

DATA QUERY REQUEST PROCESSING METHOD AND APPARATUS, DEVICE AND STORAGE MEDIUM

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] This application claims priority to Chinese Patent Application No. 202210891780.8, filed with the China National Intellectual Property Administration on July 27, 2022 and entitled "DATA QUERY REQUEST PROCESSING METHOD AND APPARATUS, DEVICE, AND STORAGE MEDIUM", which is incorporated herein by reference in its entirety.

FIELD OF THE TECHNOLOGY

[0002] The present disclosure relates to the field of databases and, in particular, to a data query request processing method and apparatus, a device, and a storage medium.

BACKGROUND OF THE DISCLOSURE

[0003] Data based on a graph database is usually stored by using a distributed data storage system. The distributed data storage system stores the data in a slice manner through a plurality of node devices, to support storage of large-scale data.

[0004] Each node device storing a data slice jointly manages all data stored on the node device using each core of a multi-core processor in the node device. When receiving a data query request for stored data, a node device creates a corresponding thread for the data query request, and processes the data query request through the created thread, to obtain a query result.

[0005] When the distributed data storage system receives parallel data query requests, for example, receives data query requests of second-level neighbors configured to query a start vertex in the graph database, the distributed data storage system concurrently queries data stored in different node devices. The node devices receiving the data query request therefore need to create threads for different data query requests. Problems arises, however, as large quantity of threads needs to be generated, which is increasingly time-consuming.

SUMMARY

[0006] The present disclosure provides a data query request processing method and apparatus, a device, and a storage medium. The technical solutions are as follows.

[0007] According to one aspect of the present disclosure, a data query request processing method applied to a first node device is provided, the first node device being one of at least two node devices in a distributed data storage system based on a graph database. The method includes:

[0008] receiving, by the first node device through a first processing model in the first node device, a data query request transmitted by a second node device, the data query request being configured for querying data related to a first vertex in the graph database, the data query request carrying an identifier of a start vertex, and the first node device storing data of the start vertex;

[0009] transmitting, by the first node device, the data query request to a second processing model in the first node device through the first processing model, the second processing model being a processing model configured to process the data of the start vertex in the first node device;

[0010] determining, by the first node device, a third processing model based on the data query request through the second processing model, the third processing model being configured to process data of the first vertex or an intermediate vertex, and the intermediate vertex being a vertex between the start vertex and the first vertex; and

[0011] transmitting, by the first node device, the data query request to the third processing model through the second processing model.

[0012] According to another aspect of the present disclosure, a data query request processing apparatus is provided. The data query request processing apparatus is one of at least two node devices in a distributed data storage system based on a graph database. The apparatus includes:

[0013] a receiving module, configured to receive, through a first processing model in the apparatus, a data query request transmitted by a second node device, the data query request being configured for querying data related to a first vertex in the graph database, the data query request carrying an identifier of a start vertex, and the apparatus storing data of the start vertex;

[0014] a transmitting module, configured to transmit the data query request to a second processing model in the apparatus through the first processing model, the second processing model being a processing model configured to process the data of the start vertex in the apparatus;

[0015] a determining module, configured to determine a third processing model based on the data query request through the second processing model, the third processing model being configured to process data of the first vertex or an intermediate vertex, and the intermediate vertex being a vertex between the start vertex and the first vertex; and

[0016] the transmitting module being configured to transmit the data query request to the third processing model through the second processing model.

[0017] According to another aspect of the present disclosure, a computer device is provided. The computer device includes a processor and a memory, the memory storing at least one program, which when loaded and executed by the processor, causes the processor to perform the data query request processing method according to the foregoing aspect.

[0018] According to another aspect of the present disclosure, a computer-readable storage medium is provided. The computer-readable storage medium stores at least one program, which when loaded and executed by a processor, causes the processor to perform the data query request processing method according to the foregoing aspect.

[0019] According to another aspect of the present disclosure, a computer program product or a computer program is provided. The computer program product or the computer program includes computer instructions stored in a computer-readable storage medium. A processor of a computer device reads the computer instructions from the computer-readable storage medium, and executes the computer instructions, to cause the computer device to perform the data query request processing method provided in various implementations of the foregoing aspects.

[0020] The technical solutions provided in the present disclosure have at least the following beneficial effects:

[0021] After receiving a data query request, a node device can directly process the data query request through a processing model in the node device, determine a processing model configured to process data of a vertex related to the data query request in a graph database, and determine a processing result of the data query request. In a process of processing the data query request, the data query request is directly processed through the processing model without creating a thread, which can avoid creating a corresponding thread when the data query request is processed, thereby accelerating data query.

BRIEF DESCRIPTION OF THE DRAWINGS

[0022] FIG. 1 is a schematic diagram of an attribute graph of a graph database according to an exemplary embodiment of the present disclosure.

[0023] FIG. 2 is a schematic diagram of a B tree-based data structure according to an exemplary embodiment of the present disclosure.

[0024] FIG. 3 is a schematic structural diagram of a computer system according to an exemplary embodiment of the present disclosure.

[0025] FIG. 4 is a schematic diagram of a data query request processing process according to an exemplary embodiment of the present disclosure.

[0026] FIG. 5 is a schematic flowchart of a data query request processing method according to an exemplary embodiment of the present disclosure.

[0027] FIG. 6 is a schematic flowchart of another data query request processing method according to an exemplary embodiment of the present disclosure.

[0028] FIG. 7 is a schematic diagram of a global consistence hash ring according to an exemplary embodiment of the present disclosure.

[0029] FIG. 8 is a schematic diagram of a local consistence hash ring according to an exemplary embodiment of the present disclosure.

[0030] FIG. 9 is a schematic diagram of a storage structure of a vertex in a graph database according to an exemplary embodiment of the present disclosure.

[0031] FIG. 10 is a schematic structural diagram of a data query request processing apparatus according to an exemplary embodiment of the present disclosure.

[0032] FIG. 11 is a schematic structural diagram of a computer device according to an exemplary embodiment of the present disclosure.

DESCRIPTION OF EMBODIMENTS

[0033] First, terms involved in the present disclosure are introduced.

[0034] Graph database: It is a database that uses a graph structure for semantic query. The graph database represents and stores data by using a vertex, an edge, and an attribute. For example, FIG. 1 is a schematic diagram of an attribute graph of a graph database according to an exemplary embodiment of the present disclosure. As shown in FIG. 1, the attribute graph of the graph database is formed by a vertex (a circle), an edge (an arrow), and an attribute. In FIG. 1, each one of three vertices has a tag of a student, attributes of the students include names, and values of the attributes are respectively a user 1, a user 2, and a user 3. An edge between vertices represents a relationship between the vertices, for example, brother, younger brother, and seen. If an edge between two vertices is bidirectional, it indicates that there is an association relationship between data of the two vertices. If an edge between two vertices is unidirectional, it indicates that there is a unidirectional association relationship between data of the two vertices.

[0035] Balance tree (B tree): It is a self-balanced tree data structure, which can maintain data in order. The B tree-based data structure allows all actions of data searching, sequential access, data insertion, and data deletion to be completed in a logarithm time, that is, as an amount of data increases, time consumed for data processing increases logarithmically rather than exponentially, with relatively high data processing efficiency.

[0036] For example, FIG. 2 is a schematic diagram of a B tree-based data structure according to an exemplary embodiment of the present disclosure. As shown in FIG. 2, the B tree includes seven nodes, and each node is formed by an index of data. A node (6) is referred to as a root node, nodes (4) and (11) are referred to as intermediate nodes, and nodes (1, 2, 3), (5), (7, 8, 9), and (12, 13) are referred to as leaf nodes.

