

LOGGING METHOD AND APPARATUS, ELECTRONIC DEVICE, AND COMPUTER-READABLE STORAGE MEDIUM

TECHNICAL FIELD

[0001] The disclosure relates to intelligent identification technologies, and particularly to a logging method and apparatus, an electronic device, and a computer-readable storage medium.

BACKGROUND

[0002] In today's computer software industry, a piece of software with complete permissions needs to be able to record operational behaviors of all users of a system in the system in form of an operation log and provide a retrieval function for statistics, auditing, tracking, tracing, and other needs.

[0003] Present software architecture designs generally adopt a form that a front end and a back end are separated, namely divided into two projects, i.e., a client interface style and a server Application Programming Interface (API). Under a normal condition, a server provides a Hypertext Transfer Protocol (HTTP) or Remote Procedure Call (RPC) API, and a client calls the back-end API through a Transmission Control Protocol (TCP) to achieve a communication purpose.

[0004] In the related art, a commonly used auditing logging manner is usually that a server responds to an API calling request in a process of recording data of the API calling request. However, when this manner is adopted, an Input/Output (I/O) operation over log data may slow down the response of the API. That is, the log data recording manner in the related art may affect the response speed to API calling requests.

SUMMARY

[0005] Embodiments of the disclosure provide a logging method and apparatus, an electronic device, and a computer-readable storage medium, which may increase the response speed to an API calling request.

[0006] The technical solutions of the embodiments of the disclosure are implemented as follows.

[0007] The embodiments of the disclosure provide a logging method, which may include that: an API calling request sent by a client is intercepted using a first asynchronous thread; log data is generated based on the API calling request, and the log data is stored in a log storage queue, the log storage queue being configured to store log data to be filtered for storage in a database; and real-time recording of a scene state is implemented using a service thread in response to a modification request in the API calling request.

[0008] In the embodiment, since the first asynchronous thread is used to intercept the API calling request sent by the client and including the modification request configured to trigger a modification operation over a working parameter, and when the service thread different from the first asynchronous thread is used to modify the working parameter according to the modification request sent by the client and record the scene state using the modified working parameter, the first asynchronous thread is also used to generate the log data according to the API calling request and store the generated log data, generation and storage of the log data may be implemented asynchronously at the same time of recording the scene state to eliminate influences between responding to the API calling request and recording of the log data, further increase the response speed to the API calling request in the process of recording the log data, and reduce recording of unnecessary log data. Therefore, the recording efficiency of the log data is improved, and the redundancy of the recorded log data is reduced.

[0009] In an embodiment, the following operations may further be included: the log data in the log storage queue is filtered based on a preset first filtering parameter, and filtered log data is stored in the database in batches.

[0010] In the embodiment, through the first filtering parameter, the redundancy of the

log data in the database may be reduced effectively, not only may the query efficiency of the log data be improved when a user queries the log data in the database, but also occupation of a disk may be reduced. In addition, the filtered log data is stored in the database in a batch processing manner, so that I/O operations over the database may be reduced, and the log storage efficiency may be improved.

[0011] In an embodiment, the following operations may further be included: user permission information is obtained using the first asynchronous thread according to a request header obtained by parsing the API calling request, and the user permission information is authenticated to obtain an authentication result.

[0012] The operation that the log data is generated based on the API calling request may include that: when the authentication result represents successful authentication, the log data is generated using the first asynchronous thread based on the API calling request.

[0013] In the embodiment, authentication is performed for the API calling request, and under the condition of successful authentication, the corresponding log data is generated according to the API calling request, so that the condition that the client makes a request of faking an operating user or user behavior data may be found timely, and the generated log data is more real.

[0014] In an embodiment, the operation that the log data is generated based on the API calling request may include that: first data is extracted from a request header obtained by parsing each calling request in the API calling request, the first data at least including user identity data and resource path data; second data is extracted from a request body and request line obtained by parsing each calling request, the second data at least including request method data and request content data; and the first data and the second data are combined to obtain a piece of log data corresponding to each calling request.

