

PERMISSION SYNCHRONIZATION METHOD AND SYSTEM FOR DISTRIBUTED STORAGE AND DATA WAREHOUSE

TECHNICAL FIELD

5 [0001] The present invention relates to the technical field of permission management, and specifically relates to a permission synchronization method and system for distributed storage and a data warehouse.

BACKGROUND

10 [0002] As data continues to scale, in more and more enterprise-level applications, a hybrid architecture of a distributed storage system and a data warehouse is adopted to store and manage data. In this architecture, data is stored decentrally on different physical nodes, user needs to span a plurality of storage nodes to access data. To ensure data security, it is needed to establish a unified permission management mechanism between the distributed
15 storage systems and the data warehouses to realize dynamic adjustment and synchronization of permissions. This requires a permission management system to accurately identify user access patterns, evaluate the security level of data, and maintain the permission consistency among storage nodes when permissions change.

 [0003] However, there are still some problems in the related technology, a traditional
20 static permission configuration method cannot adapt to the dynamically changing data access needs of the users, resulting in loose or strict permission allocation. In a distributed environment, there is a time-order dependency relationship on permission update of each storage node, but in the related technology, it lacks of systematic analysis on transmission paths of permission change, which easily leads to inconsistent permission update. When
25 there is an exception in the permission update, the related technology lacks of an effective exception detection and automatic repair mechanism, which requires manual intervention and affects the availability of the system. In existing permission synchronization solutions, they lack of full consideration on the impact of data value and degree of association on permission management, making it difficult to achieve accurate permission control.

30 [0004] In conclusion, it is necessary to solve the technical problem of permission synchronization between the distributed storage system and the data warehouse. Specifically, the user access behavior features are analyzed by a machine learning method, and a multi-dimensional permission evaluation model of a data table is established, thus adaptive adjustment of permission rules is realized. By constructing a permission change

dependency graph, the order of permission update between nodes is ensured. Through the permission transmission analysis and automatic repair mechanism, permission exception can be detected and handled in a timely manner, thus improving system reliability. The present invention can solve problems in the related technology.

5

SUMMARY OF THE INVENTION

[0005] The embodiments of the present invention provides a permission synchronization method and system for distributed storage and a data warehouse, aiming to solve the problems in the related technology.

10 [0006] In a first aspect,

[0007] the permission synchronization method for the distributed storage and the data warehouse provided by the embodiment of the present invention includes:

[0008] acquiring user operation data in a distributed storage system, generating a behavior dataset, carrying out feature extraction on the behavior dataset, and computing to
15 obtain a user access pattern feature matrix; inputting the user access pattern feature matrix into a pre-trained machine learning model, computing a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and generating an intelligent directory permission mapping table; reading structure information of a data table in the data warehouse, computing a value score, a field association intensity and a
20 cross-table access influence degree of the data table, and generating a multi-dimensional table permission evaluation model; and carrying out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, and generating an adaptive permission rule base;

[0009] deploying permission synchronization agents on physical nodes in distributed
25 storage, and allocating the adaptive permission rule base to each permission synchronization agent according to a consistency Hash algorithm; monitoring a permission change event, extracting a permission change instruction, matching the permission change instruction with rules in the adaptive permission rule base, generating a rule matching result, computing a influence range of permission change based on the rule matching result,
30 and forming an initial permission change set; carrying out dependency analysis on the initial permission change set, and generating a permission change dependency graph; ordering permission change operations based on a topological structure of the permission change dependency graph, and generating a permission update instruction sequence; writing the permission update instruction sequence into a message queue, and generating a

permission change snapshot containing an execution timestamp;

[0010] reading the permission change snapshot, and extracting execution state information of the permission update instruction sequence; comparing the execution state information with the permission change dependency graph to identify exception nodes of permission update; generating a permission transmission tree based on the positions of the exception nodes in the permission change dependency graph; computing a permission consistency score of each transmission path in the permission transmission tree; when the permission consistency score is less than a preset threshold, extracting a rule matching result corresponding to the exception nodes; generating a permission repair solution according to the rule matching result; transforming the permission repair solution into a repair instruction sequence; executing the repair instruction sequence, and updating a permission state; and writing the repaired permission state and repair process data into the adaptive permission rule base.

[0011] In an optional embodiment,

[0012] the acquiring user operation data in a distributed storage system, generating a behavior dataset, carrying out feature extraction on the behavior dataset, and computing to obtain a user access pattern feature matrix; inputting the user access pattern feature matrix into a pre-trained machine learning model, computing a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and generating an intelligent directory permission mapping table; reading structure information of a data table in the data warehouse, computing a value score, a field association intensity and a cross-table access influence degree of the data table, and generating a multi-dimensional table permission evaluation model; and carrying out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, and generating an adaptive permission rule base includes:

[0013] acquiring user access time, operation type, and access path data in an audit log in the distributed storage system, and pre-processing the user access time, operation type, and access path data by a sliding time window method, and generating the behavior dataset; extracting time-order features, operation features and path features based on the behavior dataset, vectorizing the time-order features, operation features and path feature, and generating the user access pattern feature matrix;

[0014] inputting the user access pattern feature matrix into the pre-trained machine learning model, carrying out product accumulation computation on a time attenuation factor and access times to obtain the directory access frequency weight, computing to

obtain the time sensitivity coefficient through an exponential attenuation function, and computing the permission transmission factor through an intersection-over-union of a directory permission set; constructing a directory access relationship graph, setting the directory as a graph node, setting a user access path as a graph edge, extracting progressive
 5 transmission features between directories through a graph convolution network based on the directory access relationship graph, and correcting the permission transmission factor based on the progressive transmission features; generating the intelligent directory permission mapping table based on the corrected permission transmission factor, the directory access frequency weight and the time sensitivity coefficient;

10 [0015] reading the structure information of the data table in the data warehouse, computing to obtain the sensitivity score, a reference frequency and a data volume score based on the structure information, and carrying out weighted computation on the sensitivity score, the reference frequency and the data volume score to obtain a data table value score; computing to obtain a field association coefficient and a field importance score
 15 based on the structure information, and carrying out weighted computation on the field association degree coefficient and the field importance score to obtain field association intensity; computing to obtain an inter-table access frequency and a degree of dependence based on the structure information, carrying out product accumulation computation on the inter-table access frequency and the degree of dependence to obtain the cross-table access
 20 influence degree, and generating the multi-dimensional table permission evaluation model based on the data table value score, the field association intensity and the cross-table access influence degree; and

[0016] carrying out weighted computation on the data table value score, the field association intensity and the cross-table access influence degree in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model,
 25 generating a permission rule, and storing the permission rule into the adaptive permission rule base.

[0017] In an optional embodiment,

[0018] the constructing a directory access relationship graph, setting the directory as a
 30 graph node, setting a user access path as a graph edge, extracting progressive transmission features between directories through a graph convolution network based on the directory access relationship graph, and correcting the permission transmission factor based on the progressive transmission features includes:

[0019] constructing an initial directory graph based on the access path data in the

behavior dataset, and setting a direct access frequency between directory nodes as an initial weight of an edge;

[0020] computing the shortest path distance between any two directory nodes in the initial directory graph, constructing a spatial adjacency matrix based on the shortest path distance, and multiplying the spatial adjacency matrix by the initial weight of the edge to obtain the spatial association intensity of the directory nodes;

[0021] extracting access time-order information in the behavior dataset, computing an access time interval between directory node pairs, normalizing the access time interval to obtain the time attenuation coefficient, and multiplying the time attenuation coefficient by the spatial association intensity to obtain a spatial-temporal weighted adjacency matrix;

[0022] carrying out sparsification processing on the spatial-temporal weighted adjacency matrix, and reserving the edge with a weight value greater than a preset weight threshold, and obtaining an optimized directory access relationship graph;

[0023] carrying out feature extraction on the optimized directory access relationship graph by a multilayered graph convolution network, extracting local access pattern features by a first graph convolution layer, fusing multi-hop access path features by a second graph convolution layer, and generating progressive transmission feature vectors of the directory nodes by a third graph convolution layer; and

[0024] computing a similarity matrix between the directory nodes based on the progressive transmission feature vectors, and carrying out dot product operation on the similarity matrix and an original permission transmission factor to obtain the corrected permission transmission factor.

[0025] In an optional embodiment,

[0026] the monitoring a permission change event, extracting a permission change instruction, matching the permission change instruction with rules in the adaptive permission rule base, generating a rule matching result, computing an influence range of permission change based on the rule matching result, and forming an initial permission change set; carrying out dependency analysis on the initial permission change set, and generating a permission change dependency graph includes:

[0027] monitoring the permission change event through the permission synchronization agents, extracting an operation type, a target object and a permission attribute from the permission change event, and generating the permission change instruction; constructing a dictionary tree, inserting a rule pattern string into the dictionary tree, and constructing a failure pointer for nodes in the dictionary tree, the failure pointer

pointing to the node corresponding to the longest suffix of the current node; carrying out matching in the dictionary tree along the permission change instruction, when matching fails, returning to the node corresponding to the longest suffix through the failure pointer for continuous matching till matching succeeds or returning to a root node, and generating
 5 a direct influence object set based on objects corresponding to rules which are matched successfully; recursively querying an associated rule chain based on the direct influence object set, and combining each object in the direct influence object set with the corresponding rule chain to form the initial permission change set;

[0028] carrying out dependency analysis on each object in the initial permission
 10 change set and the corresponding rule chain, starting traversing from initial rules in the rule chain, adding accessed rules into an accessed set, and recursively accessing associated rules of the current rules; if the associated rules are in the accessed set, recording a direct dependency relationship between the current rules and the associated rules; when all the associated rules of the rules are accessed, removing the current rules from the accessed set,
 15 and returning to previous rules; computing accessibility among the rules through matrix iteration, and acquiring an indirect dependency relationship; setting permission objects in the initial permission change set as nodes of the dependency graph, and setting the direct dependency relationship and the indirect dependency relationship as edges of the dependency graph; counting the trigger times of the rules in historical records, and dividing
 20 by the total execution times to obtain a trigger probability; counting the number of the permission objects influenced by the rules and dividing by the total number of the permission objects to obtain the influence degree; and setting a product of the trigger probability and the influence degree as the dependency intensity, setting the dependency intensity as an edge weight of the dependency graph, and generating the permission change
 25 dependency graph.