[0037] A B+ tree is an improvement on the B tree. A difference from the B tree lies in that a leaf node of the B+ tree includes indexes of all data. Such a B+ tree data structure has relatively good query performance and has a characteristic of a clear representation in a graph representation, so that the B+ tree data structure is widely applied to the graph database. However, processing of concurrent writing (data indexes of a plurality of pieces of data are written simultaneously) of data under such the tree (the B tree and the B+ tree) data structure is relatively troublesome, and the reason lies in that node splitting and merging are involved in a data writing process. Therefore, when a concurrent write operation is generally performed on the B tree/B+ tree, locking processing of different granularities is performed on the B tree/B+ tree, to sequentially write the B tree/B+ tree once. For example, a global tree is locked to implement a write operation. Currently, a locking manner of a finer granularity can also be configured for dealing with a high concurrent write condition.

[0038] Still referring to FIG. 2, when an index (10) is written into the B tree, the node (7, 8, 9) is split, and an index (8) is merged into the node (11), to form a node (8, 11). When a plurality of data indexes are written concurrently, the B tree needs to be locked, so that a write request is sequentially processed in a sequential write-once manner. However, this causes relatively low data processing efficiency.

[0039] Actor model: It is a concurrent computing mathematical model. The actor model may make a local decision about a to-be-executed task, to respond to a received message (task), and more actor models can also be created, to transmit more messages and determine how to respond to a next received message. A message is directly transmitted between actor models without any medium, and the message is transmitted and processed asynchronously. Communication and interaction between modules is implemented by transmitting messages between different actor

models. The philosophy of the actor model is that everything is an actor, which is similar to the philosophy of "everything is an object" applied by some object-oriented programming languages. Therefore, message transmitting of the actor model better conforms to an object-oriented original intension. The actor model belongs to a concurrent component model. An advanced stage of a concurrent programming paradigm is defined in a component manner, to avoid direct contact of a user with a basic concept such as multithread concurrency or a thread pool.

[0040] Cloud storage is a new concept extended and developed from a concept of cloud computing. A distributed cloud storage system (a storage system for short below) is a storage system that integrates a large quantity of storage devices of different types (the storage device is also referred to as a storage node) in a network by using application software or an application interface through functions such as a cluster application, a grid technology, and a distributed file storage system to cooperatively work, jointly providing data storage and service access function to the outside. For example, a distributed data storage system in this embodiment of the present disclosure is a cloud storage system.

[0041] Currently, a storage method of the storage system includes: creating a logical volume, and distributing a physical storage space to each logical volume when the logical volume is created. The physical storage space may be formed by a storage device or disks of several storage devices. A client stores data in a logical volume, that is, stores the data in a file system, and the file system divides the data into a plurality of parts, each part being an object, and the object not only including the data but also including additional information such as a data identity (ID). The file system writes each object into a physical storage space of the logical volume, and the file system records storage location information of each object, so that when the client requests accessing to the data, the file system can allow the client to access to the data according to storage location information of each object. A process in which the storage system distributes a physical storage space to a logical volume specifically includes: pre-dividing the physical storage space into stripes according to estimation of a capacity of an object stored in the logical volume (the estimation usually has a large margin relative to an actual capacity of a to-be-stored object) and a redundant array of independent disk (RAID), where one logical volume may be understood as one stripe, so that the physical storage space is distributed for the logical volume.

[0042] Database may be considered as an electronic file cabinet, that is, a place for storing an electronic file. A user may perform an operation such as add, query, update, or delete data in the file. The so-called "database" is a data set that is stored together in a particular manner and can be shared by a plurality of users, has as less redundancy as possible, and is independent of an

application program. For example, the database in this embodiment of the present disclosure is a graph database, and the data structure adopts the B tree or B+ tree structure.

[0043] A database management system (DBMS) is a computer software system designed for managing a database and generally has basic functions such as storage, interception, security assurance, and backup. The database management system may be classified according to a database model supported by the database management system, for example, a relational model or an extensible markup language (XML); or may be classified according to a computer type supported by the database management system, for example, a server cluster or a mobile phone; or may be classified according to a query language used by the database management system, for example, a structured query language (SQL) or an XML data query language (X Query); or may be classified according to performance momentum focus, for example, a maximum scale or a highest running speed; or in another classification manner. Regardless of which classification manner is used, some DBMSs can span categories, for example, supporting a plurality of query languages.

[0044] FIG. 3 is a schematic structural diagram of a computer system 300 according to an exemplary embodiment of the present disclosure. As shown in FIG. 3, the computer system 300 includes a first node device 301, a second node device 302, and a third node device 303.

[0045] The second node device 302 may be a server, or a server cluster formed by a plurality of servers, or a virtual server in a cloud computing service center, or the like. In an embodiment, the second node device 302 is a node device configured to schedule a data processing request in a distributed data storage system. For example, the second node device 302 receives an external data query request, and schedules, according to data to be queried by the data query request, the data query request to a node device that stores data that needs to be queried or a node device that stores data related to data that needs to be queried. The second node device 302 may establish a connection to the first node device 301 and the third node device 303 through a wired network or a wireless network.

[0046] The first node device 301 and the third node device 303 may be a server, or a server cluster formed by a plurality of servers, or a virtual server in a cloud computing service center, or the like. The first node device 301 and the third node device 303 store same, different, or partially same data slices, to implement distributed storage of data. When different node devices store the same data, that is, store a data duplicate, for a data duplicate consistency problem, a conflict-free replicated data type (CRDT) may be configured for converting the consistency problem to a calculation problem, and a locking problem in a process of concurrently modifying data can be

reduced in this manner. The first node device 301 may establish a connection to the third node device 303 through a wired network or a wireless network.

[0047] A quantity of node devices in the computer system 300 is only used as an example, and does not limit the computer system 300 provided in this embodiment of the present disclosure.

[0048] FIG. 4 is a schematic diagram of a data query request processing process according to an exemplary embodiment of the present disclosure. As shown in FIG. 4, a second node device is a node device in a distributed data storage system based on a graph database. After receiving a data query request 401, the second node device transmits the data query request 401 to a first processing model 4021 in a first node device 402. The data query request is configured for querying data related to a first vertex in the graph database, the data query request carries an identifier of a start vertex, and the first node device 402 stores data of the start vertex. The first vertex is also referred to as a target vertex, and the target vertex is a vertex configured to process the data query request. Alternatively, the target vertex is a vertex configured to provide a processing result of the data query request.

[0049] The first node device 402 transmits the data query request 401 to a second processing model 4022 in the first node device 402 through the first processing model 4021. The second processing model 4022 is a processing model configured to process the data of the start vertex in the first node device 402. The first node device 402 determines a third processing model based on the data query request 401 through the second processing model 4022. The third processing model is configured to process data of the first vertex or an intermediate vertex, and the intermediate vertex is a vertex between the start vertex and the first vertex.

[0050] In an embodiment, when there is the first vertex in vertices connected to the first node device, the first node device 402 can directly determine the first vertex through the second processing model 4022, and determine the third processing model configured to process the data of the first vertex. When there is no first vertex in the vertices connected to the first node device, the first node device 402 determines, through the second processing model 4022, the intermediate vertex connected to the first vertex, and determines the third processing model configured to process the data of the intermediate vertex. After determining the third processing model, the first node device 402 transmits the data query request to the third processing model through the second processing model 4022. When the third processing model is configured to process the data of the first vertex, the third processing model is configured to process the data query request and transmit a processing result of the data query request to the second node device. When the third processing model is configured to process the data of the intermediate vertex, the third processing model is

configured to continue to determine a fourth processing model by using the foregoing method and continue to transmit the data query request until the processing result of the data query request is obtained.

[0051] In an embodiment, the third processing model belongs to the first node device 402 or a third node device 403. When the third processing model belongs to the first node device 402, the first node device 402 directly transmits the data query request to the third processing model through the second processing model 4022. When the third processing model belongs to the third node device 403, the first node device 402 transmits, through the second processing model 4022, the data query request to a processing model configured to schedule the data query request in the third node device 403, and then the processing model transmits the data query request to the third processing model.

[0052] After a target processing model (for example, the third processing model) configured to process the data of the first vertex is determined in the foregoing manner, the target processing model processes the data query request, and transmits a processing result of the data query request to a processing model configured to schedule the data query request in a node device in which the target processing model is located. Then, the processing model configured to schedule the data query request transmits the processing result of the data query request to the second node device. When a plurality of processing models transmits processing results of the data query request to the second node device, the second node device summarizes the processing results of the data query request transmitted by all the processing models, to finally obtain the processing result of the data query request. In an embodiment, the processing model is an actor model. The "plurality of" is "at least two" and may be understood as two or more.