[0015] In the embodiment, data is extracted from a corresponding API calling request, and a piece of log data is generated according to the extracted data, so that conveniences are brought to subsequent filtering and recording of the log data.

[0016] In an embodiment, at least one piece of log data may be stored in the log

storage queue, and each piece of log data may include a log parameter; and the operation that the log data in the log storage queue is filtered based on the preset first filtering parameter and the filtered log data is stored in the database in batches may include that: the at least one piece of log data is acquired from the log storage queue, the log parameter of each piece of log data in the at least one piece of log data is matched with the first filtering parameter to obtain a matched matching parameter, the filtered log data is determined according to at least one piece of target log data corresponding to the matching parameter, and when the amount of the filtered log data reaches a preset amount, the filtered log data is stored in the database.

[0017] In the embodiment, a one-by-one matching method is adopted, so that the filtering accuracy of the log data may be improved, and furthermore, the redundancy of the log data stored in the database is reduced.

[0018] In an embodiment, the operation that the at least one piece of log data is acquired from the log storage queue may include that: the log data is read from the log storage queue according to a preset frequency; and when present log data is read, a next piece of log data is continued to be read from the log storage queue until a last piece of log data stored in the log storage queue is read, thereby acquiring the at least one piece of log data.

[0019] In the embodiment, in the abovementioned acquisition manner, the acquisition efficiency of the log data may be improved.

[0020] In an embodiment, the matching parameter may include at least one of a method parameter, a resource path resource, a resource path prefix parameter, a non-target parameter in a parameter blacklist, or a target parameter in a parameter whitelist.

[0021] In the embodiment, the diversity of the filtering manner may be improved, and the data recording flexibility is improved.

[0022] In an embodiment, the operation that the filtered log data is determined according to the at least one piece of target log data corresponding to the matching parameter may include that: candidate log data is determined according to the at least one piece of target log data corresponding to the matching parameter; and the filtered log data

is determined based on the candidate log data.

[0023] In the embodiment, the filtered log data is determined according to the candidate log data, so that the obtained filtered log data is more accurate.

[0024] In an embodiment, the operation that the candidate log data is determined according to the at least one piece of target log data corresponding to the matching parameter may include that: when the matching parameter is the method parameter, at least one piece of target log data corresponding to the method parameter in the at least one piece of log data is determined as the candidate log data; when the matching parameter is the resource path parameter, at least one piece of target log data corresponding to the resource path parameter is determined as the candidate log data; when the matching parameter is the resource path prefix parameter, at least one piece of target log data corresponding to the resource path prefix parameter in the at least one piece of log data is determined as the candidate log data; when the matching parameter is the at least one non-target parameter in the parameter blacklist, log data, except at least one piece of target log data corresponding to the at least one non-target parameter, in the at least one piece of log data is determined as the candidate log data; and when the matching parameter is the at least one target parameter in the parameter whitelist, at least one piece of target log data corresponding to the at least one target parameter in the at least one piece of log data is determined as the candidate log data.

[0025] In the embodiment, in the abovementioned filtering manner, multiple types of filtered log data may be obtained, and the data recording flexibility is improved.

[0026] In an embodiment, the method may further include that: a filtering parameter setting instruction containing the preset first filtering parameter is received; and a preset first initial filtering parameter is modified to the preset first filtering parameter in response to the filtering parameter setting instruction.

[0027] In the embodiment, the preset first initial filtering parameter is modified to the preset first filtering parameter according to the received filtering parameter setting instruction to subsequently filter the log data, so that the diversity of the filtering manner may be improved, and the data recording flexibility is improved.

[0028] In an embodiment, the operation that real-time recording of the scene state is implemented using the service thread in response to the modification request in the API calling request may include that: a working parameter of the service thread is modified using the service thread in response to the modification request in the API calling request to obtain a modified working parameter; a real-time state of a scene is determined based on the modified working parameter and received real-time scene data acquired by an acquisition device; and the real-time state is compared with a preset state to obtain a comparison result, and the comparison result is recorded in real time.

