

RANDOM WALKING AND CLUSTER-BASED RANDOM WALKING METHOD, APPARATUS AND DEVICE

TECHNICAL FIELD

[01] The present specification relates to the field of computer software technologies, and in particular, to random walking and a cluster-based random walking method, apparatus and device.

BACKGROUND

[02] With the rapid development of computer and Internet technologies, many services can be performed on the Internet. Graph computing is a common means for processing social online services.

[03] For example, for identification of fraud on accounts in a social risk control service, each user is considered as a node, and if there is a transfer relationship between two users, a line is present between two corresponding nodes. The line may be directionless, or may be directed according to a transfer direction. In this way, graph data including a plurality of nodes and lines may be obtained, to perform graph computing based on the graph data to implement risk control.

[04] A random walk algorithm is a relatively basic and important part of graph computing, and provides support for an upper-level complex algorithm. In existing technologies, the following random walk algorithm is generally used: Any node included in graph data is read in a database, any adjacent node of the node is further read in the database, and so on to implement random walking in the graph data.

[05] Based on the existing technologies, a more efficient random walking solution applicable to large-scale graph data is needed.

SUMMARY

[06] Embodiments of the present specification provide random walking and a cluster-based random walking method, apparatus and device, to resolve the following technical problem: a more efficient random walking solution applicable to large-scale graph data is needed.

[07] To resolve the foregoing technical problem, the embodiments of the present specification are implemented as follows:

[08] An embodiment of the present specification provides a cluster-based random walking method, including:

[09] obtaining, by a cluster, information about each node included in graph data;

[10] generating a hash table according to the information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[11] generating a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[12] An embodiment of the present specification provides a random walking method, including:

[13] obtaining a hash table that is generated according to information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[14] generating a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[15] An embodiment of the present specification provides a cluster-based random walking apparatus. The apparatus belongs to the cluster, and includes:

[16] an obtaining module, for obtaining information about each node included in graph data;

[17] a first generation module, for generating a hash table according to the information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[18] a second generation module, for generating a random sequence according to the hash table, where the random sequence reflects random walking in the graph data.

[19] An embodiment of the present specification provides a random walking apparatus, including:

[20] an obtaining module, for obtaining a hash table that is generated according to information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[21] a generation module, for generating a random sequence according to the hash table, where the random sequence reflects random walking in the graph data.

[22] An embodiment of the present specification provides a cluster-based random walking device. The device belongs to the cluster, and includes:

[23] at least one processor; and

[24] a memory communicatively connected to the at least one processor, where

[25] the memory stores instructions executable by the at least one processor to cause the at least one processor to:

[26] obtain information about each node included in graph data;

[27] generate a hash table according to the information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[28] generate a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[29] An embodiment of the present specification provides a random walking device, including:

[30] at least one processor; and

[31] a memory communicatively connected to the at least one processor, where

[32] the memory stores instructions executable by the at least one processor to cause the at least one processor to:

[33] obtain a hash table that is generated according to information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[34] generate a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[35] The foregoing at least one technical solution used in the embodiments of the present specification can achieve the following beneficial effects: a database originally storing the graph data is accessed less, the hash table is independent of the database after generated, and an adjacent node of a node can be quickly retrieved by using the hash table. The solution is applicable to large-scale graph data and is relatively efficient, and can be more efficient when implemented based on a cluster.

BRIEF DESCRIPTION OF THE DRAWINGS

[36] To describe the technical solutions in the embodiments of this specification or in the prior art more clearly, the following briefly introduces the accompanying drawings required for describing the embodiments or the prior art. Apparently, the accompanying drawings in the following description show merely some embodiments of this specification, and a person

of ordinary skill in the art may still derive other drawings from these accompanying drawings without creative efforts.

[37] FIG. 1 is a schematic diagram of an overall architecture of a solution of the present specification in an actual application scenario;

[38] FIG. 2 is a schematic flowchart of a cluster-based random walking method, according to an embodiment of the present specification;

[39] FIG. 3 is a schematic flowchart of cluster-based hash table generation in an actual application scenario, according to an embodiment of the present specification;

[40] FIG. 4 is a schematic flowchart of cluster-based random sequence generation in an actual application scenario, according to an embodiment of the present specification;

[41] FIG. 5 is a schematic flowchart of a random walking method, according to an embodiment of the present specification;

[42] FIG. 6 is a schematic structural diagram of a cluster-based random walking apparatus corresponding to FIG. 2, according to an embodiment of the present specification; and

[43] FIG. 7 is a schematic structural diagram of a random walking apparatus corresponding to FIG. 5, according to an embodiment of the present specification.

DETAILED DESCRIPTION OF THE INVENTION

[44] Embodiments of the present specification provide random walking and a cluster-based random walking method, apparatus and device.