[0053] The method provided in this embodiment of the present disclosure is applied to a scenario involving a distributed data storage system, a scenario involving big data processing, a scenario involving processing of training data related to a machine learning model, a scenario involving processing of data in applications (including applications in social, a short video, instant messaging, game, electronic commerce, financial, and other fields), a scenario involving data search (for example, a commodity search scenario), and the like.

[0054] For example, when the method provided in this embodiment of the present disclosure is applied to the scenario of processing of data of a social application, a vertex in a graph database represents data related to a user account, an edge between vertices represents that there is an association relationship between user accounts, and an attribute of the edge is configured for reflecting an attribute of the association relationship, for example, a friend relationship, a defriend

relationship, or a shielding relationship. A node device is a back-end server of the social application, and a processing model is an actor model in the back-end server. For example, a start node is a node corresponding to a first user account, and a target node is a node corresponding to a second user account that establishes a friend relationship with the first user account for the longest time in friend accounts of the first user account. By using the method provided in this embodiment of the present disclosure, the second user account meeting a condition of establishing a friend relationship for the longest time can be queried in the friend accounts of the first user account, so that data related to the second user account can be queried.

[0055] For example, when the method provided in this embodiment of the present disclosure is applied to the scenario of processing of data of a short video application, a vertex in a graph database represents data related to a user account, an edge between vertices represents that there is an association relationship between user accounts, and an attribute of the edge is configured for reflecting an attribute of the association relationship, for example, a follow relationship, a liking relationship, or a favorites relationship. A node device is a back-end server of the short video application, and a processing model is an actor model in the back-end server. For example, a start node is a node corresponding to a third user account, and a target node is a node corresponding to a fourth user account following the third user account in user accounts followed by the third user account. By using the method provided in this embodiment of the present disclosure, the fourth user account meeting a condition of following the third user account can be queried in the user accounts followed by the third user account, so that data related to the fourth user account can be queried.

[0056] For example, when the method provided in this embodiment of the present disclosure is applied to the scenario of processing of data of a game application, a vertex in a graph database represents data related to a user account, an edge between vertices represents that there is an interaction behavior between user accounts, and an attribute of the edge is configured for reflecting an attribute of the interaction behavior, for example, reflecting that there is a behavior of trading a virtual item between user accounts. A node device is a back-end server of the game application, and a processing model is an actor model in the back-end server. For example, a start node is a node corresponding to a fifth user account, and a target node is a node corresponding to a sixth user account having a highest price of a traded virtual item in user accounts having behaviors of trading virtual items with the fifth user account. By using the method provided in this embodiment of the present disclosure, the sixth user account meeting a condition that the price of the trade virtual item is the highest can be queried in the user accounts having the behaviors of trading virtual items with the fifth user account, so that data related to the sixth user account can be queried.

[0057] The method provided in this embodiment of the present disclosure is further applied to scenarios such as a cloud technology, artificial intelligence, Internet of vehicles, an intelligent transportation, and driving assistance and is configured for processing related data involved in the scenarios. When the method is applied to the scenario related to the Internet of vehicles, the node device can alternatively be an in-vehicle device, and the in-vehicle device may be a vehicle terminal.

[0058] After receiving a data query request, a node device can directly process the data query request through a processing model in the node device, determine a processing model configured to process data of a vertex (related to the data query request in a graph database), or determine a processing result of the data query request. In a process of processing the data query request, the data query request is directly processed through the processing model without creating a thread, which can avoid creating a corresponding thread when the data query request is processed, thereby accelerating data query.

[0059] FIG. 5 is a schematic flowchart of a data query request processing method according to an exemplary embodiment of the present disclosure. The method is applied to the first node device in the system shown in FIG. 3, and the first node device is one of a plurality of node devices in a distributed data storage system based on a graph database. As shown in FIG. 5, the method includes the following steps.

[0060] **Step 502:** The first node device receives, through a first processing model in the first node device, a data query request transmitted by a second node device.

[0061] The second node device is a node device configured to schedule the data query request in the distributed data storage system, and the second node device may be further configured to schedule data processing requests such as a data write request, a data deletion request, and a data modification request.

[0062] The data query request is configured for querying data related to a first vertex in the graph database, and the data query request carries an identifier of a start vertex. In an embodiment, the data query request is configured for querying the first vertex based on the start vertex, querying data of the first vertex based on the start vertex, querying a path between the start vertex and the first vertex (an edge in the graph database), querying an edge between the start vertex and the first vertex or an attribute of an edge, and querying a submap including the start vertex and the first vertex in the graph database. In an embodiment, the data query request can further carry an identifier of the first vertex, for example, the data query request is configured for querying the path between the start vertex and the first vertex.

[0063] For example, the data query request is configured for querying data of a two-level neighbor of the start vertex, that is, querying data of a vertex connected to an intermediate vertex. The intermediate vertex is connected to the start vertex, and the intermediate vertex is connected to the first vertex. In this case, there are one or more intermediate vertices, and there are one or more first vertices.

[0064] The first node device stores data of the start vertex. After receiving a data query request, the second node device determines a node device storing the data of the start vertex and transmits the data query request to a processing model in the node device.

[0065] The node device includes at least the following two processing models: a processing model for task schedule, for example, a processing model configured to schedule a data query request and a data processing request; and a processing model configured to manage a data slice. One processing model may manage one or more data slices, for example, the first node device may include a processing model configured to manage a data slice of a vertex 1 and a processing model configured to manage data slices of a vertex 2 and a vertex 3.

[0066] The first processing model is a model configured to schedule the data query request (the data processing request) in the first node device, that is, the first processing model is the processing model for task schedule. After receiving the data query request, the first processing model transmits the data query request to a processing model configured to process the data of the start vertex in the first node device. In an embodiment, the first processing model does not have data of a vertex that needs to be processed. There is a processing model having the same function as the first processing model in the node device storing the data of the vertex in the distributed data storage system. In an embodiment, the second node device stores addresses of processing models configured to schedule the data query request in different node devices. After determining the first node device, the second node device transmits the data query request to an address corresponding to the first processing model.

[0067] In an embodiment, there are a plurality of start vertices. In this case, the second node device splits the data query request according to the start vertices, determines a node device corresponding to each data query request, and transmits a corresponding data query request to a corresponding processing model in the determined node device.

[0068] **Step 504:** The first node device transmits the data query request to a second processing model in the first node device through the first processing model.

[0069] The second processing model is a processing model configured to process the data of the start vertex in the first node device, that is, the second processing model is the processing model configured to manage the data slice. In an embodiment, all data slices of a plurality of vertices stored in the first node device have corresponding processing models. After receiving the data query request, the first processing model determines, according to the identifier of the start vertex, a processing model configured to process the data of the start vertex in the first node device, to determine the second processing model.

[0070] In an embodiment, the first node device can determine the second processing model according to a local consistence hash ring through the first processing model, and then transmit the data query request to the second processing model. The local consistence hash ring is configured for reflecting a correspondence between a data slice of each vertex stored in the first node device and a processing model in the first node device. In an embodiment, when there are a plurality of start vertices, the first node device also splits the data query request through the first processing model before transmitting.

[0071] In an embodiment, the first processing model locally stores metadata of an address of each processing model in the first node device. The first node device transmits the data query request to the determined second processing model according to the metadata through the first processing model.

[0072] **Step 506:** The first node device determines a third processing model based on the data query request through the second processing model.

[0073] The third processing model is configured to process data of the first vertex or an intermediate vertex, and the intermediate vertex is a vertex between the start vertex and the first vertex, that is, the third processing model is the processing model configured to manage the data slice. For example, when the start vertex is connected to the first vertex and a query condition in the data query request is met, the third processing model is configured to process the data of the first vertex. When there is also the intermediate vertex between the start vertex and the first vertex, the third processing model is configured to process the data of the intermediate vertex. In an embodiment, the intermediate vertex is connected to the start vertex (there is an edge connecting the intermediate vertex and the start vertex in the graph database).