[0029] In the embodiment, real-time recording of the scene state is implemented based on the modified working parameter, so that the recording accuracy of the scene state may be improved.

[0030] In an embodiment, the method may further include that: when the comparison result represents that the real-time state is abnormal, prompt information is output.

[0031] In the embodiment, in the abovementioned manner, a related maintainer may timely intervene in a device related to the scene state, for example, the acquisition device, thereby ensuring that the device may run normally.

[0032] The embodiments of the disclosure provide a logging apparatus, which may include an interceptor, configured to intercept an API calling request sent by a client, generate log data based on the API calling request, and store the log data in a log storage queue, the log storage queue being configured to store log data to be filtered for storage in a database; and a service thread, configured to implement real-time recording of a scene state in response to a modification request in the API calling request.

[0033] In an embodiment, the interceptor is further configured to filter the log data in the log storage queue based on a preset first filtering parameter and store filtered log data in the database in batches.

[0034] In an embodiment, the interceptor is further configured to obtain user permission information according to a request header obtained by parsing the API calling request, authenticate the user permission information to obtain an authentication result, and when the authentication result represents successful authentication, generate the log

data based on the API calling request.

[0035] In an embodiment, the interceptor is further configured to extract first data from a request header obtained by parsing each calling request in the API calling request, the first data at least including user identity data and resource path data, extract second data from a request body and request line obtained by parsing each calling request, the second data at least including request method data and request content data, and combine the first data and the second data to obtain a piece of log data corresponding to each calling request.

[0036] In an embodiment, at least one piece of log data is stored in the log storage queue, and each piece of log data includes a log parameter; and the interceptor is further configured to acquire the at least one piece of log data from the log storage queue, match the log parameter of each piece of log data in the at least one piece of log data and the first filtering parameter to obtain a matched matching parameter, determine the filtered log data according to at least one piece of target log data corresponding to the matching parameter, and when the amount of the filtered log data reaches a preset amount, store the filtered log data in the database.

[0037] In an embodiment, the interceptor is further configured to read the log data from the log storage queue according to a preset frequency, and when present log data is read, continue to read a next piece of log data from the log storage queue until a last piece of log data stored in the log storage queue is read, thereby acquiring the at least one piece of log data.

[0038] In an embodiment, the matching parameter includes at least one of a method parameter, a resource path resource, a resource path prefix parameter, a non-target parameter in a parameter blacklist, or a target parameter in a parameter whitelist.

[0039] In an embodiment, the interceptor is further configured to determine candidate log data according to the at least one piece of target log data corresponding to the matching parameter and determine the filtered log data based on the candidate log data.

[0040] In an embodiment, the interceptor is further configured to: when the matching parameter is the method parameter, determine at least one piece of target log data

corresponding to the method parameter in the at least one piece of log data as the candidate log data; when the matching parameter is the resource path parameter, determine at least one piece of target log data corresponding to the resource path parameter as the candidate log data; when the matching parameter is the resource path prefix parameter, determine at least one piece of target log data corresponding to the resource path prefix parameter in the at least one piece of log data as the candidate log data; when the matching parameter is the at least one non-target parameter in the parameter blacklist, determine log data, except at least one piece of target log data corresponding to the at least one non-target parameter, in the at least one piece of log data as the candidate log data; and when the matching parameter is the at least one target parameter in the parameter whitelist, determine at least one piece of target log data corresponding to the at least one target parameter in the at least one piece of log data as the candidate log data.

[0041] In an embodiment, the apparatus further includes a modification module, configured to receive a filtering parameter setting instruction containing the preset first filtering parameter and modify a preset first initial filtering parameter to the preset first filtering parameter in response to the filtering parameter setting instruction.