[45] To enable a person skilled in the art to better understand the technical solutions in this specification, the technical solutions of the embodiments of this specification will be described clearly and thoroughly below with reference to the accompanying drawings of the embodiments of this specification. Apparently, the described embodiments are merely some rather than all of the embodiments of this application. Other variations may be obtained by a person of ordinary skill in the art based on the embodiments of this specification without creative efforts and shall fall within the protection scope of this application.

[46] A solution of the present specification is applicable to clusters and individual machines. Cluster-based processing of large-scale graph data is more efficient because a task (for example, a data reading task or a data synchronization task) can be split and a plurality of machines in the cluster concurrently can execute task parts assigned to them. The following embodiments are mainly described based on a cluster scenario.

[47] The solution may involve one or more clusters. For example, two clusters are involved

in FIG. 1.

[48] FIG. 1 is a schematic diagram of an overall architecture of the solution of the present specification in an actual application scenario. The overall architecture mainly includes three parts: a server cluster, a worker machine cluster, and a database. The database stores graph data for the cluster to read. The server cluster cooperates with the worker machine cluster, to implement random walking in the graph data according to data read from the database.

[49] The architecture in FIG. 1 is not unique but an example. For example, the solution may involve one cluster, and the cluster includes at least one scheduler and a plurality of worker machines. Another solution may involve one worker machine cluster and one server. Machines involved in the solution cooperate with each other to implement random walking in the graph data.

[50] The solution of the present specification is described below in detail.

[51] FIG. 2 is a schematic flowchart of a cluster-based random walking method, according to an embodiment of the present specification. Each step in FIG. 2 is performed by at least one machine (or a program on a machine) in a cluster, and different steps may be performed by different entities.

[52] A procedure in FIG. 2 includes the following steps:

[53] S202: The cluster obtains information about each node included in graph data.

[54] In embodiments of the present specification, information about a node may include: an identifier of the node, an identifier of an adjacent node of the node (which is used as an example in the following), or information that can represent the adjacent node of the node other than the identifier. The information about each node may be obtained at one time, or may be obtained at a plurality of times.

[55] Generally, the original graph data is stored in a database. In this case, the information about each node needs to be read by accessing the database. To avoid repeatedly reading data from the database and increasing the burden of the database, a plurality of machines in the cluster may respectively read information about different nodes. Further, the plurality of machines may concurrently access the database, to quickly obtain the information about the nodes.

[56] For example, each worker machine in a worker machine cluster may concurrently read information about some nodes from the database and process the information, and further synchronize the processed information to a server cluster. Alternatively, each worker machine may directly synchronize the read information about the nodes to the server cluster

for further processing. The processing includes at least generating a hash table.

[57] S204: Generate a hash table according to the information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node.

[58] In embodiments of the present specification, an identifier of any node and an identifier of an adjacent node of the node may constitute a key-value pair in the hash table. Accordingly, an identifier of an adjacent node of any node can be quickly retrieved according to the hash table and an identifier of the node, to facilitate efficient random walking in the graph data.

[59] To support a hash operation performed during retrieval, identifiers of nodes are preferably numbers. For example, magnitudes of the identifiers of the nodes are used to define an order of the nodes. Starting from 0, a node ranking first has an identifier of 0, a node ranking second has an identifier 1, and so on. The following embodiments are described based on the definition in this example.

[60] Certainly, if original identifiers of the nodes are not numbers, the original identifiers may be mapped to numbers based on a one-to-one mapping rule, to generate the hash table.

[61] In embodiments of the present specification, a node may have a plurality of adjacent nodes, a value in a corresponding key-value pair in the hash table may correspondingly include identifiers of the plurality of adjacent nodes, and in the hash table, a data structure that can accommodate a plurality of elements, such as an array or a linked list, may be used to store the value. In this way, it can be convenient to retrieve an identifier of any adjacent node by using the data structure.

[62] The array is used as an example. Each node corresponds to one array, and the array stores an identifier of each adjacent node of the corresponding node. The identifier of each adjacent node is an element in the array. An identifier of an adjacent node can be retrieved through an access operation on an array element.

[63] S206: Generate a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[64] In embodiments of the present specification, the random sequence is a sequence including identifiers of a plurality of nodes. An order of the identifiers in the random sequence is a random walking order. A maximum length of the random sequence is generally determined by a predetermined quantity of random walking steps.

[65] After the hash table is obtained, step S206 may be independently performed a plurality

of times, to obtain a plurality of independent random sequences. For example, each worker machine generates one or more random sequences according to the hash table.

[66] With the method in FIG. 2, the database originally storing the graph data is accessed less frequently. The hash table after being generated is independent of the database. An adjacent node of a node can be quickly retrieved by using the hash table. The solution is applicable to large-scale graph data and is relatively efficient, and can be more efficient as implemented based on a cluster.

[67] Based on the method in FIG. 2, embodiments of the present specification further provide some specific implementations and extended solutions of the method. The following description is provided by using the architecture in FIG. 1 as an example.