[0074] In an embodiment, when the second processing model processes the data query request, to obtain some query results of the data query request, for example, querying a path between the start vertex and the first vertex, the first node device transmits the determined query results to the first processing model through the second processing model, then the first node device

transmits the query results to the second node device through the first processing model, and the second node device summarizes the received query results of the data query request.

[0075] The third processing model is a processing model in the first node device or a processing model in a third node device.

[0076] **Step 508:** The first node device transmits the data query request to the third processing model through the second processing model.

[0077] When the start vertex is connected to the first vertex and the query condition in the data query request is met, that is, when the third processing model is configured to process the data of the first vertex, the third processing model is configured to process the data query request and transmit a processing result of the data query request to the second node device. When there is the intermediate vertex between the start vertex and the first vertex, the third processing model is configured to continue to determine a fourth processing model according to the foregoing method and continue to transmit the data query request to the fourth processing model. Then, the fourth processing model continues to process the data query request according to the foregoing method until the processing result of the data query request is finally obtained. The fourth processing model is a model configured to schedule the data query request (the data processing request) in the third node device, that is, the fourth processing model is the processing model for task schedule.

[0078] In an embodiment, the second processing model locally stores metadata of an address of a processing model configured to process data of a vertex connected to the start vertex. The first node device transmits the data query request to the third processing model according to the metadata through the second processing model.

[0079] Based on the foregoing, according to the method provided in this embodiment, after receiving a data query request, a node device can directly process the data query request through a processing model in the node device, determine a processing model configured to process data of a vertex related to the data query request in a graph database, and determine a processing result of the data query request. In a process of processing the data query request, the data query request is directly processed through the processing model without creating a thread, which can avoid creating a corresponding thread when the data query request is processed, thereby accelerating data query.

[0080] FIG. 6 is a schematic flowchart of another data query request processing method according to an exemplary embodiment of the present disclosure. The method is applied to the first node device in the system shown in FIG. 3, and the first node device is one of a plurality of node

devices in a distributed data storage system based on a graph database. As shown in FIG. 6, the method includes the following steps.

[0081] Step 602: The first node device receives, through a first processing model in the first node device, a data query request transmitted by a second node device.

[0082] The second node device is a node device configured to schedule a data query request in the distributed data storage system. The data query request is configured for querying data related to a first vertex in the graph database, and the data query request carries an identifier of a start vertex.

[0083] After receiving a data query request, the second node device determines a node device storing the data of the start vertex and transmits the data query request to a processing model in the node device. The first node device stores data of the start vertex. In an embodiment, each node device in the distributed data storage system stores a data slice of at least one vertex in the graph database, and a correspondence between data slices of vertices stored in the node devices and the node devices is established through a global consistence hash ring. For example, a node device 1 stores a data slice of a vertex 1 in the graph database, a node device 2 stores data slices of a vertex 2 and a vertex 3 in the graph database, and a node device 3 stores a data slice of a vertex 4 in the graph database. A correspondence between the node devices and the vertices in the graph database may be established through the global consistence hash ring.

[0084] For example, a modulo operation can be performed on an identifier of a node device, to obtain a fixed value. A modulo operation can also be performed on an identifier of each vertex in the graph database, to obtain a fixed value. For example, the modulo operation is performed by using 2^{32} . Result values obtained by performing the modulo operation on 2^{32} by using the identifiers of the node devices or the identifiers of the vertices can form a circular ring, and the circular ring may be referred to as the global consistence hash ring. Similar to a clock, a circle of the clock may be understood as a circle formed by 60 points, and the global consistence hash ring may be imagined as a circle formed by 2^{32} points. The values obtained through the modulo operation are locations of different vertices or node devices on the global consistence hash ring. Then, a node device may be found on the global consistence hash ring in a clockwise direction based on a location of a vertex on the global consistence hash ring, and the found first node device is determined as a node device configured to store a data slice of the vertex. 2^{32} configured for the modulo operation is only used as an example, and another value may alternatively be used. " $\wedge$ " represents exponentiation, for example, " 2^{32} " is 2^{32} .

[0085] After receiving the data query request, the second node device determines a second location on the global consistence hash ring according to the identifier of the start vertex, and then can determine the first node device corresponding to the data query request according to the second location and a location of each node device of the distributed data storage system on the global consistence hash ring. In an embodiment, the second node device does not store data of the vertex in the graph database.

[0086] For example, FIG. 7 is a schematic diagram of a global consistence hash ring according to an exemplary embodiment of the present disclosure. As shown in FIG. 7, after receiving the data query request, the second node device calculates a hash value according to the identifier of the start vertex, and determines the second location on the global consistence hash ring according to the hash value. For example, the second location is a location of a vertex 701. Then a 1st node device, that is, a node device 703, is found clockwise on the global consistence hash ring, and the node device 703 is determined as the first node device. A location of a node device on the global consistence hash ring is determined by calculating a hash value of an identifier of the node device. Alternatively, the second location is a location of a vertex 702. In this case, the second node device finds a 1st node device, that is, a node device 704, clockwise on the global consistence hash ring, and in this case, the second node device determines the node device 704 as the first node device.

[0087] The first processing model is a model configured to schedule the data query request in the first node device. After receiving the data query request, the first processing model transmits the data query request to a processing model configured to process the data of the start vertex in the first node device.

[0088] **Step 604:** The first node device determines a second processing model in the first node device through the first processing model.

[0089] The second processing model is a processing model configured to process the data of the start vertex in the first node device. In an embodiment, all data slices of a plurality of vertices stored in the first node device have corresponding processing models. After receiving the data query request, the first processing model determines, according to the identifier of the start vertex, a processing model configured to process the data of the start vertex in the first node device, to determine the second processing model.

[0090] In an embodiment, the first node device stores data slices of a plurality of vertices in the graph database, and a correspondence between each data slice and a processing model in the first node device is established through a local consistence hash ring. The first node device can

determine a first location on the local consistence hash ring according to the identifier of the start vertex through the first processing model. Then, the first node device can determine the second processing model corresponding to the data query request according to the first location and a location of each processing model in the first node device on the local consistence hash ring through the first processing model.

[0091] For example, FIG. 8 is a schematic diagram of a local consistence hash ring according to an exemplary embodiment of the present disclosure. As shown in FIG. 8, after the first processing model receives the data query request, the first node device calculates a hash value according to the identifier of the start vertex through the first processing model, and determines the first location on the local consistence hash ring according to the hash value. For example, the first location is a location of a vertex 801. Then a 1st processing model, that is, a processing model 803, is found clockwise on the local consistence hash ring, and the processing model 803 is determined as the second processing model. A location of a processing model on the local consistence hash ring is determined by calculating a hash value of an identifier of the processing model. Alternatively, the first location is a location of a vertex 802. In this case, the first node device finds a 1st processing model, that is, a processing model 804, clockwise on the local consistence hash ring through the first processing model, and in this case, the first node device determines the processing model 804 as the second processing model through the first processing model.

[0092] **Step 606:** The first node device transmits the data query request to the second processing model in the first node device through the first processing model.

[0093] In an embodiment, the first node device transmits the data query request to a message queue of the second processing model through the first processing model. The message queue of the second processing model is configured to store a task for processing (e.g., a to-be-processed task) of the second processing model. In an embodiment, the second processing model sequentially processes tasks in the message queue until processing of all tasks are completed.

[0094] **Step 608:** The first node device determines, through the second processing model according to a storage structure of a start vertex in a graph database and a query condition in the data query request, a third processing model configured to process data of a first vertex.

[0095] The storage structure of the start vertex in the graph database includes an attribute of the start vertex, an identifier of a vertex having a connection relationship with the start vertex, and an attribute of the connection relationship. In an embodiment, the vertex having the connection relationship with the start vertex may be divided into a vertex connected to the start vertex and having an edge pointing to the start vertex and a vertex connected to the start vertex and having an

edge pointing to the vertex connected to the start vertex. The attribute of the connection relationship includes attributes of edges in different directions.