[0029] In an optional embodiment,

[0030] the ordering permission change operations based on a topological structure of the permission change dependency graph, and generating a permission update instruction sequence; writing the permission update instruction sequence into a message queue, and
 30 generating a permission change snapshot containing an execution timestamp includes:

[0031] computing an in-degree of each node of the permission change dependency graph, and adding the nodes with the in-degrees of zero into a candidate permission node set;

[0032] computing the sum of out-edge weights of each candidate permission node in

the candidate permission node set, selecting a target permission node with the maximum sum of the out-edge weights, adding the change operation corresponding to the target permission node to an permission update instruction sequence, removing out-edges of the target permission node, updating in-degree values of subsequent permission nodes adjacent
 5 to the target permission node, and removing the target permission node from the candidate permission node set; when the in-degree values of the subsequent permission nodes are zero, adding the subsequent permission nodes into the candidate permission node set; repeatedly executing the steps until the candidate permission node set is empty;

[0033] allocating the timestamp to each instruction in the permission update instruction
 10 sequence by an incremental counter; creating a permission update theme in a distributed message queue, transmitting a preparation request containing update contents to all participating nodes, and waiting for all the participating nodes to return a preparation completion response; after successful responses of all the participating nodes are received, transmitting a submission request to all the participating nodes, and writing the permission
 15 update instruction sequence in the permission update theme; and transforming a node relationship and edge weight information of the permission update instruction sequence, the timestamps and the permission change dependency graph into a binary format, and generating the permission change snapshot.

[0034] In an optional embodiment,

20 [0035] the reading the permission change snapshot, and extracting execution state information of the permission update instruction sequence; comparing the execution state information with the permission change dependency graph to identify exception nodes of permission update; generating a permission transmission tree based on the positions of the exception nodes in the permission change dependency graph includes:

25 [0036] constructing the permission change snapshot into a permission state matrix, the permission state matrix being formed by permission node rows, time-order columns and matrix elements of execution state information; carrying out difference operation on the permission state matrix and the permission change dependency graph to obtain a state deviation matrix; when a deviation value in the state deviation matrix exceeds a preset state
 30 deviation threshold, marking corresponding permission nodes as exception nodes;

[0037] constructing a forward transmission layer and a reverse transmission layer of the permission change dependency graph based on the exception nodes, the forward transmission layer downwards progressively traversing in a layer-by-layer way through a permission inheritance relationship, and the reverse transmission layer upwards backtracks

and traverses in a layer-by-layer way through a permission dependency relationship; on each of the forward transmission layer and the reverse transmission layer, computing and determining a transmission influence factor according to the state deviation value of the exception nodes, a topological distance between the nodes and a cumulative attenuation
 5 coefficient of a path; when the transmission influence factor is less than a preset truncation threshold, determining a transmission boundary of the corresponding layer, and stopping traversing in the corresponding direction; and combining the traversal results of the forward transmission layer and the reverse transmission layer, and generating the permission transmission tree.

10 [0038] In an optional embodiment,

[0039] the computing a permission consistency score of each transmission path in the permission transmission tree; when the permission consistency score is less than a preset threshold, extracting a rule matching result corresponding to the exception nodes; generating a permission repair solution according to the rule matching result; transforming
 15 the permission repair solution into a repair instruction sequence; executing the repair instruction sequence, and updating a permission state; and writing the repaired permission state and repair process data into the adaptive permission rule base includes:

[0040] for each transmission path in the permission transmission tree, computing the permission consistency score through the permission numerical value difference between
 20 the transmission influence factor on the transmission path and an adjacent permission node; when the permission consistency score is less than a preset score threshold, extracting a rule matching result corresponding to the exception node;

[0041] acquiring a rule weight value and a rule influence direction of each rule from the rule matching result, computing a correction direction of a current permission value and
 25 a target permission value of the exception node, sorting by descending according to the rule weight value, and selecting the rule with the maximum rule weight value and the rule influence direction the same as the correction direction as a repair rule; applying the repair rule to the exception node, and determining a rule repair parameter based on a state deviation value of the exception node; generating the repair instruction sequence according
 30 to the repair rule and the repair parameter;

[0042] executing the repair instruction sequence in a preset time window, and recording execution start time, execution completion time, an execution state code and a state variation after execution of each repair instruction; when detecting an exception execution state code or that the state variation does not meet a preset variation threshold,

trigger the corresponding repair instruction to be executed again;

[0043] computing state deviation improvement degrees before and after execution of the repair instruction sequence, a state fluctuation condition in the execution process and an execution success rate of the repair instruction; and writing the state deviation
5 improvement degrees, the state fluctuation condition, the execution success rate and the permission state after repair into the adaptive permission rule base.

[0044] In a second aspect,

[0045] the permission synchronization system for the distributed storage and the data warehouse provided by the embodiment of the present invention includes:

10 [0046] a first unit, configured to acquire user operation data in a distributed storage system, generate a behavior dataset, carry out feature extraction on the behavior dataset, and compute to obtain a user access pattern feature matrix; input the user access pattern feature matrix into a pre-trained machine learning model, compute a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and
15 generate an intelligent directory permission mapping table; read structure information of a data table in the data warehouse, compute a value score, a field association intensity and a cross-table access influence degree of the data table, and generate a multi-dimensional table permission evaluation model; and carry out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission
20 evaluation model, and generate an adaptive permission rule base;

[0047] a second unit, configured to deploy permission synchronization agents on physical nodes in distributed storage, and allocate the adaptive permission rule base to each permission synchronization agent according to a consistency Hash algorithm; monitor a permission change event, extracting a permission change instruction, match the permission
25 change instruction with rules in the adaptive permission rule base, generate a rule matching result, compute a influence range of permission change based on the rule matching result, and form an initial permission change set; carry out dependency analysis on the initial permission change set, and generate a permission change dependency graph; order permission change operations based on a topological structure of the permission change
30 dependency graph, and generate a permission update instruction sequence; write the permission update instruction sequence into a message queue, and generate a permission change snapshot containing an execution timestamp; and

[0048] a third unit, configured to read the permission change snapshot, and extract execution state information of the permission update instruction sequence; compare the

execution state information with the permission change dependency graph to identify exception nodes of permission update; generate a permission transmission tree based on the positions of the exception nodes in the permission change dependency graph; compute a permission consistency score of each transmission path in the permission transmission tree; when the permission consistency score is less than a preset threshold, extract a rule matching result corresponding to the exception nodes; generate a permission repair solution according to the rule matching result; transform the permission repair solution into a repair instruction sequence; execute the repair instruction sequence, and update a permission state; and write the repaired permission state and repair process data into the adaptive permission rule base.

[0049] In a third aspect,

[0050] an electronic device provided by the embodiment of the present invention includes:

[0051] a processor; and

[0052] a memory configured to store processor executable instructions;

[0053] the processor is configured to call instructions stored in the memory to execute the abovementioned method.

[0054] In a fourth aspect,

[0055] the embodiment of the present invention provides a computer-readable storage medium having computer program instructions stored therein, when executed by a processor, the computer program instructions execute the abovementioned method.

[0056] According to the embodiments of the present invention, the user access pattern is learned by the machine learning model, the permission mapping table and the rule base are intelligently generated, and the permission update operations are sorted based on the dependency graph, therefore, unnecessary permission change operations are reduced and the permission synchronization efficiency is improved; fine-grained permission control is realized by the multi-dimensional table permission evaluation model and the consistency Hash algorithm, and permission errors are repaired in time by permission change dependency analysis and exception node identification, and thus permission leakage and abuse are effectively prevented; and the permission rule base is automatically generated, permission exception is automatically identified and repaired, and the permission state is automatically updated without manual intervention, so that the complexity and cost of permission management are reduced and the automation of the permission management is realized.

BRIEF DESCRIPTION OF THE DRAWINGS

[0057] FIG. 1 is a flowchart of a permission synchronization method for distributed storage and a data warehouse according to an embodiment of the present invention; and

5 [0058] FIG. 2 is a flowchart of a permission synchronization system for distributed storage and a data warehouse according to an embodiment of the present invention.

DETAILED DESCRIPTION

[0059] In order to make the purpose, technical solution and advantages of the
10 embodiment of the present invention clearer, the technical solutions in the embodiments of the present invention will be clearly and completely described in combination with the accompanying drawings in the embodiments of the present invention. Definitely, the described embodiments are only a part of the embodiments of the present invention, not all of them. Based on the embodiments in the present invention, all other embodiments
15 obtained by those of ordinary skill in the art without carrying out creative labor fall within the scope of protection of the present invention.