[68] In embodiments of the present specification, the cluster may include a server cluster and a worker machine cluster. For S202, the obtaining, by the cluster, information about each node included in graph data may include:

[69] reading, by the worker machine cluster from a database, an identifier of an adjacent node of each node included in the graph data, where each worker machine reads identifiers of adjacent nodes of respective nodes. It should be noted that, if the worker machine cluster does not know an identifier of a node, the worker machine cluster may read the identifier of the node, and read an identifier of an adjacent node of the node according to the identifier (as a primary key in the database) of the node.

[70] For example, it is assumed that there are five nodes with identifiers 0 to 4. The worker machine cluster includes a worker machine 0, a worker machine 1, and a worker machine 2. Each worker machine reads identifiers of adjacent nodes of some nodes from the database. For example, the worker machine 0 reads identifiers (0 and 2) of adjacent nodes of the node 1 and identifiers (1, 3, and 4) of the nodes adjacent to the node 2; the worker machine 1 reads an identifier (1) of an adjacent node of the node 0; and the worker machine 2 reads identifiers (2 and 4) of adjacent nodes of the node 3 and identifiers (2 and 3) of the nodes adjacent to the node 4.

[71] In embodiments of the present specification, each worker machine may generate an incomplete hash table according to an identifier read by a respective worker machine and an identifier of a corresponding node of an adjacent node having the identifier that is read by the respective worker machine.

[72] Further, the worker machine cluster may synchronize incomplete hash tables to the server cluster. Accordingly, the server cluster can obtain a complete hash table consisting of the incomplete hash tables. Specifically, the server cluster may obtain the complete hash

table by specially integrating (for example, splitting a hash table or merging hash tables) the incomplete hash tables; or may not integrate the incomplete hash tables but consider, after the worker machine cluster completes synchronization, all synchronized data as a whole, that is, the complete hash table. Each server in the server cluster may store the complete hash table, or may store only a part of the complete hash table.

[73] The hash table in step S204 may be a complete hash table, or may be an incomplete hash table. It should be noted that, if the hash table in step S204 is an incomplete hash table, subsequent random walking is correspondingly performed in nodes involved in the incomplete hash table (the nodes are only some nodes in the graph data), and any worker machine may generate a random sequence according to an incomplete hash table generated by the worker machine without depending on the synchronization and the server cluster.

[74] An action after the synchronization is further described. The server cluster may further synchronize the complete hash table to each worker machine, so that each worker machine can generate a random sequence according to the complete hash table.

[75] A description is provided based on the foregoing example with reference to FIG. 3. FIG. 3 is a schematic flowchart for generating cluster-based hash table in an actual application scenario, according to an embodiment of the present specification.

[76] In FIG. 3, a data table in the database records an identifier of an adjacent node of each node by using an identifier of the node as a primary key. An adjacent node of a node 0 is a node 1. Adjacent nodes of the node 1 are the node 0 and a node 2. Adjacent nodes of the node 2 are the node 1, a node 3, and a node 4. Adjacent nodes of the node 3 are the node 2 and the node 4. Adjacent nodes of the node 4 are the node 2 and the node 3. As described above, worker machines 0 to 2 preferably may concurrently read identifiers of adjacent nodes of some nodes from the database.

[77] Each worker machine correspondingly generates an incomplete hash table according to identifiers read by the worker machine. A hash table generated by the worker machine 0 includes two key-value pairs, a hash table generated by the worker machine 1 includes one key-value pair, and a hash table generated by the worker machine 2 includes two key-value pairs.

[78] The worker machine cluster synchronizes all generated incomplete hash tables to the server cluster. It can be seen that the server cluster obtains a complete hash table and servers 0 to 2 each store a part of the complete hash table.

[79] The server cluster synchronizes the complete hash table to each worker machine.

Accordingly, each worker machine may independently generate a random sequence according to the complete hash table.

[80] In embodiments of the present specification, for step S206, the generating a random sequence according to the hash table may specifically include:

[81] randomly determining, by a worker machine, an identifier from identifiers of the nodes as an identifier of a target node; retrieving a corresponding value in the hash table by using the identifier of the target node as a key, where the corresponding value includes identifiers of adjacent nodes of the target node; determining a quantity of the identifiers of the adjacent nodes included in the corresponding value; randomly determining a non-negative integer less than the quantity, and obtaining an identifier of a (non-negative integer)th adjacent node included in the corresponding value; and performing iterative calculation by using the (non-negative integer)th adjacent node as a target node, to generate a random sequence including identifiers of target nodes obtained in sequence.

[82] A further description based on the example of FIG. 3 is provided with reference to FIG. 4. FIG. 4 is a flowchart of cluster-based random sequence generation in an actual application scenario, according to an embodiment of the present specification.

