

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 the number 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 number 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

number 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

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 message client calls the external interface of the application-level code in a synchronous or asynchronous manner to acquire the event message flow. The event message client returns the acquired event message flow to the event message 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 message 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 message client. The event message 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 message 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.