[0042] In an embodiment, the service thread is further configured to modify an own working parameter in response to the modification request in the API calling request to obtain a modified working parameter, determine a real-time state of a scene based on the modified working parameter and received real-time scene data acquired by an acquisition device, compare the real-time state and a preset state to obtain a comparison result, and record the comparison result in real time.

[0043] In an embodiment, the service thread is further configured to, when the comparison result represents that the real-time state is abnormal, output prompt information.

[0044] The embodiments of the disclosure provide an electronic device, which may include a memory and a processor.

[0045] The memory may be configured to store an executable computer program.

[0046] The processor may be configured to execute the executable computer program stored in the memory to implement the logging method.

[0047] The embodiments of the disclosure provide a computer-readable storage medium, which may store a computer program, configured to be executed by a processor to implement the logging method.

BRIEF DESCRIPTION OF THE DRAWINGS

[0048] FIG. 1 is an optional architecture diagram of an exemplary logging system according to an embodiment of the disclosure.

[0049] FIG. 2 is an optional flowchart of an exemplary logging method according to an embodiment of the disclosure.

[0050] FIG. 3 is an optional flowchart of an exemplary logging method according to an embodiment of the disclosure.

[0051] FIG. 4 is an optional flowchart of an exemplary logging method according to an embodiment of the disclosure.

[0052] FIG. 5 is a flowchart of an exemplary method for generating log data and responding to an API calling request according to an embodiment of the disclosure.

[0053] FIG. 6 is an optional flowchart of an exemplary logging method according to an embodiment of the disclosure.

[0054] FIG. 7 is an optional flowchart of an exemplary logging method according to an embodiment of the disclosure.

[0055] FIG. 8 is a flowchart of an exemplary method for filtering log data and cleaning log data stored in a database according to an embodiment of the disclosure.

[0056] FIG. 9 is a structure composition diagram of a logging apparatus according to an embodiment of the disclosure.

[0057] FIG. 10 is a structure composition diagram of an electronic device according to an embodiment of the disclosure.

DETAILED DESCRIPTION

[0058] For making the objectives, technical solutions, and advantages of the disclosure clearer, the disclosure will further be described below in combination with the drawings in detail. The described embodiments should not be considered as limits to the disclosure. All other embodiments obtained by those of ordinary skill in the art without creative work shall fall within the scope of protection of the disclosure.

[0059] "Some embodiments" involved in the following descriptions describes a subset of all possible embodiments. However, it can be understood that "some embodiments" may be the same subset or different subsets of all the possible embodiments, and may be combined without conflicts.

[0060] Term "first/second/third" involved in the following descriptions is only for distinguishing similar objects and does not represent a specific sequence of the objects. It can be understood that "first/second/third" may be interchanged to specific sequences or orders if allowed to implement the embodiments of the disclosure described herein in sequences except the illustrated or described ones.

[0061] Unless otherwise defined, all technological and scientific terms used in the disclosure have meanings the same as those usually understood by those skilled in the art of the disclosure. The terms used in the disclosure are only adopted to describe the embodiments of the disclosure and not intended to limit the disclosure.

[0062] Before the embodiments of the disclosure are further described in detail, nouns and terms involved in the embodiments of the disclosure will be described. The nouns and terms involved in the embodiments of the disclosure are suitable for the following explanations.

[0063] In today's computer software industry, a piece of software with complete

permissions needs to be able to record operational behaviors of all users of a system in the system in form of an operation log and provide a retrieval function for statistics, auditing, tracking, tracing, and other needs. Present software architecture designs generally adopt a form that a front end and a back end are separated, namely divided into two projects, i.e., a client interface style and a server API. Under a normal condition, a server provides an HTTP or RPC API, and a client calls the server API through a TCP to achieve a communication purpose.

[0064] In the related art, a commonly used auditing logging manner is usually that control is implemented at a client, and the client records some behaviors of a user on an interface and sends them to a server for storage through a logging API. Another manner is that the server responds to an API calling request in a process of recording data of the API calling request. However, the first manner needs a large number of data requests, so most of a bandwidth is occupied. When the second manner is adopted, although the problem that most of the bandwidth is occupied in the first manner is solved, an I/O operation over log data may slow down a response of the API. That is, the log data recording manners in the related art may affect the response speed to API calling requests.