[83] It is assumed that the graph data includes N nodes in total, an nth node has an identifier n, and n starts from 0. In FIG. 4, N=5, a complete hash table correspondingly includes five key-value pairs, a value of each key-value pair is stored in an array, and an identifier of an nth adjacent node of a node corresponding to the key is an nth element of the array.

[84] The worker machine generates an integer belonging to $[0, N-1=4]$, that is, the worker machine determines an identifier of a target node of nodes; retrieves a corresponding value in the complete hash table by using the identifier of the target node as a key; performs query to determine a quantity of elements included in an array storing the value, that is, a quantity of identifiers of adjacent nodes included in the value; determines a non-negative integer j less than the quantity; and obtains an identifier of a jth adjacent node of the target node by reading an jth element of the array.

[85] It is assumed that the identifier of the target node is 2 and j=1. Then the target node is the node 2, the array is [1, 3, 4], and an obtained identifier of a first adjacent node of the target node is a first element of the array, that is, 3. Accordingly, random walking from the node 2 to the node 3 is implemented, and then the node 3 is used as a target node for iterative calculation, to continue the random walking. In this way, identifiers of a plurality of nodes sequentially included in the random walking constitute a random sequence.

[86] In FIG. 4, a quantity of random walking steps is preset to 8, and a quantity of batches is preset to 5. A matrix may be used to represent the calculation results. For example, the quantity of random walking steps is a quantity of columns of the matrix, and the quantity of batches is a quantity of rows of the matrix. In this way, each row of the matrix may store a random sequence.

[87] The quantity of random walking steps defines a maximum length of random sequences. Every time a random sequence reaches the maximum length, a next random sequence independent of the random sequence may start to be generated.

[88] The quantity of batches defines a maximum quantity of random sequences that have been generated by each worker machine but not yet written to the database. When the maximum quantity is reached, the worker machine may write a plurality of random sequences that have been generated by the worker machine but not yet written to the database (in a form of a corresponding matrix). For example, in FIG. 5, when a plurality of random sequences that currently have been generated by the worker machine 2 but not yet written reaches the maximum quantity 5, the corresponding matrix may be written to the database.

[89] For example, a first random sequence (3, 4, 3, 2, 4, 2, 3, 2) generated by the worker machine 0 in FIG. 4 represents a process of random walking through the following nodes: the node 3, the node 4, the node 3, the node 2, the node 4, the node 2, the node 3, and the node 2.

[90] Further, a threshold may be preset, to limit a maximum total quantity of random sequences generated by the entire worker machine cluster. When the preset threshold is reached, each worker machine may stop generating a random sequence.

[91] In addition, during actual application, some worker machines in the worker machine cluster may become abnormal, and lose the hash table obtained through synchronization. For example, if the worker machines store the obtained hash table only in a memory, data in the memory may be lost upon crash. In this case, when recovered to normal, the worker machines may re-obtain the lost hash table from the server cluster. The worker machine 2 in FIG. 4 is shown for this case.

[92] The foregoing mainly describes the solution of the present specification based on a cluster scenario. The solution of the present specification is also applicable to other scenarios than the cluster scenario. For example, based on the same idea, an embodiment of the present specification further provides a schematic flowchart of a random walking method, as shown in FIG. 5.

[93] A procedure in FIG. 5 may be performed by a single computing device, or may be a

plurality of computing devices. The procedure includes the following steps:

[94] S502: Obtain a hash table that is generated according to information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node.

[95] In step S502, an entity for generating the hash table is not limited in the present application. Generally, as long as the graph data is not changed, the hash table generated according to the graph data may be repeatedly used.

[96] S504: Generate a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[97] Based on the same idea, embodiments of the present specification further provide apparatuses corresponding to the foregoing methods, as shown in FIG. 6 and FIG. 7.

[98] FIG. 6 is a schematic structural diagram of a cluster-based random walking apparatus corresponding to FIG. 2, according to an embodiment of the present specification. The apparatus belongs to the cluster, and includes:

[99] an obtaining module 601, for obtaining information about each node included in graph data;

[100] a first generation module 602, for generating a hash table according to the information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[101] a second generation module 603, for generating a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[102] Optionally, the cluster includes a server cluster and a worker machine cluster; and

[103] the obtaining, by the obtaining module 601, information about each node included in graph data includes:

[104] reading, by the worker machine cluster from a database, an identifier of an adjacent node of each node included in the graph data, where each worker machine reads identifiers of adjacent nodes of respective nodes.