[0096] In an embodiment, each processing model stores a resource related to a vertex corresponding to the processing model, that is, stores a storage structure of a vertex that the processing model is responsible for processing. For example, a processing model 1 is configured to process data of a vertex 1 and a vertex 4, a processing model 2 is configured to process data of a vertex 2, and a processing model 3 is configured to process data of a vertex 3. The processing model 1 stores a storage structure of the vertex 1 and a storage structure of the vertex 4, the processing model 2 stores a storage structure of the vertex 2, and the processing model 3 stores a storage structure of the vertex 3.

[0097] For example, FIG. 9 is a schematic diagram of a storage structure of a vertex in a graph database according to an exemplary embodiment of the present disclosure. As shown in (a) in FIG. 9, in a graph database, a vertex 1 is respectively connected to a vertex 2, a vertex 3, and a vertex 4. An edge between the vertex 1 and the vertex 2 points to the vertex 1, edges between the vertex 1 and the vertex 3 respectively point to the vertex 1 and the vertex 3, and an edge between the vertex 1 and the vertex 4 points to the vertex 4. The vertex 2 is connected to the vertex 3, and an edge between the vertex 2 and the vertex 3 points to the vertex 2. The vertex 2 is connected to the vertex 4, and an edge between the vertex 2 and the vertex 4 points to the vertex 4. The vertex 3 is connected to the vertex 4, and an edge between the vertex 3 and the vertex 4 points to the vertex 3.

[0098] As shown in (b) in FIG. 9, a processing model 1 is configured to process data of the vertex 1 and the vertex 4, a processing model 2 is configured to process data of the vertex 2, and a processing model 3 is configured to process data of the vertex 3. The processing model 1 stores storage structures of the vertex 1 and the vertex 4, where Vp1 represents an attribute of the vertex 1 (V1), and V1_in includes an identifier of a vertex whose edge points to the vertex 1 in vertices connected to the vertex 1 and an attribute of the edge between the vertex and the vertex 1, for example, Ep3 represents an attribute of the edge pointing to the vertex 1 between the vertex 1 and the vertex 3, and Ep2 represents an attribute of the edge pointing to the vertex 1 between the vertex 1 and the vertex 2. V1_out includes an identifier of a vertex whose edge points to the connected vertex in the vertices connected to the vertex 1 and an attribute of the edge between the vertex and the vertex 1. Vp4 represents an attribute of the vertex 4 (V4), V4_in includes an identifier of a vertex whose edge points to the vertex 4 in vertices connected to the vertex 4, and V4_out includes an identifier of a vertex whose edge points to the connected vertex in the vertices connected to the vertex 4 and an attribute of the edge between the vertex and the vertex 4. The processing model 2

stores a storage structure of the vertex 2, where Vp2 represents an attribute of the vertex 2 (V2), V2_in includes an identifier of a vertex whose edge points to the vertex 2 in vertices connected to the vertex 2, and V2_out includes an identifier of a vertex whose edge points to the connected vertex in the vertices connected to the vertex 2 and an attribute of the edge between the vertex and the vertex 2. The processing model 3 stores a storage structure of the vertex 3, where Vp3 represents an attribute of the vertex 3 (V3), V3_in identifier of a vertex whose edge points to the vertex 3 in vertices connected to the vertex 3, and V3_out includes an identifier of a vertex whose edge points to the connected vertex in the vertices connected to the vertex 3 and an attribute of the edge between the vertex and the vertex 3.

[0099] The first node device queries a vertex that meets the query condition from vertices having the connection relationship with the start vertex, through the second processing model according to one or at least two of the attribute of the start vertex, the vertices having the connection relationship with the start vertex, and the attribute of the connection relationship. When a vertex in the vertices having the connection relationship with the start vertex meets the query condition, the first node device determines, through the second processing model, the vertex meeting the query condition as the first vertex. Then the first node device can determine, through the second processing model, the third processing model configured to process the data of the first vertex. In an embodiment, the second processing model stores information about a processing model configured to process data of a vertex connected to the processing model. After the first vertex connected to the processing model is determined, the third processing model can be determined according to the data.

[00100] For example, still referring to FIG. 9, the data query request is configured for querying a first-level neighbor of the vertex 1 for data of a vertex whose edge points to the vertex 1 and an attribute of the edge meets the query condition. The first node device determines the vertex 2 as the first vertex through the second processing model according to the storage structure of the vertex 1, and then may determine the third processing model configured to process data of the vertex 2.

[00101] **Step 610:** The first node device determines, through the second processing model according to the storage structure of the start vertex in the graph database and the query condition in the data query request, a third processing model configured to process data of an intermediate vertex.

[00102] The intermediate vertex is a vertex between the start vertex and the first vertex. When the first node device queries the storage structure of the start vertex through the second

processing model according to the method in step 608, when no vertex meets the query condition in the vertices having the connection relationship with the start vertex, the first node device determines, through the second processing model, the vertex having the connection relationship with the start vertex as the intermediate vertex, for example, determines all the vertices having the connection relationship with the start vertex as the intermediate vertices. Then the first node device determines, through the second processing model, the third processing model configured to process the data of the intermediate vertices. In an embodiment, the second processing model stores information about a processing model configured to process data of a vertex connected to the processing model. After the intermediate vertex connected to the processing model is determined, the third processing model can be determined according to the data.

[00103] For example, still referring to FIG. 9, the data query request is configured for querying a second-level neighbor of the vertex 1. If the vertex 1 is not directly connected to the vertex 4, the first node device can query, through the second processing model according to the storage structure of the vertex 1, for only the vertex 2 and the vertex 3 that are connected, in this case, determines the vertex 2 and the vertex 3 as intermediate vertices, and determines both a processing model configured to process data of the vertex 2 and a processing model configured to process data of the vertex 3 as the third processing models.

[00104] **Step 612:** The first node device transmits the data query request to the third processing model through the second processing model.

[00105] When the third processing model belongs to the first node device, the first node device transmits the data query request to the third processing model in the first node device through the second processing model.

[00106] When the third processing model belongs to a third node device, the first node device transmits the data query request to a fourth processing model in the third node device through the second processing model, the fourth processing model being a processing model configured to schedule the data query request in the third node device. The third node device is configured to transmit the data query request to the third processing model through the fourth processing model, so that the third processing model can receive the data query request.

[00107] In an embodiment, when the data query request is transmitted across the node devices, for example, when the first node device transmits the data query request to the fourth processing model in the third node device through the second processing model, the data query request is transmitted to the fourth processing model based on a remote procedure call (RPC). The

first node device transmits the data query request to a message queue of the third processing model (the fourth processing model) through the second processing model.

[00108] When the start vertex is connected to the first vertex and the query condition in the data query request is met, that is, when the third processing model is configured to process the data of the first vertex, the third processing model is configured to process the data query request and transmit a processing result of the data query request to the second node device. When there is the intermediate vertex between the start vertex and the first vertex, that is, when the third processing model is configured to process the data of the intermediate vertex, the third processing model is configured to continue to determine the fourth processing model according to the foregoing method and continue to transmit the data query request to the fourth processing model. Then, the fourth processing model continues to process the data query request according to the foregoing method until the processing result of the data query request is finally obtained.

[00109] For example, when the third processing model determines the processing result of the data query request, that is, when the third processing model is configured to process the data of the first vertex, the node device in which the third processing model is located is configured to transmit, through the third processing model, the processing result of the data query request to a processing model configured to schedule the data query request in the node device in which the third processing model is located. Then, the node device in which the third processing model is located is further configured to transmit the processing result of the data query request to the second node device through the processing model configured to schedule the data query request.

[00110] In an embodiment, when a plurality of processing models transmits processing results of the data query request to the second node device, for example, querying paths between the start vertex and the first vertex, a plurality of edges need to be queried and fed back to the second node device. In this case, the second node device summarizes the processing results of the data query request transmitted by all the processing models, to finally obtain the processing result of the data query request.