[0060] The technical solutions of the present invention are described in detail by specific embodiments below. The following specific embodiments can be combined with each other, and the same or similar concepts or processes may not be repeated in some
20 embodiments.

[0061] FIG. 1 is a flowchart of a permission synchronization method for distributed storage and a data warehouse according to an embodiment of the present invention, as shown in FIG. 1, the method includes:

[0062] S101: Acquire user operation data in a distributed storage system, generate a
25 behavior dataset, carry out feature extraction on the behavior dataset, and compute to obtain a user access pattern feature matrix; input the user access pattern feature matrix into a pre-trained machine learning model, compute a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and generate an intelligent directory permission mapping table; read structure information of a data table in the data
30 warehouse, compute a value score, a field association intensity and a cross-table access influence degree of the data table, and generate a multi-dimensional table permission evaluation model; and carry out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, and generate an adaptive permission rule base.

[0063] In this embodiment, the user operation data is acquired and the behavior dataset is generated, feature extraction is carried out, so that the access pattern of a user can be accurately described, which provides a data driving basis for subsequent permission management; the intelligent directory permission mapping table is generated by computing
 5 the directory access frequency weight, the time sensitivity coefficient and the permission transmission factor, thus the permission setting of a directory can be dynamically and intelligently adjusted, thereby improving the flexibility and adaptive capability of the permission management; the multi-dimensional permission assessment model is generated by computing the value score, the field association intensity and the cross-table access
 10 influence degree of the data table, thus the permission dependency relationship among the data tables can be comprehensively analyzed and the permission setting is ensured to be more reasonable and accurate; and in combination with the intelligent directory permission mapping table and the multi-dimensional table permission assessment model, the adaptive permission rule base is generated by weighted computation, and the permission rules can
 15 be dynamically adjusted according to system changes, thereby improving the automation and intelligence levels of the permission management.

[0064] S102: Deploy permission synchronization agents on physical nodes in distributed storage, and allocate the adaptive permission rule base to each permission synchronization agent according to a consistency Hash algorithm; monitor a permission
 20 change event, extracting a permission change instruction, match the permission change instruction with rules in the adaptive permission rule base, generate a rule matching result, compute a influence range of permission change based on the rule matching result, and form an initial permission change set; carry out dependency analysis on the initial permission change set, and generate a permission change dependency graph; order
 25 permission change operations based on a topological structure of the permission change dependency graph, and generate a permission update instruction sequence; write the permission update instruction sequence into a message queue, and generate a permission change snapshot containing an execution timestamp.

[0065] In this embodiment, the permission synchronization agents are deployed on the
 30 physical nodes in distributed storage, and the adaptive permission rule base is dynamically allocated in combination with the consistency Hash algorithm, thus the permission rules can be efficiently and uniformly distributed to each node, and synchronization and consistency of distributed permissions are realized; the permission synchronization agents can monitor the permission change event in real time, automatically extract and match the

permission change instruction, compute the influence range of permission change based on the rule matching result and quickly form the initial permission change set, therefore, accurate control and response to the permission change are realized; through dependency analysis and generation of the permission change dependency graph, the system can understand the mutual relationship and the influence path of the permission change, thus ensuring the rationality of the permission change and avoiding permission conflicts and inconsistency; the permission change operations are sorted based on the topological structure of the permission change dependency graph, and the permission update instruction sequence is generated, so that the permission update is executed orderly and efficiently and the conflicts and delay of the permission change are reduced; and the permission change snapshot containing the execution timestamp is generated, which provides a complete permission change historical record, thereby assisting permission auditing and problem troubleshooting and ensuring the transparency and traceability of the permission change.

[0066] S103: Read the permission change snapshot, and extract execution state information of the permission update instruction sequence; compare the execution state information with the permission change dependency graph to identify exception nodes of permission update; generate a permission transmission tree based on the positions of the exception nodes in the permission change dependency graph; compute a permission consistency score of each transmission path in the permission transmission tree; when the permission consistency score is less than a preset threshold, extract a rule matching result corresponding to the exception nodes; generate a permission repair solution according to the rule matching result; transform the permission repair solution into a repair instruction sequence; execute the repair instruction sequence, and updating a permission state; and write the repaired permission state and repair process data into the adaptive permission rule base.

[0067] In this embodiment, by reading the permission change snapshot and comparing the execution state information, the system can quickly identify the exception nodes in permission update, thus ensuring the accuracy of the permission change process and preventing permission setting errors or conflicts; the permission transmission tree is generated based on the exception nodes, and the permission consistency score of each transmission path is computed, which is conducive to comprehensively evaluating the influence range and consistency of permission change, thereby avoiding potential permission transmission problem; when the permission consistency score is less than the

threshold, the system can automatically extract the rule matching result and generate a permission repair solution, thus ensuring the correctness and consistency of permission configuration, avoiding manual intervention, and improving the management efficiency; the repair solution is transformed into the repair instruction sequence and is executed, and
 5 the permission state is automatically updated, so that dynamic adjustment and optimization of system permission are ensured and the permission configuration is ensured to continuously meet security policies and actual demands; and the repaired permission state and repair process data will be written in the adaptive permission rule base, a complete permission change historical record will be provided, which improves the transparency,
 10 audit and traceability of permission management, and assists compliance check and problem tracking.

[0068] In an optional embodiment, the acquiring user operation data in a distributed storage system, generating a behavior dataset, carrying out feature extraction on the behavior dataset, and computing to obtain a user access pattern feature matrix; inputting the
 15 user access pattern feature matrix into a pre-trained machine learning model, computing a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and generating an intelligent directory permission mapping table; reading structure information of a data table in the data warehouse, computing a value score, a field association intensity and a cross-table access influence degree of the data
 20 table, and generating a multi-dimensional table permission evaluation model; and carrying out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, and generating an adaptive permission rule base includes:

[0069] acquiring user access time, operation type, and access path data in an audit log
 25 in the distributed storage system, and pre-processing the user access time, operation type, and access path data by a sliding time window method, and generating the behavior dataset; extracting time-order features, operation features and path features based on the behavior dataset, vectorizing the time-order features, operation features and path feature, and generating the user access pattern feature matrix;

30 [0070] inputting the user access pattern feature matrix into the pre-trained machine learning model, carrying out product accumulation computation on a time attenuation factor and access times to obtain the directory access frequency weight, computing to obtain the time sensitivity coefficient through an exponential attenuation function, and computing the permission transmission factor through an intersection-over-union of a

directory permission set; constructing a directory access relationship graph, setting the directory as a graph node, setting a user access path as a graph edge, extracting progressive transmission features between directories through a graph convolution network based on the directory access relationship graph, and correcting the permission transmission factor
 5 based on the progressive transmission features; generating the intelligent directory permission mapping table based on the corrected permission transmission factor, the directory access frequency weight and the time sensitivity coefficient;

[0071] reading the structure information of the data table in the data warehouse, computing to obtain the sensitivity score, a reference frequency and a data volume score
 10 based on the structure information, and carrying out weighted computation on the sensitivity score, the reference frequency and the data volume score to obtain a data table value score; computing to obtain a field association coefficient and a field importance score based on the structure information, and carrying out weighted computation on the field association degree coefficient and the field importance score to obtain field association
 15 intensity; computing to obtain an inter-table access frequency and a degree of dependence based on the structure information, carrying out product accumulation computation on the inter-table access frequency and the degree of dependence to obtain the cross-table access influence degree, and generating the multi-dimensional table permission evaluation model based on the data table value score, the field association intensity and the cross-table access
 20 influence degree; and

[0072] carrying out weighted computation on the data table value score, the field association intensity and the cross-table access influence degree in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, generating a permission rule, and storing the permission rule into the adaptive permission
 25 rule base.

[0073] In a specific implementation, data such as the user access time, operation type, and access path are collected from the audit log of the distributed storage system. For example, a user A accessed the /data/sales/202310 directory at 8:00 a.m. on October 27, 2023 and executed a read operation; a user B accessed the /data/marketing/202310
 30 directory at 8:05 a.m. on October 27, 2023 and executed a write operation. Such information will be recorded and collected.

[0074] Then, the sliding time window method is used for preprocessing the collected data to generate the behavioral dataset. For example, if the time window is set to be one day, the user operation data within one day will be aggregated. The read operation of the user A

accessing the /data/sales/202310 directory on October 27, 2023, and the write operation of the user B accessing the /data/marketing/202310 Directory on the same day will be included in the behavior dataset on October 27, 2023.

[0075] Based on the generated behavior dataset, the time-order features, operation features, and path features are extracted. The time-order features include access time period (morning, afternoon, evening), and access frequency (access times per hour), etc.; the operation features include read, write, delete, etc.; and the path features include hierarchical depth of the accessed directory, the name of the accessed directory, etc. These features are then vectorized to form the user access pattern feature matrix. For example, the feature vector of the user A can be expressed as [morning, twice an hour, read, 3 layers, sales], and the feature vector of the user B can be expressed as [morning, once every hour, write, 3 layers, marketing].

[0076] The user access pattern feature matrix is inputted into the pre-trained machine learning model, and the directory access frequency weight, the time sensitivity coefficient and the permission transmission factor are computed. The directory access frequency weight is obtained by accumulation calculation using the time attenuation factor and the access times. For example, the access weight of the latest 7 days is higher than that of the access 7 days ago. The time sensitivity coefficient is computed through the exponential decay function. For example, the closer the access time is, the higher the weight is. The permission transmission factor is computed through the intersection-over-union of the directory permission set. For example, if a user has a read permission for a parent directory, the user possibly has a read permission for subdirectories.