[105] Optionally, the generating, by the first generation module 602, a hash table according to the information about each node includes:

[106] generating, by each worker machine, an incomplete hash table according to an identifier read by a respective worker machine and an identifier of a corresponding node of an adjacent node having the identifier that is read by the respective worker machine;

- [107] synchronizing, from the worker machine cluster to the server cluster, each incomplete hash table; and
- [108] obtaining, by the server cluster, a complete hash table according to each incomplete hash table.
- [109] Optionally, before the second generation module 603 generates the random sequence according to the hash table, the server cluster synchronizes the complete hash table to each worker machine, so that each worker machine generates a random sequence according to the complete hash table.
- [110] Optionally, the generating, by the second generation module 603, a random sequence according to the hash table specifically includes:
- [111] randomly determining, by a worker machine, an identifier from identifiers of the nodes as an identifier of a target node;
- [112] retrieving a corresponding value in the hash table by using the identifier of the target node as a key, where the corresponding value includes identifiers of adjacent nodes of the target node;
- [113] determining a quantity of the identifiers of the adjacent nodes included in the corresponding value;
- [114] randomly determining a non-negative integer less than the quantity, and obtaining an identifier of a (non-negative integer)th adjacent node included in the corresponding value; and
- [115] performing iterative calculation using the (non-negative integer)th adjacent node as a target node, to generate the random sequence including identifiers of target nodes obtained in sequence.
- [116] Optionally, the corresponding value is stored in an array, an identifier of an nth adjacent node of the target node is an nth element of the array, and n starts from 0; and
- [117] the non-negative integer is j, and the obtaining, by the second generation module 603, an identifier of a (non-negative integer)th adjacent node included in the corresponding value includes:
- [118] obtaining an identifier of a jth adjacent node of the target node by reading a jth element of the array.
- [119] Optionally, the generating, by the worker machine, a random sequence including identifiers of target nodes obtained in sequence includes:
- [120] when a total quantity of the target nodes obtained in sequence reaches a preset

quantity of random walking steps, generating, by the worker machine, the random sequence including the identifiers of the target nodes obtained in sequence.

[121] Optionally, the generating, by the second generation module 603, a random sequence specifically includes:

[122] generating, by each worker machine, a random sequence until a total quantity of generated random sequences reaches a threshold.

[123] Optionally, if the hash table is locally lost, the worker machine re-obtains the hash table from the server cluster.

[124] FIG. 7 is a schematic structural diagram of a random walking apparatus corresponding to FIG. 5, according to an embodiment of the present specification. The apparatus includes:

[125] an obtaining module 701, for obtaining a hash table that is generated according to information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[126] a generation module 702, for generating a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[127] Based on the same idea, an embodiment of the present specification further provides a cluster-based random walking device corresponding to FIG. 2. The device belongs to the cluster, and includes:

[128] at least one processor; and

[129] a memory communicatively connected to the at least one processor, where

[130] the memory stores instructions executable by the at least one processor to cause the at least one processor to:

[131] obtain information about each node included in graph data;

[132] generate a hash table according to the information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[133] generate a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[134] Based on the same idea, an embodiment of the present specification further provides a random walking device corresponding to FIG. 5, including:

[135] at least one processor; and

[136] a memory communicatively connected to the at least one processor, where

[137] the memory stores instructions executable by the at least one processor to cause the at least one processor to:

[138] obtain a hash table that is generated according to information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[139] generate a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[140] Based on the same idea, an embodiment of the present specification further provides a non-transitory computer storage medium corresponding to FIG. 2. The non-transitory computer storage medium stores a computer executable instruction, and the computer executable instruction is set to:

[141] obtain information about each node included in graph data;

[142] generate a hash table according to the information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[143] generate a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[144] Based on the same idea, an embodiment of the present specification further provides a non-transitory computer storage medium corresponding to FIG. 5. The non-transitory computer storage medium stores a computer executable instruction, and the computer executable instruction is set to:

[145] obtain a hash table that is generated according to information about each node, where a key of the hash table includes an identifier of a respective node, and a value of the hash table includes an identifier of an adjacent node of the respective node; and

[146] generate a random sequence according to the hash table, where the random sequence reflects random walks in the graph data.

[147] Embodiments of this specification are described above. Other embodiments fall within the scope of the appended claims. In some embodiments, the actions or steps recorded in the claims may be performed in sequences different from those in the embodiments and an expected result may still be achieved. In addition, the processes depicted in the accompanying drawings do not necessarily require specific sequences or consecutive sequences to achieve an expected result. In some implementations, multitasking and parallel

processing may be feasible or beneficial.

[148] The embodiments of this specification are all described in a progressive manner, for same or similar parts in the embodiments, refer to these embodiments, and descriptions of each embodiment focus on a difference from other embodiments. Especially, an apparatus embodiment, a device embodiment, or a non-transitory computer storage medium embodiment is basically similar to a method embodiment, and therefore is described briefly; for related parts, refer to partial descriptions in the method embodiment.

[149] The apparatus, the device, and the non-transitory computer storage medium provided in the embodiments of this specification correspond to the method, so that the apparatus, the device, and the non-transitory computer storage medium also have beneficial technical effects similar to those of the corresponding method. Since the beneficial technical effects of the method have been described in detail above, the beneficial technical effects of the corresponding apparatus, device, and non-transitory computer storage medium are not described herein again.