[0065] The embodiments of the disclosure provide a logging method and apparatus, an electronic device, and a computer-readable storage medium, which may increase the response speed to an API calling request. An exemplary application of the electronic device provided in the embodiments of the disclosure will be described below. The electronic device provided in the embodiments of the disclosure may be implemented as various types of user terminals such as a notebook computer, a tablet computer, a desktop computer, a set-top box, and a mobile device (for example, a mobile phone, a portable music player, a personal digital assistant, a dedicated messaging device, and a portable game device), or may be implemented as an independent physical server, or a server cluster consisting of multiple physical servers or a distributed system, or may be implemented as a cloud server that provides basic cloud computing service such as cloud service, a cloud database, cloud computing, a cloud function, cloud storage, network service, cloud communication, middleware service, domain name service, security service, a Content Delivery Network (CDN), and a big data and artificial intelligence platform. However, the electronic device is not limited thereto.

[0066] Referring to FIG. 1, FIG. 1 is an optional architecture diagram of a logging system 10 according to an embodiment of the disclosure. The logging system 10 includes a client 100 (a client 100A is shown as an example) and a server 200 (a server 200A is shown as an example).

[0067] In some embodiments of the disclosure, a scene state may be a game state of a game table, and moreover, FIG. 1 will be described below taking this as an example. In FIG. 1, the server 200A may be configured to record game states of multiple intelligent game tables in real time. The server 200A may store a working parameter, and for example, may store an angle of a camera arranged on each intelligent game table, game player information, heat maps, and game rules of the intelligent game tables, and an own network configuration parameter of the server 200A, such as a network address, a maximum connection number, a connected client address, and a port number. The client 100A may be an intelligent device that provides a destination website for a user, for example, a desktop computer or a notebook computer. The user logs in to an operation and maintenance management system corresponding to the server 200A through the destination website provided by the client 100A, and manages the working parameter of the server 200A through the operation and maintenance management system. When the user logs in to the operation and maintenance management system corresponding to the server 200A through the client 100A and manages the working parameter of the server 200A through the system, on one hand, the server 200A may intercept an API calling request sent by the client using a first asynchronous thread, generate log data based on the API calling request, and store the log data in a log storage queue configured to store log data to be filtered for storage in a database, and on the other hand, the server 200A may implement real-time recording of the game state of the game table using a service thread different from the first asynchronous thread in response to a modification request in the API calling request.

[0068] Reference is made to FIG. 2. FIG. 2 is an optional flowchart of a logging method according to an embodiment of the disclosure. Descriptions will be made below taking a server as an execution body in combination with the operations shown in FIG. 2.

[0069] In S101, an API calling request sent by a client is intercepted using a first

asynchronous thread.

[0070] In the embodiment of the disclosure, the server may intercept the API calling request sent by the client in a process of communicating with the client.

[0071] In the embodiment of the disclosure, the server may include interceptors arranged at all its own APIs for information interaction with clients. After the interceptors are arranged, all API calling requests sent by clients to the APIs of the server are required to pass through the interceptors arranged at the corresponding APIs, so the API calling request sent by the client to the server may be obtained through the interceptor arranged at the API of the server.

[0072] In the embodiment of the disclosure, the interceptor may be implemented through software codes, or may be implemented in another manner. No limits are made thereto in the embodiment of the disclosure.

[0073] In the embodiment of the disclosure, the API calling request may be an HTTP request.

[0074] In S102, log data is generated based on the API calling request, and the log data is sent to a log storage queue, the log storage queue being configured to store log data to be filtered for storage in a database.

[0075] In the embodiment of the disclosure, when intercepting the API calling request sent by the client through the first asynchronous thread, the server also generates the log data corresponding to the API calling request through the first asynchronous thread and sends the generated log data to its own log storage queue for storage, the log storage queue being configured to store the log data to be filtered for storage in the database.