[00111] In an embodiment, a plurality of threads of each processor core of the first node device are bound to the plurality of processing models in the first node device, so that each thread can independently run a processing model, thereby avoiding resource preemption between the processing models. The processing models in the distributed data storage system are all actor models, for example, the first processing model, the second processing model, the third processing model, and the fourth processing model are the actor models. In an embodiment, the first node device includes a plurality of hard disks configured to store data, and the hard disk has a binding

relationship with a processor core of the second node device. Through the binding relationship, each actor model can have own independent storage and computing resources, and can implement lock-free processing in the data processing process.

[00112] The method provided in this embodiment of the present disclosure may also be configured for processing a data processing request in addition to processing the data query request. When processing the data processing request, the second node device determines, according to an identifier of a vertex carried by the data processing request, a node device storing data of the vertex, and transmits the data processing request to a processing model configured to schedule the data processing request (the data query request) in the node device. Then, the processing model transmits the data processing request to a processing model configured to process the data of the vertex in the node device, and the processing model responsible for processing the data of the vertex processes the data processing request. In the scenario, data of vertices in the graph database is stored in the node device in a slice manner, and different processing models in the node device are responsible for data processing of data of different vertices. When concurrent processing can be performed on the data of different vertices, the database does not need to be locked, and different processing models concurrently process the data of different vertices, to improve data processing efficiency. When a data structure based on a B tree or B+ tree is used, locking of different granularities on the tree can be avoided, thereby improving the data processing efficiency.

[00113] For a synchronous locking problem of conventional data modification, according to the method provided in this embodiment of the present disclosure, a data slice is adjusted according to a feature of an existing multi-core processor. Each data slice is managed by an independent actor model, and the actor model is not bound to a core of a specified central processing unit (CPU). In this way, a logic slice on each conventional node device can be converted into a real physical slice, and data is managed in a finer granularity, which is more adaptable to a scenario of high concurrency modification. In addition, this fine-granularity management manner also makes the scalability more flexible, especially applied to an expansion requirement on a cloud.

[00114] Based on the foregoing, according to the method provided in this embodiment, after receiving a data query request, a node device can directly process the data query request through a processing model in the node device, determine a processing model configured to process data of a vertex related to the data query request in a graph database, and determine a processing result of the data query request. In a process of processing the data query request, the data query request is directly processed through the processing model without creating a thread, which can avoid creating a corresponding thread when the data query request is processed, thereby accelerating data query.

[00115] According to the method provided in this embodiment, a convenient manner of determining the third processing model is further provided by determining the third processing model according to the storage structure of the start vertex in the graph database.

[00116] According to the method provided in this embodiment, when the first vertex is determined according to the storage structure, it is further determined that the third processing model configured to process the first vertex processes the data query request, so that the data query request can be quickly distributed to a corresponding processing model for processing.

[00117] According to the method provided in this embodiment, when the intermediate vertex is determined according to the storage structure, it is further determined that the third processing model configured to process the intermediate vertex processes the data query request, so that the third processing model can continue to query the first vertex according to the data query request.

[00118] According to the method provided in this embodiment, a vertex for which each processing model in each node device is responsible is further maintained through the local consistence hash ring. Therefore, a quantity of processing models in each node device can be dynamically increased or reduced without affecting a relationship between a processing model and a corresponding vertex.

[00119] According to the method provided in this embodiment, the data query request is further transmitted to a message queue of the processing model, and the request is processed immediately after the message queue of the processing model has the message. In this way, request processing efficiency can be improved.

[00120] According to the method provided in this embodiment, a manner of scheduling the data query request is further provided by transmitting the data query request to a processing model configured to schedule the request in different node devices.

[00121] According to the method provided in this embodiment, a manner of transmitting the data query request is further provided by transmitting the data query request to processing models in different node devices.

[00122] According to the method provided in this embodiment, when transmission is performed across node devices, a manner of transmitting the data query request across node devices is further provided by performing transmission in a RPC manner.

[00123] According to the method provided in this embodiment, in a data query process, a manner of directly pushing a query result after the query result is determined implements

high-performance query compared with a conventional manner of pulling a query result level by level.

[00124] According to the method provided in this embodiment, a manner of efficiently determining a data query result is provided by summarizing the processing results of the data query request.

[00125] According to the method provided in this embodiment, by binding the processing model to the thread of the processor core, and by increasing or decreasing the processor core, a function of scaling out/in the processing model at the processor core level can be further implemented.

[00126] According to the method provided in this embodiment, a model configured to process the data query request is further provided by processing the data query request by the actor model.

[00127] A chronological order of steps of the method provided in the embodiments of the present disclosure may be properly adjusted, and the steps may also be correspondingly added or deleted according to a situation. Any variation readily figured out by a person skilled in the art within the technical scope disclosed in the present disclosure shall fall within the protection scope of the present disclosure, and details are not described herein again.

[00128] In a specific example, still referring to FIG. 4, the data query request is configured for querying a second-level neighbor of a vertex m ($id=\$id$), and the data query request may be represented as $Match(m)-[*2]->(n)$, where $id(m)=\$id$ return n . When a user transmits a data query request to a query layer (that is, the second node device) of the distributed data storage system, the query layer transmits, according to a global data slice configuration (the global consistence hash ring), the data query request to a node device storing data of the vertex m . After receiving the data query request, the node device transmits, according to a local data slice rule (the local consistence hash ring) through an actor model configured to perform scheduling, the data query request to a message queue of an actor model responsible for processing the data of the vertex m . After receiving a message (the data query request), the actor model parses the message, and executes a corresponding instruction according to a message instruction. For example, in this case, the actor model obtains all first-level neighbors of the vertex m , and then distributes the message to actor models related to the first-level neighbors by invoking a dispatch vertex function. In addition, data involving a second-level neighbor found by the actor model is transmitted to a message queue of an actor model responsible for query distribution in the node device, feeding back some query results to the query layer. In addition, after receiving the message, other actor models that receive the

message query locally according to information in the message, and return query results to the actor model configured for query distribution in the node device, feeding back some query results to the query layer. When a message is transmitted across node devices, the message is transmitted to the actor model responsible for query distribution in the node device, and the actor model performs the above distribution operation and finally returns a data query result to the query layer. The query process may be analogous to: Map (Actors[DestList ()], GetDestList ()), and the semantics thereof is to apply a method of obtaining a target list (GetDestList ()) to the relevant actor model, and then summarize all returned data to finally obtain the processing result of the data query request.

[00129] FIG. 10 is a schematic structural diagram of a data query request processing apparatus according to an exemplary embodiment of the present disclosure. The apparatus is one of a plurality of node devices in a graph database-based distributed data storage system. As shown in FIG. 10, the apparatus includes the following modules.

[00130] A receiving module 1001 is configured to receive, through a first processing model in the apparatus, a data query request transmitted by a second node device, the data query request being configured for querying data related to a first vertex in the graph database, the data query request carrying an identifier of a start vertex, and the apparatus storing data of the start vertex.

[00131] A transmitting module 1002 is configured to transmit the data query request to a second processing model in the apparatus through the first processing model, the second processing model being a processing model configured to process the data of the start vertex in the apparatus.

[00132] A determining module 1003 is configured to determine a third processing model based on the data query request through the second processing model, the third processing model being configured to process data of the first vertex or an intermediate vertex, and the intermediate vertex being a vertex between the start vertex and the first vertex.

[00133] The transmitting module 1002 is configured to transmit the data query request to the third processing model through the second processing model.

[00134] In a design, the determining module 1003 is configured to:

[00135] determine the third processing model according to a storage structure of the start vertex in the graph database and a query condition in the data query request through the second processing model.