[150] In the 1990s, improvements of a technology can be clearly distinguished between hardware improvements (for example, improvements to a circuit structure such as a diode, a transistor, a switch, etc.) and software improvements (improvements to a method procedure). However, with the development of technology, improvements of many method procedures can be considered as direct improvements of hardware circuit structures. Designers almost all program an improved method procedure to a hardware circuit, to obtain a corresponding hardware circuit structure. Therefore, it does not mean that the improvement of a method procedure cannot be implemented by using a hardware entity module. For example, a programmable logic device (PLD) such as a field programmable gate array (FPGA) is a type of integrated circuit whose logic function is determined by a user by programming the device. The designers perform voluntary programming to "integrate" a digital system into a single PLD without requiring a chip manufacturer to design and prepare a dedicated integrated circuit chip. Moreover, nowadays, instead of manually making integrated circuit chips, this programming is mostly implemented by using "logic compiler" software, which is similar to the software compiler used in program development and writing. The original code is written in a specific programming language before compiling, and this language is referred to as a hardware description language (HDL). There are various kinds of HDLs, for example, advanced Boolean expression language (ABEL), altera hardware description language (AHDL), Confluence, Cornell university programming language (CUPL), HDCal, Java

hardware description language (JHDL), Lava, Lola, MyHDL, PALASM, Ruby hardware description language (RHDL), and the like. Currently, the most commonly used HDLs are very-high-speed integrated circuit hardware description language (VHDL) and Verilog. A person skilled in the art should also understand that as long as a method procedure is logically programmed and then programmed to an integrated circuit by using the foregoing hardware description languages, a hardware circuit that implements the logical method procedure can be easily obtained.

[151] The controller can be implemented in any suitable manner, for example, the controller can take the form of, for example, a microprocessor or processor and a computer-readable medium storing computer-readable program code (for example, software or firmware) executable by the processor, a logic gate, a switch, an application-specific integrated circuit (ASIC), a programmable logic controller and an embedded microcontroller. Examples of the controller include, but are not limited to, the following microcontrollers: ARC 625D, Atmel AT91SAM, Microchip PIC18F26K20 and Silicone Labs C8051F320. The memory controller can also be implemented as part of the memory control logic. A person skilled in the art will also appreciate that, in addition to implementing the controller in the form of pure computer-readable program code, it is also possible to implement the controller in the form of a logic gate, switch, application-specific integrated circuit, programmable logic controller, and embedded microcontroller and other forms to achieve the same function. Such a controller can thus be considered as a hardware component and apparatuses included therein for implementing various functions can also be considered as structures inside the hardware component. Alternatively, apparatuses configured to implement various functions can be considered as both software modules implementing the method and structures inside the hardware component.

[152] The system, the apparatus, the module or the unit described in the foregoing embodiments can be specifically implemented by a computer chip or an entity or implemented by a product having a certain function. A typical implementation device is a computer. Specifically, the computer can be, for example, a personal computer, a laptop computer, a cellular phone, a camera phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email device, a game console, a tablet computer, a wearable device, or a combination of any of these devices.

[153] For ease of description, when the apparatus is described, the apparatus is divided into units according to functions, which are separately described. Certainly, in

implementation of this specification, the function of the units may be implemented in a same piece of or multiple pieces of software and/or hardware.

[154] A person skilled in the art should understand that the embodiments of this specification may be provided as a method, a system, or a computer program product. Therefore, the embodiments of this specification may use a form of hardware only embodiments, software only embodiments, or embodiments with a combination of software and hardware. Moreover, the embodiments of this specification may use a form of a computer program product that is implemented on one or more computer-usable storage media (including but not limited to a disk memory, a CD-ROM, an optical memory, and the like) that include computer-usable program code.

[155] This specification is described with reference to the flowchart and/or block diagram of the method, device (system), and computer program product of the embodiments of this specification. It should be understood that computer program instructions can implement each procedure and/or block in the flowcharts and/or block diagrams and a combination of procedures and/or blocks in the flowcharts and/or block diagrams. These computer program instructions may be provided to a general-purpose computer, a special-purpose computer, an embedded processor, or a processor of another programmable data processing device to generate a machine, so that an apparatus configured to implement functions specified in one or more procedures in the flowcharts and/or one or more blocks in the block diagrams is generated by using instructions executed by the general-purpose computer or the processor of another programmable data processing device.

[156] These computer program instructions may also be stored in a computer readable memory that can guide a computer or another programmable data processing device to work in a specific manner, so that the instructions stored in the computer readable memory generate a product including an instruction apparatus, where the instruction apparatus implements functions specified in one or more procedures in the flowcharts and/or one or more blocks in the block diagrams.

[157] These computer program instructions may also be loaded into a computer or another programmable data processing device, so that a series of operation steps are performed on the computer or another programmable data processing device to generate processing implemented by a computer, and instructions executed on the computer or another programmable data processing device provide steps for implementing functions specified in one or more procedures in the flowcharts and/or one or more blocks in the block diagrams.