[0077] To compute the permission transmission factor more accurately, the directory access graph is constructed. The directory is set as the graph node and the user access path is set as the graph edge. For example, if the user accesses /data/sales/202310, the three directories /data, /data/sales, and /data/sales/202310 form a path that is connected by edges in the graph. The graph convolutional network is used for extracting the progressive transmission features among the directories, and the permission transmission factors are corrected based on these features. For example, if a user frequently accesses the subdirectories in the /data/sales directory, the permission transmission factor of the /data/sales directory will be higher.

[0078] The intelligent directory permission mapping table is generated based on the corrected permission transmission factor, the directory access frequency weight and the time sensitivity coefficient. The table records the access frequency weight, the time

sensitivity coefficient and the permission transmission factor of each directory.

[0079] The structure information of the data table in the data warehouse is read, and the value score, the field association intensity and the cross-table access influence degree of the data table are computed. The value score of the data table is obtained by weighted
5 computation using the sensitivity score, the reference frequency and the data volume score. The field association intensity is obtained by weighted computation using the field association degree coefficient and the field importance score. The cross-table access influence degree is obtained by accumulation computation using the inter-table access frequency and the dependency degree.

10 [0080] The multi-dimensional table permission evaluation model is generated based on the value score, the field association intensity and the cross-table access influence degree of the data table.

[0081] Finally, weighted computation is carried out on the data table value score, the field association intensity and the cross-table access influence degree in the intelligent
15 directory permission mapping table and multi-dimensional table permission evaluation model to generate the permission rules, and the rules are stored in the adaptive permission rule base.

[0082] In this embodiment, by analyzing the user access behaviors, the data access permission of the user can be controlled more accurately, thus sensitive data leakage is
20 prevented; the adaptive permission rule base can automatically adjust the permission rules according to the user behaviors, so that manual intervention is reduced and the management cost is reduced; and the user can acquire corresponding access permissions according to actual needs, which avoids unnecessary permission application processes and improves the user experience.

25 [0083] In an optional embodiment, the constructing a directory access relationship graph, setting the directory as a graph node, setting a user access path as a graph edge, extracting progressive transmission features between directories through a graph convolution network based on the directory access relationship graph, and correcting the permission transmission factor based on the progressive transmission features includes:

30 [0084] constructing an initial directory graph based on the access path data in the behavior dataset, and setting a direct access frequency between directory nodes as an initial weight of an edge;

[0085] computing the shortest path distance between any two directory nodes in the initial directory graph, constructing a spatial adjacency matrix based on the shortest path distance, and

multiplying the spatial adjacency matrix by the initial weight of the edge to obtain the spatial association intensity of the directory nodes;

[0086] extracting access time-order information in the behavior dataset, computing an access time interval between directory node pairs, normalizing the access time interval to obtain
5 the time attenuation coefficient, and multiplying the time attenuation coefficient by the spatial association intensity to obtain a spatial-temporal weighted adjacency matrix;

[0087] carrying out sparsification processing on the spatial-temporal weighted adjacency matrix, and reserving the edge with a weight value greater than a preset weight threshold, and obtaining an optimized directory access relationship graph;

10 [0088] carrying out feature extraction on the optimized directory access relationship graph by a multilayered graph convolution network, extracting local access pattern features by a first graph convolution layer, fusing multi-hop access path features by a second graph convolution layer, and generating progressive transmission feature vectors of the directory nodes by a third graph convolution layer; and

15 [0089] computing a similarity matrix between the directory nodes based on the progressive transmission feature vectors, and carrying out dot product operation on the similarity matrix and an original permission transmission factor to obtain the corrected permission transmission factor.

[0090] In a specific embodiment, firstly, the behavior dataset of the user in the system is collected, and the dataset contains the path and time information of the user accessing
20 different directories. For example, the user A accessed the directory/home/user/documents at 10:00 a.m. on January 1, 2024, and then accessed the directory/home/user/pictures at 10:05 a.m..

[0091] Secondly, the initial directory graph is constructed based on the collected access path data. Each directory in the system is set as a node in the graph, and if the user
25 directly accesses a directory B from a directory A, a directed edge will be established between the directory A and the directory B. The initial weight of the edge is set as the frequency of direct access from the directory A to the directory B. For example, if the user accesses the directory/home/user/documents from the directory/home/user 10 times, the initial weight of that side will be 10.

30 [0092] Thirdly, the shortest path distance between any two directory nodes in the initial directory graph is computed. The shortest path distance can be computed by a breadth-first search algorithm. Based on the computed shortest path distance, the spatial adjacency matrix is constructed. An element value of the spatial adjacency matrix is a reciprocal of the shortest path distance between the two directory nodes. The spatial

adjacency matrix is multiplied by the initial weight of the edge to obtain the spatial association intensity of the directory nodes. For example, if the initial weights of the directory A and the directory B are 10 and the shortest path distance is 2, the spatial association intensity will be $10 \times (1/2)=5$.

5 [0093] Fourthly, the access time-order information in the behavior dataset is extracted, and the access time interval between directory node pairs is computed. For example, the user accesses the directory A and then accesses the directory B, and the time interval of two accesses is 5 min. After the access time interval is normalized, the time attenuation coefficient will be obtained. For example, the time interval can be normalized by the
10 exponential attenuation function. The time attenuation coefficient is multiplied by the spatial association intensity to obtain the spatial-temporal weighted adjacency matrix. For example, if the time attenuation coefficient is 0.8 and the spatial association intensity is 5, the value of the corresponding element in the spatial-temporal weighted adjacency matrix will be $0.8 \times 5=4$.

15 [0094] Sparsification processing is carried out on the spatial-temporal weighted adjacency matrix, and the edges with weight value greater than the preset weight threshold are reserved to obtain the optimized directory access relationship graph. For example, if the weight threshold is set to be 3, only the edges with the weight value are greater than 3 will be reserved.

20 [0095] Feature extraction is carried out on the optimized directory access relationship graph by the three-layer graph convolutional network. The local access pattern features are extracted by the first graph convolutional layer, for example, which directories are accessed immediately after a certain directory is accessed frequently. The multi-hop access path features are fused by the second graph convolutional layer, for example, an access pattern
25 that a user finally reaches the target directory after accessing a plurality of intermediate directories is considered. The progressive transmission feature vectors of directory nodes are generated by the third graph convolutional layer, and the vectors represent transmission features of the directories in the access paths.

[0096] The similarity matrix among the directory nodes is computed based on the
30 generated progressive transmission feature vectors. For example, the similarity is computed by cosine similarity. Dot product operation is carried out on the similarity matrix and the original permission transmission factor to obtain the corrected permission transmission factor. For example, if the original permission transmission factor is 0.5 and the similarity is 0.8, the corrected permission transmission factor will be $0.5 \times 0.8=0.4$.

[0097] In this embodiment, the security of permission management is enhanced: the transmission of permissions can be controlled more accurately by considering the access relationship and transmission features among the directories, thus the security risks caused by excessive transmission of the permissions can be avoided; the permission transmission factor can be adjusted dynamically according to the actual access situation, which makes the permission management more flexible, and adapts to different application scenarios; and the features are extracted automatically and the permission transmission factor is corrected through the graph convolutional network, thereby reducing manual intervention, and improving the efficiency of the permission management.

[0098] In an optional embodiment, the monitoring a permission change event, extracting a permission change instruction, matching the permission change instruction with rules in the adaptive permission rule base, generating a rule matching result, computing a influence range of permission change based on the rule matching result, and forming an initial permission change set; carrying out dependency analysis on the initial permission change set, and generating a permission change dependency graph includes:

[0099] monitoring the permission change event through the permission synchronization agents, extracting an operation type, a target object and a permission attribute from the permission change event, and generating the permission change instruction; constructing a dictionary tree, inserting a rule pattern string into the dictionary tree, and constructing a failure pointer for nodes in the dictionary tree, the failure pointer pointing to the node corresponding to the longest suffix of the current node; carrying out matching in the dictionary tree along the permission change instruction, when matching fails, returning to the node corresponding to the longest suffix through the failure pointer for continuous matching till matching succeeds or returning to a root node, and generating a direct influence object set based on objects corresponding to rules which are matched successfully; recursively querying an associated rule chain based on the direct influence object set, and combining each object in the direct influence object set with the corresponding rule chain to form the initial permission change set;

[0100] carrying out dependency analysis on each object in the initial permission change set and the corresponding rule chain, starting traversing from initial rules in the rule chain, adding accessed rules into an accessed set, and recursively accessing associated rules of the current rules; if the associated rules are in the accessed set, recording a direct dependency relationship between the current rules and the associated rules; when all the associated rules of the rules are accessed, removing the current rules from the accessed set, and returning to previous rules; computing accessibility among the rules through matrix iteration, and acquiring an indirect dependency

relationship; setting permission objects in the initial permission change set as nodes of the dependency graph, and setting the direct dependency relationship and the indirect dependency relationship as edges of the dependency graph; counting the trigger times of the rules in historical records, and dividing by the total execution times to obtain a trigger probability; counting the
 5 number of the permission objects influenced by the rules and dividing by the total number of the permission objects to obtain the influence degree; and setting a product of the trigger probability and the influence degree as the dependency intensity, setting the dependency intensity as an edge weight of the dependency graph, and generating the permission change dependency graph.