[00136] In a design, the storage structure of the start vertex in the graph database includes an attribute of the start vertex, an identifier of a vertex having a connection relationship with the start

vertex, and an attribute of the connection relationship. The determining module 1003 is configured to:

[00137] query, through the second processing model according to one or at least two of: the attribute of the start vertex, the vertex having the connection relationship with the start vertex, and the attribute of the connection relationship, the vertex having the connection relationship with the start vertex for a vertex meeting the query condition;

[00138] when a vertex in the vertices having the connection relationship with the start vertex meets the query condition, determine, through the second processing model, the vertex meeting the query condition as the first vertex; and

[00139] determine, through the second processing model, the third processing model configured to process the data of the first vertex.

[00140] In a design, the storage structure of the start vertex in the graph database includes an attribute of the start vertex, an identifier of a vertex having a connection relationship with the start vertex, and an attribute of the connection relationship. The determining module 1003 is configured to:

[00141] query a vertex that meets the query condition from vertices having the connection relationship with the start vertex, through the second processing model according to one or at least two of: the attribute of the start vertex, the vertices having the connection relationship with the start vertex, and the attribute of the connection relationship;

[00142] when no vertex meets the query condition in the vertices having the connection relationship with the start vertex, determine, through the second processing model, the vertex having the connection relationship with the start vertex as the intermediate vertex; and

[00143] determine, through the second processing model, the third processing model configured to process the data of the intermediate vertex.

[00144] In a design, the apparatus stores data slices of a plurality of vertices in the graph database, and a correspondence between each of the data slices and the processing model in the apparatus is established through a local consistence hash ring. The determining module 1003 is configured to:

[00145] determine a first location on the local consistence hash ring according to the identifier of the start vertex through the first processing model; and

[00146] determine the second processing model corresponding to the data query request according to the first location and a location of each processing model in the apparatus on the local consistence hash ring through the first processing model.

[00147] In a design, the transmitting module 1002 is configured to:

[00148] transmit the data query request to a message queue of the second processing model through the first processing model, where

[00149] the message queue of the second processing model is configured to store a task for processing of the second processing model.

[00150] In a design, the third processing model belongs to the apparatus. The transmitting module 1002 is configured to:

[00151] transmit the data query request to the third processing model through the second processing model.

[00152] In a design, the third processing model belongs to a third node device. The transmitting module 1002 is configured to:

[00153] transmit the data query request to a fourth processing model in the third node device through the second processing model, where the fourth processing model is a processing model configured to schedule the data query request in the third node device; and

[00154] the third node device is configured to transmit the data query request to the third processing model through the fourth processing model.

[00155] In a design, the transmitting module 1002 is configured to:

[00156] transmit the data query request to the fourth processing model based on a remote procedure call through the second processing model.

[00157] In a design, when the third processing model determines a processing result of the data query request, a node device in which the third processing model is located is configured to transmit, through the third processing model, the processing result of the data query request to a processing model configured to schedule the data query request in the node device in which the third processing model is located.

[00158] The node device in which the third processing model is located is further configured to transmit the processing result of the data query request to the second node device through the processing model configured to schedule the data query request in the node device in which the third processing model is located.

[00159] In a design, the third processing model determines a processing result of the data query request.

[00160] When the third processing model belongs to the first node device, the first node device is configured to transmit the processing result of the data query request to the first processing model through the third processing model, the first processing model being a processing model configured to schedule the data query request in the first node device, and the first node device is further configured to transmit the processing result of the data query request to the second node device through the first processing model; and

[00161] when the third processing model belongs to a third node device, the third node device is configured to transmit the processing result of the data query request to a fourth processing model through the third processing model, the fourth processing model being a processing model configured to schedule the data query request in the third node device, and the third node device is further configured to transmit the processing result of the data query request to the second node device through the fourth processing model.

[00162] In a design, a plurality of processing models transmits processing results of the data query request to the second node device; and

[00163] the second node device is configured to summarize the processing results of the data query request transmitted by the at least two processing models.

[00164] In a design, a plurality of threads of each processor core of the apparatus are bound to a plurality of processing models in the apparatus one by one.

[00165] In a design, the first processing model, the second processing model, and the third processing model are actor models.

[00166] An embodiment of the present disclosure further provides a computer device, including: a processor and a memory, the memory storing at least one program, the at least one program being loaded and executed by the processor to implement the data query request processing method according to the foregoing method embodiments.

[00167] In an embodiment, the computer device is a server. For example, FIG. 11 is a schematic structural diagram of a computer device according to an exemplary embodiment of the present disclosure.

[00168] The computer device 1100 includes a central processing unit (CPU) 1101, a system memory 1104 including a random access memory (RAM) 1102 and a read-only memory (ROM) 1103, and a system bus 1105 connecting the system memory 1104 to the CPU 1101. The computer

device 1100 further includes a basic input/output (I/O) system 1106 assisting in transmitting information between components in the computer device, and a mass storage device 1107 configured to store an operating system 1113, an application program 1114, and another program module 1115.

[00169] The basic I/O system 1106 includes a display 1108 configured to display information and an input device 1109 such as a mouse or a keyboard that is configured to input information by a user. The display 1108 and the input device 1109 are both connected to the central processing unit 1101 through an input/output controller 1110 connected to the system bus 1105. The basic I/O system 1106 may further include the I/O controller 1110 for receiving and processing input from a plurality of other devices such as a keyboard, a mouse, and an electronic stylus. Similarly, the input/output controller 1110 further provides output to a display screen, a printer, or other types of output devices.

[00170] The mass storage device 1107 is connected to the central processing unit 1101 through a mass storage controller (not shown) connected to the system bus 1105. The mass storage device 1107 and an associated computer-readable storage medium thereof provide non-transitory storage for the computer device 1100. That is, the mass storage device 1107 may include a computer-readable medium (not shown) such as a hard disk or a compact disc ROM (CD-ROM) drive.

[00171] Without loss of generality, the computer-readable storage medium may include a computer storage medium and a communication medium. The computer storage medium includes volatile and non-transitory, removable and non-removable media that store information such as computer-readable storage instructions, data structures, program modules, or other data and that are implemented by using any method or technology. The computer storage medium includes a RAM, a ROM, an erasable programmable read-only register (EPROM), an electrically erasable programmable ROM (EEPROM), a flash memory or another solid-state memory technology, a CD-ROM, a digital versatile disc (DVD) or another optical memory, a tape cartridge, a magnetic cassette, a magnetic disk memory, or another magnetic storage device. Certainly, it may be known by a person skilled in the art that the computer storage medium is not limited to the foregoing several types. The system memory 1104 and the mass storage device 1107 may be collectively referred to as a memory.

[00172] The memory stores one or more programs, and the one or more programs are configured to be executed by one or more CPUs 1101. The one or more programs comprise instructions used for implementing the foregoing method embodiments, and the CPU 1101 executes

the one or more programs to implement the method provided in the foregoing method embodiments.

[00173] According to the various embodiments of the present disclosure, the computer device 1100 may further be connected, through a network such as the Internet, to a remote computer device on the network for running. That is, the computer device 1100 may be connected to a network 1112 by a network interface unit 1111 connected to the system bus 1105, or may be connected to another type of network or remote computer device system (not shown) by a network interface unit 1111.

[00174] The memory further includes one or more programs. The one or more programs are stored in the memory and comprise steps to be executed by the computer device in the method provided in the embodiments of the present disclosure.

[00175] An embodiment of the present disclosure further provides a computer-readable storage medium, storing at least one program, when being loaded and executed by one or more processors of a computer device, the at least one program implementing the data query request processing method provided in the foregoing method embodiments.

[00176] The present disclosure further provides a computer program product or a computer program, including computer instructions stored in a computer-readable storage medium. A processor of a computer device reads the computer instructions from the computer-readable storage medium, and executes the computer instructions, to cause the computer device to perform the data query request processing method provided in the foregoing method embodiments.

CLAIMS

1. A data query request processing method, applied to a first node device, the first node device being one of at least two node devices in a distributed data storage system based on a graph database, the method comprising:

receiving, by the first node device through a first processing model in the first node device, a data query request transmitted by a second node device, the data query request being configured for querying data related to a first vertex in the graph database, the data query request carrying an identifier of a start vertex, and the first node device storing data of the start vertex;

transmitting, by the first node device, the data query request to a second processing model in the first node device through the first processing model, the second processing model being a processing model configured to process the data of the start vertex in the first node device;

determining, by the first node device, a third processing model based on the data query request through the second processing model, the third processing model being configured to process data of the first vertex or an intermediate vertex, and the intermediate vertex being a vertex between the start vertex and the first vertex; and

transmitting, by the first node device, the data query request to the third processing model through the second processing model.