[158] In a typical configuration, the computer device includes one or more processors (CPUs), an input/output interface, a network interface, and a memory.

[159] The memory may include a form such as a persistent memory, a random-access memory (RAM) and/or a non-transitory memory of computer readable media, for example, a read-only memory (ROM) or a flash memory (RAM). The memory is an example of the computer readable medium.

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

[161] It should also be noted that the terms “include”, “comprise” and any other variants mean to cover the non-exclusive inclusion. Thereby, the process, method, article, or device which include a series of elements not only include those elements, but also include other elements which are not clearly listed, or include the inherent elements of the process, method, article and device. Without further limitation, the element defined by a phrase “include one” does not exclude other same elements in the process, method, article or device which include the element.

[162] This specification can be described in the general context of computer-executable instructions executed by a computer, for example, a program module. Generally, the program module includes a routine, a program, an object, a component, a data structure, and the like for executing a particular task or implementing a particular abstract data type. This specification may also be practiced in distributed computing environments in which tasks are performed by remote processing devices that are connected by using a communications

network. In a distributed computing environment, the program module may be located in both local and remote computer storage media including storage devices.

[163] The embodiments of this specification are all described in a progressive manner, for same or similar parts in the embodiments, refer to these embodiments, and descriptions of each embodiment focus on a difference from other embodiments. Especially, a system embodiment is basically similar to a method embodiment, and therefore is described briefly; for related parts, reference may be made to partial descriptions in the method embodiment.

[164] The foregoing descriptions are merely embodiments of this specification and are not intended to limit this application. For a person skilled in the art, various modifications and variations can be made to this application. Any modification, equivalent replacement, or improvement made without departing from the spirit and principle of this application shall fall within the scope of the claims of this application.

CLAIMS

What is claimed is:

1. A cluster-based random walking method, comprising:
obtaining, by a cluster, information about each node included in graph data;
generating a hash table according to the information about each node, wherein a key of the hash table comprises an identifier of a respective node, and a value of the hash table comprises an identifier of an adjacent node of the respective node; and
generating a random sequence according to the hash table, wherein the random sequence reflects random walks in the graph data.
2. The method according to claim 1, wherein the cluster comprises a server cluster and a worker machine cluster; and
the obtaining, by a cluster, information about each node included in graph data comprises:
reading, by the worker machine cluster from a database, an identifier of an adjacent node of each node included in the graph data, wherein each worker machine reads identifiers of adjacent nodes of respective nodes.
3. The method according to claim 2, wherein the generating a hash table according to the information about each node comprises:
generating, by each worker machine, an incomplete hash table according to an identifier read by a respective worker machine and an identifier of a corresponding node of an adjacent node having the identifier that is read by the respective worker machine;
synchronizing, from the worker machine cluster to the server cluster, each incomplete hash table; and
obtaining, by the server cluster, a complete hash table according to each incomplete hash table.
4. The method according to claim 3, wherein before the generating a random sequence according to the hash table, the method further comprises:
synchronizing, from the server cluster to each worker machine, the complete hash table, so that each worker machine generates a random sequence according to the complete hash table.
5. The method according to claim 2, wherein the generating a random sequence according to the hash table comprises:

randomly determining, by a worker machine, an identifier from identifiers of the nodes as an identifier of a target node;

retrieving a corresponding value in the hash table using the identifier of the target node as a key, wherein the corresponding value comprises identifiers of adjacent nodes of the target node;

determining a quantity of the identifiers of the adjacent nodes included in the corresponding value;

randomly determining a non-negative integer less than the quantity, and obtaining an identifier of a (non-negative integer)th adjacent node included in the corresponding value; and

performing iterative calculation using the (non-negative integer)th adjacent node as a target node, to generate the random sequence comprising identifiers of target nodes obtained in sequence.

6. The method according to claim 5, wherein the corresponding value is stored in an array, an identifier of an nth adjacent node of the target node is an nth element of the array, and n starts from 0; and

the non-negative integer is j, and the obtaining an identifier of a (non-negative integer)th adjacent node included in the corresponding value comprises:

obtaining an identifier of a jth adjacent node of the target node by reading a jth element of the array.

7. The method according to claim 5, wherein the generating a random sequence comprising identifiers of target nodes obtained in sequence comprises:

when a total quantity of the target nodes obtained in sequence reaches a preset quantity of random walking steps, generating the random sequence comprising the identifiers of the target nodes obtained in sequence.

8. The method according to claim 2, wherein the generating a random sequence comprises:

generating, by each worker machine, a random sequence until a total quantity of generated random sequences reaches a threshold.

9. The method according to claim 5, wherein the method further comprises:

if the hash table is locally unavailable at a respective worker machine, re-obtaining, by the respective worker machine, the hash table from the server cluster.