[0101] In a specific embodiment, firstly, the system monitors the permission change
 10 event through the permission synchronization agents. For example, an administrator changes the access permission of the user A to a document B, and the permission synchronization agents will capture the event. The operation type (change), the target object (document B) and the permission attribute (access permission) are extracted from the event, and the permission change instruction: “Change access permission for document
 15 B” is generated.

[0102] Secondly, the system carries out rule matching by the pre-constructed adaptive permission rule base. The rule base adopts a dictionary tree structure, and the failure pointer is constructed for each node for quick matching. For example, the rule base contains rules “All users have read permission for all documents” and “User A has write
 20 permission for document B”. The permission change instruction “Change access permission for document B” is matched with the rule base. The matching process is carried out along the dictionary tree, and if the matching fails, it will returns to the node corresponding to the longest suffix through the failure pointer for continuing matching. In this embodiment, the “Change access permission for document B” is successfully matched
 25 with “User A has write permission for document B”. Based on the rule of successful matching, the direct influence object set will be generated, such as the user A and the document B in this example.

[0103] Thirdly, the system recursively queries an associated rule chain based on the direct influence object set to form the initial permission change set. For example, if there is
 30 another rule: “Users in group containing user A has read permission for document B”, and the user A belongs to a group C, the rule and the “User A has write permission for document B” form a rule chain. The direct influence objects (the user A and the document B) are combined with the corresponding rule chains to form the initial permission change set.

[0104] Fourthly, dependency analysis is carried out on the initial permission change set to generate the permission change dependency graph. Traversing is started from the initial rule in the rule chain, for example, it starts from the “User A has write permission for document B”. The accessed rules are added into the accessed set. The associated rules of the current rules are recursively accessed, for example, the “Users in group containing user A has read permission for document B”. If the associated rules are in the accessed set, the direct dependency relationship from the current rules to the associated rules is recorded. When all associated rules of the rule are accessed, the current rules are removed from the accessed set, and it is returned to the previous rule. Finally the direct dependency relationship among all the rules can be obtained by a plurality of iterations. For example, the “User A has write permission for document B” directly depends on the “Users in group containing user A has read permission for document B”.

[0105] In order to acquire the indirect dependency relationship, the system adopts a pattern similar to graph traversal, and computes the accessibility among the rules by the plurality of iterations. For example, if a rule X depends on a rule Y and the rule Y depends on a rule Z, the rule X indirectly depends on the rule Z.

[0106] Fifthly, the system sets the permission objects (the user A, the document B, and the group C) in the initial permission change set as the nodes of the dependency graph, and sets the direct dependency relationship and the indirect dependency relationship as the edges of the dependency graph. The trigger times of the rule in a historical record and the number of influenced permission objects are counted, the dependency intensity is computed and is set as the edge weight of the dependency graph. For example, if the historical trigger times of the “User A has write permission for document B” are 100 times and the total execution times of the system are 1,000 times, the trigger probability will be 0.1. If the number of permission objects influenced by the rule is 10 and the total number of permission objects of the system is 100, the influence degree will be 0.1. The dependency intensity is a product of the trigger probability and the influence degree, namely 0.01. Finally, the permission change dependency graph with nodes and edges having weights is generated.

[0107] In this embodiment, the influence range of permission change is automatically analyzed, which reduces the workload of manual analysis, thereby improving the permission management efficiency; the influence of permission change is accurately evaluated, thus potential safety risks can be avoided, and the safety of permission management is enhanced; and by constructing the permission change dependency graph,

the permission can be managed more finely, thus a more flexible permission control strategy is realized.

[0108] In an optional embodiment, the ordering permission change operations based on a topological structure of the permission change dependency graph, and generating a permission update instruction sequence; writing the permission update instruction sequence into a message queue, and generating a permission change snapshot containing an execution timestamp includes:

[0109] computing an in-degree of each node of the permission change dependency graph, and adding the nodes with the in-degrees of zero into a candidate permission node set;

[0110] computing the sum of out-edge weights of each candidate permission node in the candidate permission node set, selecting a target permission node with the maximum sum of the out-edge weights, adding the change operation corresponding to the target permission node to an permission update instruction sequence, removing out-edges of the target permission node, updating in-degree values of subsequent permission nodes adjacent to the target permission node, and removing the target permission node from the candidate permission node set; when the in-degree values of the subsequent permission nodes are zero, adding the subsequent permission nodes into the candidate permission node set; repeatedly executing the steps until the candidate permission node set is empty;

[0111] allocating the timestamp to each instruction in the permission update instruction sequence by an incremental counter; creating a permission update theme in a distributed message queue, transmitting a preparation request containing update contents to all participating nodes, and waiting for all the participating nodes to return a preparation completion response; after successful responses of all the participating nodes are received, transmitting a submission request to all the participating nodes, and writing the permission update instruction sequence in the permission update theme; and transforming a node relationship and edge weight information of the permission update instruction sequence, the timestamps and the permission change dependency graph into a binary format, and generating the permission change snapshot.

[0112] In a specific embodiment, firstly, the permission change dependence graph is constructed. The graph takes the permission nodes as vertexes and the dependence relationship between permission change operations as edges. Each vertex represents a specific permission, such as a read-write permission for a certain resource. Each edge represents the dependence relationship between two permission change operations, for example, it is needed to grant the user the read permission previous to write permission. The weight of each edge represents the intensity of dependence, for example, if the change

priorities of certain permissions are higher, the weights of the corresponding edges are higher. For example, a user needs to obtain the read permission for the document A and the document B and the write permission for the document A. If the write permission for the document A depends on the read permission for the document A, a node of “Read document A” points to a node of “Write document A” in the graph, and the weight of the edge can be set to be a higher value.

[0113] Secondly, the in-degree of each node in the graph is computed, namely the number of the edges pointing to the node. The nodes with the in-degrees of zero are added into the candidate permission node set. The nodes with the in-degrees of zero represent that the corresponding permission change operations do not have pre-dependence and can be executed preferentially. In the above example, the in-degrees of the nodes of “Read document B” and “Read document A” are zero, so the nodes will be added into the candidate permission node set.

[0114] Secondly, the target permission node with the maximum edge weight sum is selected from the candidate permission node set. It indicates that the node with the maximum subsequent operation weight sum and depending on the current node is selected to be executed preferentially. The change operation corresponding to the target permission node is added into the permission update instruction sequence. The out-edge of the target permission node is removed, the in-degree value of the subsequent permission node adjacent to the target permission node is updated, thus the target permission node is processed and is removed from the candidate permission node set. When the in-degree value of the subsequent permission node is zero, the subsequent permission node is added into the candidate permission node set. This step is repeated till the candidate permission node set is empty. If the weight of the edge of “Read document A” pointing to “Write document A” is 10 and an edge of “Read document B” does not point to other nodes. The “Read document A” is selected, and the corresponding operation of “Grant user permission for read document A” is added into the instruction sequence. Then the out-edge of the “Read document A” is removed, and the in-degree value of the “Write document A” is updated. At the moment, the in-degree of the “Write document A” is zero, and it is added into the candidate permission node set. Finally, the “Write document A” is selected, “Grant user permission for write document A” is added into the instruction sequence, then the “Read document B” is selected, and the “Grant user permission for read document A” is added into the instruction sequence.

[0115] The timestamp is allocated for each instruction in the permission update

instruction sequence by the incremental counter, so as to ensure the execution sequence of the instruction. For example, the timestamp of a first instruction is 1, and the timestamp of a second instruction is 2.

[0116] A permission update theme is created in the distributed message queue. The
 5 preparation request containing the update content is transmitted to all participating nodes, and it waits for all the participating nodes to return a preparation completion response. After receiving successful responses of all the participating nodes, the submission request is transmitted to all the participating nodes, and the permission update instruction sequence is written into the permission update theme. It ensures the data consistency of all the nodes.

10 [0117] Thirdly, the node relationship and the edge weight information of the permission update instruction sequence, the timestamps and the permission change dependency graph are transformed into the binary format, and the permission change snapshot is generated. For example, the information can be serialized into binary data by Protocol Buffers, thus facilitating storage and transmission.

15 [0118] In this embodiment, the permission change operations are sorted, which avoids unnecessary waiting and conflicts, thereby improving the efficiency of permission update; by means of the distributed message queue and the two-stage submission mechanism, the data consistency of all the participating nodes is ensured, and errors caused by data inconsistency are avoided; and the generated permission change snapshot contains
 20 complete permission change information, thereby facilitating subsequent auditing and rollback operations, and improving the reliability and security of the system.

[0119] In an optional embodiment, the reading the permission change snapshot, and extracting execution state information of the permission update instruction sequence; comparing the execution state information with the permission change dependency graph to
 25 identify exception nodes of permission update; generating a permission transmission tree based on the positions of the exception nodes in the permission change dependency graph includes:

[0120] constructing the permission change snapshot into a permission state matrix, the permission state matrix being formed by permission node rows, time-order columns and matrix
 30 elements of execution state information; carrying out difference operation on the permission state matrix and the permission change dependency graph to obtain a state deviation matrix; when a deviation value in the state deviation matrix exceeds a preset state deviation threshold, marking corresponding permission nodes as exception nodes;

[0121] constructing a forward transmission layer and a reverse transmission layer of the

permission change dependency graph based on the exception nodes, the forward transmission layer downwards progressively traversing in a layer-by-layer way through a permission inheritance relationship, and the reverse transmission layer upwards backtracks and traverses in a layer-by-layer way through a permission dependency relationship; on each of the forward
 5 transmission layer and the reverse transmission layer, computing and determining a transmission influence factor according to the state deviation value of the exception nodes, a topological distance between the nodes and a cumulative attenuation coefficient of a path; when the transmission influence factor is less than a preset truncation threshold, determining a transmission boundary of the corresponding layer, and stopping traversing in the corresponding direction; and
 10 combining the traversal results of the forward transmission layer and the reverse transmission layer, and generating the permission transmission tree.