2. The method according to claim 1, wherein the determining, by the first node device, the third processing model based on the data query request through the second processing model comprises:

determining, by the first node device, the third processing model according to a storage structure of the start vertex in the graph database and a query condition in the data query request through the second processing model.

3. The method according to claim 2, wherein the storage structure of the start vertex in the graph database comprises an attribute of the start vertex, an identifier of a vertex having a connection relationship with the start vertex, and an attribute of the connection relationship; and

the determining, by the first node device, the third processing model according to the storage structure of the start vertex in the graph database and the query condition in the data query request through the second processing model comprises:

querying a vertex that meets the query condition from vertices having the connection relationship with the start vertex, by the first node device through the second processing model

according to one or at least two of the attribute of the start vertex, the vertices having the connection relationship with the start vertex, and the attribute of the connection relationship;

in response to a vertex in the vertices having the connection relationship with the start vertex meeting the query condition, determining, by the first node device through the second processing model, the vertex meeting the query condition as the first vertex; and

determining, by the first node device through the second processing model, the third processing model configured to process the data of the first vertex.

4. The method according to claim 2, wherein the storage structure of the start vertex in the graph database comprises an attribute of the start vertex, an identifier of a vertex having a connection relationship with the start vertex, and an attribute of the connection relationship; and

the determining, by the first node device, the third processing model according to the storage structure of the start vertex in the graph database and the query condition in the data query request through the second processing model comprises:

querying a vertex that meets the query condition from vertices having the connection relationship with the start vertex, by the first node device through the second processing model according to one or at least two of the attribute of the start vertex, the vertices having the connection relationship with the start vertex, and the attribute of the connection relationship;

in response to no vertex meeting the query condition in the vertices having the connection relationship with the start vertex, determining, by the first node device through the second processing model, the vertex having the connection relationship with the start vertex as the intermediate vertex; and

determining, by the first node device through the second processing model, the third processing model configured to process the data of the intermediate vertex.

5. The method according to any one of claims 1 to 4, wherein the first node device stores data slices of at least two vertices in the graph database, and a correspondence between each of the data slices and a processing model in the first node device is established through a local consistence hash ring; and

the method further comprises:

determining, by the first node device, a first location on the local consistence hash ring according to the identifier of the start vertex through the first processing model; and

determining, by the first node device, the second processing model corresponding to the data

query request according to the first location and a location of each processing model in the first node device on the local consistence hash ring through the first processing model.

6. The method according to any one of claims 1 to 4, wherein the transmitting, by the first node device, the data query request to the second processing model in the first node device through the first processing model comprises:

transmitting, by the first node device, the data query request to a message queue of the second processing model through the first processing model, wherein

the message queue of the second processing model is configured to store a task for processing of the second processing model.

7. The method according to any one of claims 1 to 4, wherein the third processing model belongs to the first node device; and

the transmitting, by the first node device, the data query request to the third processing model through the second processing model comprises:

transmitting, by the first node device, the data query request to the third processing model in the first node device through the second processing model.

8. The method according to any one of claims 1 to 4, wherein the third processing model belongs to a third node device; and

the transmitting, by the first node device, the data query request to the third processing model through the second processing model comprises:

transmitting, by the first node device, the data query request to a fourth processing model in the third node device through the second processing model, wherein the fourth processing model is a processing model configured to schedule the data query request in the third node device; and

the third node device is configured to transmit the data query request to the third processing model through the fourth processing model.

9. The method according to claim 8, wherein the transmitting, by the first node device, the data query request to the fourth processing model in the third node device through the second processing model comprises:

transmitting, by the first node device, the data query request to the fourth processing model based on a remote procedure call through the second processing model.

10. The method according to any one of claims 1 to 4, wherein the third processing model

determines a processing result of the data query request;

in response to the third processing model belonging to the first node device, the first node device is configured to transmit the processing result of the data query request to the first processing model through the third processing model, the first processing model being a processing model configured to schedule the data query request in the first node device, and the first node device is further configured to transmit the processing result of the data query request to the second node device through the first processing model; and

in response to the third processing model belonging to a third node device, the third node device is configured to transmit the processing result of the data query request to a fourth processing model through the third processing model, the fourth processing model being a processing model configured to schedule the data query request in the third node device, and the third node device is further configured to transmit the processing result of the data query request to the second node device through the fourth processing model.

11. The method according to claim 10, wherein at least two processing models transmit processing results of the data query request to the second node device; and

the second node device is configured to summarize the processing results of the data query request transmitted by the at least two processing models.

12. The method according to any one of claims 1 to 4, wherein at least two threads of a processor core of the first node device are bound to at least two processing models in the first node device one after another.

13. The method according to any one of claims 1 to 4, wherein the first processing model, the second processing model, and the third processing model are actor models.

14. A data query request processing apparatus, being one of at least two node devices in a distributed data storage system based on a graph database, the apparatus comprising:

a receiving module, configured to receive, through a first processing model in the apparatus, a data query request transmitted by a second node device, the data query request being configured for querying data related to a first vertex in the graph database, the data query request carrying an identifier of a start vertex, and the apparatus storing data of the start vertex;

a transmitting module, configured to transmit the data query request to a second processing model in the apparatus through the first processing model, the second processing model being a processing model configured to process the data of the start vertex in the apparatus; and

a determining module, configured to determine a third processing model based on the data query request through the second processing model, the third processing model being configured to process data of the first vertex or an intermediate vertex, and the intermediate vertex being a vertex between the start vertex and the first vertex; and

the transmitting module, being configured to transmit the data query request to the third processing model through the second processing model.

15. The apparatus according to claim 14, wherein

the determining module is configured to determine the third processing model according to a storage structure of the start vertex in the graph database and a query condition in the data query request through the second processing model.

16. The apparatus according to claim 15, wherein the storage structure of the start vertex in the graph database comprises an attribute of the start vertex, an identifier of a vertex having a connection relationship with the start vertex, and an attribute of the connection relationship; and

the determining module is configured to:

query a vertex that meets the query condition from vertices having the connection relationship with the start vertex, through the second processing model according to one or at least two of the attribute of the start vertex, the vertices having the connection relationship with the start vertex, and the attribute of the connection relationship;

in response to a vertex in the vertices having the connection relationship with the start vertex meeting the query condition, determine, through the second processing model, the vertex meeting the query condition as the first vertex; and

determine, through the second processing model, the third processing model configured to process the data of the first vertex.

17. The apparatus according to claim 15, wherein the storage structure of the start vertex in the graph database comprises an attribute of the start vertex, an identifier of a vertex having a connection relationship with the start vertex, and an attribute of the connection relationship; and

the determining module is configured to:

query a vertex that meets the query condition from vertices having the connection relationship with the start vertex, through the second processing model according to one or at least two of the attribute of the start vertex, the vertices having the connection relationship with the start vertex, and the attribute of the connection relationship;

in response to no vertex meeting the query condition in the vertices having the connection relationship with the start vertex, determine, through the second processing model, the vertex having the connection relationship with the start vertex as the intermediate vertex; and

determine, through the second processing model, the third processing model configured to process the data of the intermediate vertex.

18. A computer device, comprising a processor and a memory, the memory storing at least one program, which when loaded and executed by the processor, causes the processor to perform the data query request processing method according to any one of claims 1 to 13.

19. A computer-readable storage medium, storing at least one program, which when loaded and executed by a processor, causes the processor to perform the data query request processing method according to any one of claims 1 to 13.

20. A computer program product, comprising computer instructions stored in a computer-readable storage medium; and a processor of a computer device reading the computer instructions from the computer-readable storage medium and executing the computer instructions to cause the computer device to perform the data query request processing method according to any one of claims 1 to 13.