10. A random walking method, comprising:

obtaining a hash table that is generated according to information about each node,

wherein a key of the hash table comprises an identifier of a respective node, and a value of the hash table comprises an identifier of an adjacent node of the respective node; and

generating a random sequence according to the hash table, wherein the random sequence reflects random walks in the graph data.

11. A cluster-based random walking apparatus, wherein the apparatus belongs to a cluster and comprises:

an obtaining module, for obtaining information about each node included in graph data;

a first generation module, for generating a hash table according to the information about each node, wherein a key of the hash table comprises an identifier of a respective node, and a value of the hash table comprises an identifier of an adjacent node of the respective node; and

a second generation module, for generating a random sequence according to the hash table, wherein the random sequence reflects random walks in the graph data.

12. The apparatus according to claim 11, wherein the cluster comprises a server cluster and a worker machine cluster; and

the obtaining, by the obtaining module, information about each node included in graph data comprises:

reading, by the worker machine cluster from a database, an identifier of an adjacent node of each node included in the graph data, wherein each worker machine reads identifiers of adjacent nodes of respective nodes.

13. The apparatus according to claim 12, wherein the generating, by the first generation module, a hash table according to the information about each node comprises:

generating, by each worker machine, an incomplete hash table according to an identifier read by a respective worker machine and an identifier of a corresponding node of an adjacent node having the identifier that is read by the respective worker machine;

synchronizing, from the worker machine cluster to the server cluster, each incomplete hash table; and

obtaining, by the server cluster, a complete hash table according to each incomplete hash table.

14. The apparatus according to claim 13, wherein before the second generation module generates the random sequence according to the hash table, the server cluster synchronizes the complete hash table to each worker machine, so that each worker machine generates a random sequence according to the complete hash table.

15. The apparatus according to claim 12, wherein the generating, by the second

generation module, a random sequence according to the hash table comprises:

- randomly determining, by a worker machine, an identifier from identifiers of the nodes as an identifier of a target node;

- retrieving a corresponding value in the hash table using the identifier of the target node as a key, wherein the corresponding value comprises identifiers of adjacent nodes of the target node;

- determining a quantity of the identifiers of the adjacent nodes included in the corresponding value;

- randomly determining a non-negative integer less than the quantity, and obtaining an identifier of a (non-negative integer)th adjacent node included in the corresponding value; and

- performing iterative calculation using the (non-negative integer)th adjacent node as a target node, to generate the random sequence comprising identifiers of target nodes obtained in sequence.

16. The apparatus according to claim 15, wherein the corresponding value is stored in an array, an identifier of an nth adjacent node of the target node is an nth element of the array, and n starts from 0; and

- the non-negative integer is j, and the obtaining, by the second generation module, an identifier of a (non-negative integer)th adjacent node included in the corresponding value comprises:

- obtaining an identifier of a jth adjacent node of the target node by reading a jth element of the array.

17. The apparatus according to claim 15, wherein the generating, by the worker machine, a random sequence comprising identifiers of target nodes obtained in sequence comprises:

- when a total quantity of the target nodes obtained in sequence reaches a preset quantity of random walking steps, generating, by the worker machine, the random sequence comprising the identifiers of the target nodes obtained in sequence.

18. The apparatus according to claim 12, wherein the generating, by the second generation module, a random sequence comprises:

- generating, by each worker machine, a random sequence until a total quantity of generated random sequences reaches a threshold.

19. The apparatus according to claim 15, wherein if the hash table is locally unavailable at a respective worker machine, the respective worker machine re-obtains the hash table from the server cluster.

20. A random walking apparatus, comprising:

- an obtaining module, for obtaining a hash table that is generated according to information about each node, wherein a key of the hash table comprises an identifier of a respective node, and a value of the hash table comprises an identifier of an adjacent node of the respective node; and

- a generation module, for generating a random sequence according to the hash table, wherein the random sequence reflects random walks in the graph data.

21. A cluster-based random walking device, wherein the device belongs to a cluster and comprises:

- at least one processor; and

- a memory communicatively connected to the at least one processor, wherein the memory stores instructions executable by the at least one processor to cause at least one processor to:

 - obtain information about each node included in graph data;

 - generate a hash table according to the information about each node, wherein a key of the hash table comprises an identifier of a respective node, and a value of the hash table comprises an identifier of an adjacent node of the respective node; and

 - generate a random sequence according to the hash table, wherein the random sequence reflects random walks in the graph data.

22. A random walking device, comprising:

- at least one processor; and

- a memory communicatively connected to the at least one processor, wherein the memory stores instructions executable by the at least one processor to cause the at least one processor to:

 - obtain a hash table that is generated according to information about each node, wherein a key of the hash table comprises an identifier of a respective node, and a value of the hash table comprises an identifier of an adjacent node of the respective node; and

 - generate a random sequence according to the hash table, wherein the random sequence reflects random walks in the graph data.