[0122] In a specific embodiment, firstly, the permission change snapshot data is read. The snapshot records the execution state of each permission change operation, such as “Succeeded”, “Failed”, and “Partially succeeded”. If the permission change system records
 15 the access permission change condition of the users A, B, and C to resources X, Y, and Z, and the snapshot data is stored in a JSON format.

[0123] Secondly, the execution state information of the permission update instruction sequence is extracted. The execution state of each permission update operation is extracted from the permission change snapshot to form a state sequence. For example, the permission
 20 update operation state of the user A to the resource X is “Succeeded”, the permission update operation state of the user B to the resource Y is “Failed”, and the permission update operation state of the user C to the resource Z is “Succeeded”.

[0124] Thirdly, the execution state information is compared with the permission change dependency graph to identify the exception nodes of permission update. The permission change dependency graph describes the dependency relationship among
 25 different permission nodes. For example, the permission change of the user A depends on the authorization of the administrator, and the permission change of the user B depends on the authorization of the user A. The nodes with exception execution state can be identified by comparing the state sequence with the dependency graph. For example, if the
 30 permission update operation state of the user A is “Failed” and the authorization operation state of the administrator is “Succeeded”, the permission nodes of the user A are marked as the exception nodes.

[0125] Fourthly, the permission transmission tree is generated based on the positions of the exception nodes in the permission change dependency graph.

[0126] Firstly, the permission change snapshot is constructed into the permission state matrix. The rows in the matrix represent the permission nodes, the columns represent the time order, and matrix elements represent the execution state information of corresponding nodes at corresponding time points. For example, the first row in the matrix represents the user A, the first column represents a first permission change, and matrix elements (1, 1) represent the execution state of the user A in the first permission change. If the permission of the user A on the resource X is “Succeeded” in the first change and is “Failed” in a second change, the matrix elements (1, 1) are “Succeeded”, and (1, 2) are “Failed”.

[0127] Secondly, the difference operation is carried out on the permission state matrix and the permission change dependency graph to obtain the state deviation matrix. The state deviation matrix reflects the difference between the permission node state and an expected state. For example, if the expected permission state of the user A on the resource X is “Succeeded”, and the actual state is “Failed”, the corresponding element value in the state deviation matrix is “Failed” - “Succeeded” = negative value, which indicates a negative deviation.

[0128] Thirdly, when the deviation value in the state deviation matrix exceeds a preset state deviation threshold, the corresponding permission node is marked as the exception node. For example, if the state deviation threshold is set to be -0.5, all nodes with the deviation value of less than -0.5 will be marked as the exception nodes.

[0129] Fourthly, the forward transmission layer and the reverse transmission layer of the permission change dependency graph are constructed based on the exception nodes. The forward transmission layer performs downwards progressive traversing in a layer-by-layer way by the permission inheritance relationship, and the reverse transmission layer performs upwards backtracking traversal in a layer-by-layer way by the permission inheritance relationship. For example, if the permission of the user B depends on the user A, the user A is the node of the user B at the reverse transmission layer, and the user B is a node of the user A at the forward transmission layer.

[0130] In each of the forward transmission layer and the reverse transmission layer, the transmission influence factor is computed and determined according to the state deviation value of the exception node, the topological distance between nodes and the cumulative attenuation coefficient of the path. The transmission influence factor represents the influence degree of the exception node on other nodes. For example, if the user A is the exception node, and the user B depends on the user A, the transmission influence factor of the user A on the user B will be attenuated along with the increase of the topological

distance.

[0131] When the transmission influence factor is less than a preset truncation threshold, the transmission boundary of the corresponding layer is determined, and traversing in a corresponding direction is stopped. For example, if the truncation threshold is set to be 0.1, all nodes with the transmission influence factors smaller than 0.1 are considered to exceed the transmission boundary, and traversing will be stopped.

[0132] Fifthly, the traversing results of the forward transmission layer and the reverse transmission layer are combined to generate the permission transmission tree. The permission transmission tree clearly shows the influence range of the exception node.

[0133] In this embodiment, by identifying the exception node and analyzing the influence range thereof, the potential security risks can be discovered in time, and corresponding measures can be carried out, thereby preventing permission abuse and data leakage; by automatically analyzing the influence of permission change, manual intervention can be reduced, and the permission management efficiency can be improved; and by analyzing the permission transmission tree, a influence range of the permission change can be controlled more accurately, and system faults or business interruption caused by accidental permission change can be avoided.

[0134] In an optional embodiment, the computing a permission consistency score of each transmission path in the permission transmission tree; when the permission consistency score is less than a preset threshold, extracting a rule matching result corresponding to the exception nodes; generating a permission repair solution according to the rule matching result; transforming the permission repair solution into a repair instruction sequence; executing the repair instruction sequence, and updating a permission state; and writing the repaired permission state and repair process data into the adaptive permission rule base includes:

[0135] for each transmission path in the permission transmission tree, computing the permission consistency score through the permission numerical value difference between the transmission influence factor on the transmission path and an adjacent permission node; when the permission consistency score is less than a preset score threshold, extracting a rule matching result corresponding to the exception node;

[0136] acquiring a rule weight value and a rule influence direction of each rule from the rule matching result, computing a correction direction of a current permission value and a target permission value of the exception node, sorting by descending according to the rule weight value, and selecting the rule with the maximum rule weight value and the rule influence direction the

same as the correction direction as a repair rule; applying the repair rule to the exception node, and determining a rule repair parameter based on a state deviation value of the exception node; generating the repair instruction sequence according to the repair rule and the repair parameter;

[0137] executing the repair instruction sequence in a preset time window, and recording
5 execution start time, execution completion time, an execution state code and a state variation after execution of each repair instruction; when detecting an exception execution state code or that the state variation does not meet a preset variation threshold, trigger the corresponding repair instruction to be executed again;

[0138] computing state deviation improvement degrees before and after execution of the
10 repair instruction sequence, a state fluctuation condition in the execution process and an execution success rate of the repair instruction; and writing the state deviation improvement degrees, the state fluctuation condition, the execution success rate and the permission state after repair into the adaptive permission rule base.

[0139] In a specific embodiment, firstly, the permission transmission tree is
15 constructed. The permission transmission tree reflects the inheritance and transfer relationship of the permission in the system. The root node of the tree represents the highest permission, and child nodes inherit the permission of the father node and can transfer the permission downwards. Each node contains a permission value which represents the permission level of the node. For example, if the permission value of the root
20 node is 10, the permission value of the child nodes may be 8, 7 and the like, and the lower the value is, the lower the permission level is.

[0140] Secondly, the permission consistency score is computed. Each transmission
path in the permission transmission tree is traversed, and the permission consistency score is computed according to the permission value difference between the adjacent permission
25 nodes on the path and the predefined transmission influence factor. The transmission influence factor is used for measuring an acceptable range of the permission value difference between the adjacent nodes. For example, if the permission value of the father node is 8, the permission value of the child nodes is 5, and the transmission influence factor is 0.2, the permission consistency score of the path segment will be high; and if the
30 permission value of the child nodes is 1, the score will be low, which indicates that there may be permission exception.

[0141] Thirdly, the exception nodes and the rule matching result are extracted. When the permission consistency score of a certain transmission path is less than the preset threshold, it is considered that there is permission exception in some nodes on the path. The

exception nodes are extracted, and rule matching is carried out according to the adaptive permission rule base. The rule base contains a series of predefined permission repair rules, and each rule corresponds to a specific permission abnormality scenario. For example, the rule base may contain rule of “Permission of child nodes must not be less than half of father node” and the like.

[0142] An optimal repair rule is selected. A plurality of repair rules may be matched for each exception node. The optimal repair rule is selected according to the weight value and the rule influence direction of each rule. The rule weight value represents the importance of the rule, and the rule influence direction represents a correction direction (increase or decrease) of the rule to the permission value. For example, if the current permission value of the exception node is 1 and the target permission value is 5, the rule that the rule influence direction is “Increase”. The rule with the maximum weight value is selected as the final repair rule in a plurality of candidate rules.

[0143] The repair instruction sequence is generated and executed. Rule repair parameters are determined according to the selected repair rule and the state deviation value (a difference value between the current permission value and the target permission value) of the exception node, and the repair instruction sequence is generated. The repair instruction sequence contains a series of operation instructions for changing the permission value of the exception node. For example, if the repair rule is “Set permission of child nodes to be half of parent node”, and the permission value of the parent node is 8, the generated repair instruction sequence is “Set permission of child nodes to be 4”. The repair instruction sequence is executed in the preset time window, and relevant data in the execution process is recorded, such as an execution state code and a state variation. If an execution exception is detected, a retry mechanism will be triggered.

[0144] The adaptive permission rule base is updated. The adaptive permission rule base is updated according to the execution result of the repair instruction sequence. Data such as the repaired permission state, the state deviation relieving degree, a state fluctuation situation and a repair instruction execution success rate will be written into the rule base for reference and optimization of subsequent permission repair. For example, if the execution success rate of a certain repair rule is always low, the weight value of the rule can be adjusted or the rule can be changed.

