. The server according to claim 11, wherein the application client parameter further comprises a service identifier, and in the parsing the application client parameter, the processor further performs the steps of:
serving the blockchains of the same service type as a physical blockchain if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request, and dividing the physical blockchain into a plurality of virtual blockchains;
acquiring virtual blockchain identifiers corresponding to the plurality of virtual blockchains;
performing hash calculations respectively according to the service type, the service identifier and each virtual blockchain identifier to obtain a corresponding set of hash results; and
selecting the virtual blockchain identifier corresponding to the transaction request according to the set of hash results.

14. The server according to claim 11, wherein in the parsing the application client parameter, the processor further performs the steps of:
acquiring a mapping table when a number of the blockchains changes, wherein the mapping table comprises a mapping relationship between the application client parameter and the blockchain identifier, and the application client parameter comprises a service type and a service identifier; and
retrieving the corresponding blockchain identifier in the mapping table according to the mapping relationship and the service type and the service identifier carried in the transaction request, and recording the retrieved blockchain identifier as the selected blockchain identifier.

15. The server according to claim 11, when the computer-readable instructions are executed by the processor, the processor further performs the steps of:
receiving an event message flow acquisition request sent by the application client;
sending the event message flow acquisition request to the blockchain to which the transaction request is assigned;
acquiring a corresponding event message flow when the blockchain performs the transaction through the application-level code and returning the event message flow to the application client.

16. One or more non-transitory computer-readable storage medium having computer-readable instructions stored thereon which, when executed by one or more processors, cause the one or more processors to perform the steps of:
receiving a transaction request sent by an application client, wherein the transaction request carries an application client parameter, and the application client parameter comprises a service type;
acquiring blockchain identifiers of a plurality of blockchains, wherein the blockchain has a corresponding service type and has an application-level code deployed thereon;
parsing the application client parameter and selecting the blockchain identifier

corresponding to the service type carried by the transaction request; and
assigning the transaction request to the blockchain corresponding to the selected blockchain identifier, so that the application-level code deployed on the blockchain completes a transaction according to the application client parameter.

17. The non-transitory computer-readable storage medium according to claim 16, wherein the application client parameter further comprises a service identifier, and in the parsing the application client parameter, the one or more processors further perform the steps of:
performing hash calculations respectively according to the service type, the service identifier and each blockchain identifier to obtain a corresponding set of hash results if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request; and
selecting the blockchain identifier corresponding to the transaction request according to the set of hash results.

18. The non-transitory computer-readable storage medium according to claim 16, wherein the application client parameter further comprises a service identifier, and in the parsing the application client parameter, the one or more processors further perform the steps of:
serving the blockchains of the same service type as a physical blockchain if there are a plurality of blockchain identifiers corresponding to the service type carried in the transaction request, and dividing the physical blockchain into a plurality of virtual blockchains;
acquiring virtual blockchain identifiers corresponding to the plurality of virtual blockchains;
performing hash calculations respectively according to the service type, the service identifier and each virtual blockchain identifier to obtain a corresponding set of hash results; and
selecting the virtual blockchain identifier corresponding to the transaction request

according to the set of hash results.

19. The non-transitory computer-readable storage medium according to claim 16, wherein in the parsing the application client parameter, the one or more processors further perform the steps of:
acquiring a mapping table when a number of the blockchains changes, wherein the mapping table comprises a mapping relationship between the application client parameter and the blockchain identifier, and the application client parameter comprises a service type and a service identifier; and
retrieving the corresponding blockchain identifier in the mapping table according to the mapping relationship and the service type and the service identifier carried in the transaction request, and recording the retrieved blockchain identifier as the selected blockchain identifier.

20. The non-transitory computer-readable storage medium according to claim 16, wherein when the computer-readable instructions are executed by the one or more processors, the one or more processors further perform the steps of:
receiving an event message flow acquisition request sent by the application client;
sending the event message flow acquisition request to the blockchain to which the transaction request is assigned; and
acquiring a corresponding event message flow when the blockchain performs the transaction through the application-level code and returning the event message flow to the application client.