[0145] As an example, if there is an exception node in the permission transmission tree, its current permission value is 1, and its target permission value is 5. According to the rule base matching result, there are two candidate rules: a rule A: the permission value is

increased by 2, the weight value is 0.8; and a rule B: the permission value is increased by 1, the weight value is 0.5. Since the weight value of the rule A is higher and the rule influence direction is the same as the correction direction, the rule A is selected as the repair rule. The rule repair parameter is determined to be 2 according to the state deviation value of 4. The generated repair instruction sequence is “Increase permission value by 2”. After executing the instruction, the permission value of the exception node is 3. The repaired permission value, the state deviation improvement degree and other data are written in the adaptive permission rule base for subsequent permission repair.

[0146] In this embodiment, the permission exception can be automatically identified and repaired without manual intervention, thereby improving the permission management efficiency; through permission consistency score and the adaptive rule base, the permission configuration is ensured to conform to predefined rules and strategies, and the situation of too high or too low permission is avoided; and the permission exception is repaired in time, which can effectively prevent unauthorized access and data leakage, thereby improving the security of the system.

[0147] FIG. 2 is a flowchart of a permission synchronization system for distributed storage and a data warehouse according to an embodiment of the present invention, as shown in FIG. 2, the system includes:

[0148] a first unit, configured to acquire user operation data in a distributed storage system, generate a behavior dataset, carry out feature extraction on the behavior dataset, and compute to obtain a user access pattern feature matrix; input the user access pattern feature matrix into a pre-trained machine learning model, compute a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and generate an intelligent directory permission mapping table; read structure information of a data table in the data warehouse, compute a value score, a field association intensity and a cross-table access influence degree of the data table, and generate a multi-dimensional table permission evaluation model; and carry out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, and generate an adaptive permission rule base;

[0149] a second unit, configured to deploy permission synchronization agents on physical nodes in distributed storage, and allocate the adaptive permission rule base to each permission synchronization agent according to a consistency Hash algorithm; monitor a permission change event, extracting a permission change instruction, match the permission change instruction with rules in the adaptive permission rule base, generate a rule matching

result, compute a influence range of permission change based on the rule matching result, and form an initial permission change set; carry out dependency analysis on the initial permission change set, and generate a permission change dependency graph; order permission change operations based on a topological structure of the permission change dependency graph, and generate a permission update instruction sequence; write the permission update instruction sequence into a message queue, and generate a permission change snapshot containing an execution timestamp; and

[0150] a third unit, configured to read the permission change snapshot, and extract execution state information of the permission update instruction sequence; compare the execution state information with the permission change dependency graph to identify exception nodes of permission update; generate a permission transmission tree based on the positions of the exception nodes in the permission change dependency graph; compute a permission consistency score of each transmission path in the permission transmission tree; when the permission consistency score is less than a preset threshold, extract a rule matching result corresponding to the exception nodes; generate a permission repair solution according to the rule matching result; transform the permission repair solution into a repair instruction sequence; execute the repair instruction sequence, and update a permission state; and write the repaired permission state and repair process data into the adaptive permission rule base.

[0151] In a third aspect,

[0152] an electronic device provided by the embodiment of the present invention includes:

[0153] a processor; and

[0154] a memory configured to store processor executable instructions;

[0155] the processor is configured to call instructions stored in the memory to execute the abovementioned method.

[0156] In a fourth aspect,

[0157] the embodiment of the present invention provides a computer-readable storage medium having computer program instructions stored therein, when executed by a processor, the computer program instructions execute the abovementioned method.

[0158] The present invention can be a method, an apparatus, a system, and/or a computer program product. The computer program product can include a computer-readable storage medium that contains computer-readable program instructions for executing the present invention in various aspects. Finally, it is to be noted that: the above embodiments are only used for illustrating the technical solution of the present

invention, not to limit it. Although the present invention is described in detail with reference to the aforementioned embodiments, those of ordinary skill in the art should understand that it may still modify the technical solutions recorded in the aforementioned embodiments, or replace some or all of the technical features thereof. These modifications
5 or substitutions do not make the essence of the corresponding technical solution out of the scope of the technical solutions of each embodiment of the present invention.

CLAIMS

We claim:

1. A permission synchronization method for distributed storage and a data warehouse, comprising:

5 acquiring user operation data in a distributed storage system, generating a behavior dataset, carrying out feature extraction on the behavior dataset, and computing to obtain a user access pattern feature matrix; inputting the user access pattern feature matrix into a pre-trained machine learning model, computing a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and generating an intelligent
10 directory permission mapping table; reading structure information of a data table in the data warehouse, computing a value score, a field association intensity and a cross-table access influence degree of the data table, and generating a multi-dimensional table permission evaluation model; and carrying out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation
15 model, and generating an adaptive permission rule base;

 deploying permission synchronization agents on physical nodes in distributed storage, and allocating the adaptive permission rule base to each permission synchronization agent according to a consistency Hash algorithm; monitoring a permission change event, extracting a permission change instruction, matching the permission change instruction
20 with rules in the adaptive permission rule base, generating a rule matching result, computing a influence range of permission change based on the rule matching result, and forming an initial permission change set; carrying out dependency analysis on the initial permission change set, and generating a permission change dependency graph; ordering permission change operations based on a topological structure of the permission change
25 dependency graph, and generating a permission update instruction sequence; writing the permission update instruction sequence into a message queue, and generating a permission change snapshot containing an execution timestamp;

 reading the permission change snapshot, and extracting execution state information of the permission update instruction sequence; comparing the execution state information with
30 the permission change dependency graph to identify exception nodes of permission update; generating a permission transmission tree based on the positions of the exception nodes in the permission change dependency graph; computing a permission consistency score of each transmission path in the permission transmission tree; when the permission consistency score is less than a preset threshold, extracting a rule matching result

corresponding to the exception nodes; generating a permission repair solution according to the rule matching result; transforming the permission repair solution into a repair instruction sequence; executing the repair instruction sequence, and updating a permission state; and writing the repaired permission state and repair process data into the adaptive permission rule base.

2. The method according to claim 1, wherein the acquiring user operation data in a distributed storage system, generating a behavior dataset, carrying out feature extraction on the behavior dataset, and computing to obtain a user access pattern feature matrix; inputting the user access pattern feature matrix into a pre-trained machine learning model, computing a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and generating an intelligent directory permission mapping table; reading structure information of a data table in the data warehouse, computing a value score, a field association intensity and a cross-table access influence degree of the data table, and generating a multi-dimensional table permission evaluation model; and carrying out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, and generating an adaptive permission rule base comprises:

acquiring user access time, operation type, and access path data in an audit log in the distributed storage system, and pre-processing the user access time, operation type, and access path data by a sliding time window method, and generating the behavior dataset; extracting time-order features, operation features and path features based on the behavior dataset, vectorizing the time-order features, operation features and path feature, and generating the user access pattern feature matrix;

inputting the user access pattern feature matrix into the pre-trained machine learning model, carrying out product accumulation computation on a time attenuation factor and access times to obtain the directory access frequency weight, computing to obtain the time sensitivity coefficient through an exponential attenuation function, and computing the permission transmission factor through an intersection-over-union of a directory permission set; constructing a directory access relationship graph, setting the directory as a graph node, setting a user access path as a graph edge, extracting progressive transmission features between directories through a graph convolution network based on the directory access relationship graph, and correcting the permission transmission factor based on the progressive transmission features; generating the intelligent directory permission mapping table based on the corrected permission transmission factor, the directory access frequency

weight and the time sensitivity coefficient;

reading the structure information of the data table in the data warehouse, computing to obtain the sensitivity score, a reference frequency and a data volume score based on the structure information, and carrying out weighted computation on the sensitivity score, the reference frequency and the data volume score to obtain a data table value score; computing to obtain a field association coefficient and a field importance score based on the structure information, and carrying out weighted computation on the field association degree coefficient and the field importance score to obtain field association intensity; computing to obtain an inter-table access frequency and a degree of dependence based on the structure information, carrying out product accumulation computation on the inter-table access frequency and the degree of dependence to obtain the cross-table access influence degree, and generating the multi-dimensional table permission evaluation model based on the data table value score, the field association intensity and the cross-table access influence degree; and

carrying out weighted computation on the data table value score, the field association intensity and the cross-table access influence degree in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, generating a permission rule, and storing the permission rule into the adaptive permission rule base.

3. The method according to claim 2, wherein the constructing a directory access relationship graph, setting the directory as a graph node, setting a user access path as a graph edge, extracting progressive transmission features between directories through a graph convolution network based on the directory access relationship graph, and correcting the permission transmission factor based on the progressive transmission features comprises:

constructing an initial directory graph based on the access path data in the behavior dataset, and setting a direct access frequency between directory nodes as an initial weight of an edge;

computing the shortest path distance between any two directory nodes in the initial directory graph, constructing a spatial adjacency matrix based on the shortest path distance, and multiplying the spatial adjacency matrix by the initial weight of the edge to obtain the spatial association intensity of the directory nodes;

extracting access time-order information in the behavior dataset, computing an access time interval between directory node pairs, normalizing the access time interval to obtain the time attenuation coefficient, and multiplying the time attenuation coefficient by the

spatial association intensity to obtain a spatial-temporal weighted adjacency matrix;

carrying out sparsification processing on the spatial-temporal weighted adjacency matrix, and reserving the edge with a weight value greater than a preset weight threshold, and obtaining an optimized directory access relationship graph;

- 5 carrying out feature extraction on the optimized directory access relationship graph by a multilayered graph convolution network, extracting local access pattern features by a first graph convolution layer, fusing multi-hop access path features by a second graph convolution layer, and generating progressive transmission feature vectors of the directory nodes by a third graph convolution layer; and
- 10 computing a similarity matrix between the directory nodes based on the progressive transmission feature vectors, and carrying out dot product operation on the similarity matrix and an original permission transmission factor to obtain the corrected permission transmission factor.

4. The method according to claim 1, wherein the monitoring a permission change event, extracting a permission change instruction, matching the permission change instruction with rules in the adaptive permission rule base, generating a rule matching result, computing a influence range of permission change based on the rule matching result, and forming an initial permission change set; carrying out dependency analysis on the initial permission change set, and generating a permission change dependency graph
- 20 comprises:

- monitoring the permission change event through the permission synchronization agents, extracting an operation type, a target object and a permission attribute from the permission change event, and generating the permission change instruction; constructing a dictionary tree, inserting a rule pattern string into the dictionary tree, and constructing a
- 25 failure pointer for nodes in the dictionary tree, the failure pointer pointing to the node corresponding to the longest suffix of the current node; carrying out matching in the dictionary tree along the permission change instruction, when matching fails, returning to the node corresponding to the longest suffix through the failure pointer for continuous matching till matching succeeds or returning to a root node, and generating a direct
- 30 influence object set based on objects corresponding to rules which are matched successfully; recursively querying an associated rule chain based on the direct influence object set, and combining each object in the direct influence object set with the corresponding rule chain to form the initial permission change set;

carrying out dependency analysis on each object in the initial permission change set

and the corresponding rule chain, starting traversing from initial rules in the rule chain, adding accessed rules into an accessed set, and recursively accessing associated rules of the current rules; if the associated rules are in the accessed set, recording a direct dependency relationship between the current rules and the associated rules; when all the associated rules of the rules are accessed, removing the current rules from the accessed set, and returning to previous rules; computing accessibility among the rules through matrix iteration, and acquiring an indirect dependency relationship; setting permission objects in the initial permission change set as nodes of the dependency graph, and setting the direct dependency relationship and the indirect dependency relationship as edges of the dependency graph; counting the trigger times of the rules in historical records, and dividing by the total execution times to obtain a trigger probability; counting the number of the permission objects influenced by the rules and dividing by the total number of the permission objects to obtain the influence degree; and setting a product of the trigger probability and the influence degree as the dependency intensity, setting the dependency intensity as an edge weight of the dependency graph, and generating the permission change dependency graph.

5. The method according to claim 4, wherein the ordering permission change operations based on a topological structure of the permission change dependency graph, and generating a permission update instruction sequence; writing the permission update instruction sequence into a message queue, and generating a permission change snapshot containing an execution timestamp comprises:

computing an in-degree of each node of the permission change dependency graph, and adding the nodes with the in-degrees of zero into a candidate permission node set;

computing the sum of out-edge weights of each candidate permission node in the candidate permission node set, selecting a target permission node with the maximum sum of the out-edge weights, adding the change operation corresponding to the target permission node to an permission update instruction sequence, removing out-edges of the target permission node, updating in-degree values of subsequent permission nodes adjacent to the target permission node, and removing the target permission node from the candidate permission node set; when the in-degree values of the subsequent permission nodes are zero, adding the subsequent permission nodes into the candidate permission node set; repeatedly executing the steps until the candidate permission node set is empty;

allocating the timestamp to each instruction in the permission update instruction sequence by an incremental counter; creating a permission update theme in a distributed message queue, transmitting a preparation request containing update contents to all

participating nodes, and waiting for all the participating nodes to return a preparation completion response; after successful responses of all the participating nodes are received, transmitting a submission request to all the participating nodes, and writing the permission update instruction sequence in the permission update theme; and transforming a node
 5 relationship and edge weight information of the permission update instruction sequence, the timestamps and the permission change dependency graph into a binary format, and generating the permission change snapshot.

6. The method according to claim 1, wherein the reading the permission change snapshot, and extracting execution state information of the permission update instruction
 10 sequence; comparing the execution state information with the permission change dependency graph to identify exception nodes of permission update; generating a permission transmission tree based on the positions of the exception nodes in the permission change dependency graph comprises:

constructing the permission change snapshot into a permission state matrix, the
 15 permission state matrix being formed by permission node rows, time-order columns and matrix elements of execution state information; carrying out difference operation on the permission state matrix and the permission change dependency graph to obtain a state deviation matrix; when a deviation value in the state deviation matrix exceeds a preset state deviation threshold, marking corresponding permission nodes as exception nodes;

20 constructing a forward transmission layer and a reverse transmission layer of the permission change dependency graph based on the exception nodes, the forward transmission layer downwards progressively traversing in a layer-by-layer way through a permission inheritance relationship, and the reverse transmission layer upwards backtracks and traverses in a layer-by-layer way through a permission dependency relationship; on
 25 each of the forward transmission layer and the reverse transmission layer, computing and determining a transmission influence factor according to the state deviation value of the exception nodes, a topological distance between the nodes and a cumulative attenuation coefficient of a path; when the transmission influence factor is less than a preset truncation threshold, determining a transmission boundary of the corresponding layer, and stopping
 30 traversing in the corresponding direction; and combining the traversal results of the forward transmission layer and the reverse transmission layer, and generating the permission transmission tree.

7. The method according to claim 6, wherein the computing a permission consistency score of each transmission path in the permission transmission tree; when the permission

consistency score is less than a preset threshold, extracting a rule matching result corresponding to the exception nodes; generating a permission repair solution according to the rule matching result; transforming the permission repair solution into a repair instruction sequence; executing the repair instruction sequence, and updating a permission state; and writing the repaired permission state and repair process data into the adaptive permission rule base comprises:

for each transmission path in the permission transmission tree, computing the permission consistency score through the permission numerical value difference between the transmission influence factor on the transmission path and an adjacent permission node; when the permission consistency score is less than a preset score threshold, extracting a rule matching result corresponding to the exception node;

acquiring a rule weight value and a rule influence direction of each rule from the rule matching result, computing a correction direction of a current permission value and a target permission value of the exception node, sorting by descending according to the rule weight value, and selecting the rule with the maximum rule weight value and the rule influence direction the same as the correction direction as a repair rule; applying the repair rule to the exception node, and determining a rule repair parameter based on a state deviation value of the exception node; generating the repair instruction sequence according to the repair rule and the repair parameter;

executing the repair instruction sequence in a preset time window, and recording execution start time, execution completion time, an execution state code and a state variation after execution of each repair instruction; when detecting an exception execution state code or that the state variation does not meet a preset variation threshold, trigger the corresponding repair instruction to be executed again;

computing state deviation improvement degrees before and after execution of the repair instruction sequence, a state fluctuation condition in the execution process and an execution success rate of the repair instruction; and writing the state deviation improvement degrees, the state fluctuation condition, the execution success rate and the permission state after repair into the adaptive permission rule base.

8. A permission synchronization system for distributed storage and a data warehouse, configured to implement the method according to any one of claims 1 to 7, comprising:

a first unit, configured to acquire user operation data in a distributed storage system, generate a behavior dataset, carry out feature extraction on the behavior dataset, and compute to obtain a user access pattern feature matrix; input the user access pattern feature

matrix into a pre-trained machine learning model, compute a directory access frequency weight, a time sensitivity coefficient and a permission transmission factor, and generate an intelligent directory permission mapping table; read structure information of a data table in the data warehouse, compute a value score, a field association intensity and a cross-table
 5 access influence degree of the data table, and generate a multi-dimensional table permission evaluation model; and carry out weighted computation on parameters in the intelligent directory permission mapping table and the multi-dimensional table permission evaluation model, and generate an adaptive permission rule base;

a second unit, configured to deploy permission synchronization agents on physical
 10 nodes in distributed storage, and allocate the adaptive permission rule base to each permission synchronization agent according to a consistency Hash algorithm; monitor a permission change event, extracting a permission change instruction, match the permission change instruction with rules in the adaptive permission rule base, generate a rule matching result, compute a influence range of permission change based on the rule matching result,
 15 and form an initial permission change set; carry out dependency analysis on the initial permission change set, and generate a permission change dependency graph; order permission change operations based on a topological structure of the permission change dependency graph, and generate a permission update instruction sequence; write the permission update instruction sequence into a message queue, and generate a permission
 20 change snapshot containing an execution timestamp; and

a third unit, configured to read the permission change snapshot, and extract execution state information of the permission update instruction sequence; compare the execution state information with the permission change dependency graph to identify exception nodes of permission update; generate a permission transmission tree based on the positions of the
 25 exception nodes in the permission change dependency graph; compute a permission consistency score of each transmission path in the permission transmission tree; when the permission consistency score is less than a preset threshold, extract a rule matching result corresponding to the exception nodes; generate a permission repair solution according to the rule matching result; transform the permission repair solution into a repair instruction
 30 sequence; execute the repair instruction sequence, and update a permission state; and write the repaired permission state and repair process data into the adaptive permission rule base.

9. An electronic device, comprising: a processor; and

a memory configured to store processor executable instructions;

the processor is configured to call instructions stored in the memory to execute the

method according to any one of claims 1 to 7.

10. A computer-readable storage medium having computer program instructions stored therein, wherein when executed by a processor, the computer program instructions execute the method according to any one of claims 1 to 7.