[0076] In the embodiment of the disclosure, for each API calling request, a piece of log data may be correspondingly generated. Therefore, for multiple API calling requests, multiple pieces of log data may be correspondingly generated.

[0077] In S103, real-time recording of a scene state is implemented using a service thread in response to a modification request in the API calling request.

[0078] In the embodiment of the disclosure, after the API calling request sent by the client is intercepted, the server may further parse the intercepted API calling request using the service thread different from the first asynchronous thread to obtain the modification request, perform corresponding processing according to the modification request to respond to the modification request, and after responding to the modification request, record the scene state in real time based on a processing result obtained by performing corresponding processing according to the modification request.

[0079] In some embodiments of the disclosure, S103 may be implemented by S1031 to S1032.

[0080] In S1031, a working parameter of the service thread is modified using the service thread in response to the modification request in the API calling request to obtain a modified working parameter.

[0081] In S1032, real-time recording of the scene state is implemented based on the modified working parameter.

[0082] In the embodiments of the disclosure, the server may further parse the received API calling request using the service thread different from the first asynchronous thread to obtain the modification request configured to trigger a modification operation over the working parameter, modify its own working parameter according to data contained in the modification request to obtain the modified working parameter, and record the scene state in real time based on the modified working parameter.

[0083] In some embodiments of the disclosure, the service thread may further include a service processing sub thread and a recording sub thread. The service processing sub thread is configured to modify its own working parameter in response to the modification request in the API calling request to obtain the modified working parameter. The recording sub thread is configured to implement real-time recording of the scene state based on the modified working parameter.

[0084] In some embodiments of the disclosure, the data contained in the modification request may be a parameter to be modified and a target parameter, so that the server may modify the parameter to be modified in its own working parameter to the target

parameter according to the modification request.

[0085] For example, when the scene state is a game state of a game table, the parameter to be modified is an angle of a camera arranged on intelligent game table A and the target parameter is θ_1 , the server may modify the angle of the camera arranged on intelligent game table A in its own working parameter from original θ_0 to θ_1 to change the angle of the camera arranged on intelligent game table A. Furthermore, the camera arranged on intelligent game table A may change its own image acquisition angle according to the target parameter. Therefore, the server may subsequently continue to record a game state of intelligent game table A in real time according to a scene image, acquired by the camera of which the image acquisition angle is updated, of intelligent game table A.

[0086] In some other embodiments of the disclosure, the data contained in the modification request may also be other data if the server may correctly modify its own working parameter according to the data in the modification request. No specific limits are made thereto in the embodiments of the disclosure.

[0087] In the embodiments of the disclosure, since the first asynchronous thread is used to intercept the API calling request sent by the client and including the modification request configured to trigger the modification operation over the working parameter, and when the service thread different from the first asynchronous thread is used to modify the working parameter according to the modification request sent by the client and record the scene state using the modified working parameter, the first asynchronous thread is also used to generate the log data according to the API calling request and store the generated log data, generation and storage of the log data may be implemented asynchronously at the same time of recording the scene state to eliminate influences between responding to the API calling request and recording of the log data, further increase the response speed to the API calling request in the process of recording the log data, and reduce recording of unnecessary log data. Therefore, the recording efficiency of the log data is improved, and the redundancy of the recorded log data is reduced.

[0088] In some embodiments of the disclosure, S1032 may further be implemented by S1 to S2, and after S2, S3 may further be executed.

[0089] In S1, a real-time state of a scene is determined based on the modified working parameter and received real-time scene data acquired by an acquisition device.

[0090] In the embodiments of the disclosure, when completing modifying its own working parameter, the server may obtain the real-time state of the scene using the modified working parameter and the real-time scene data (for example, the scene image) received from the acquisition device (for example, the camera).

[0091] In some embodiments of the disclosure, when the scene state is the game state of the game table, the acquisition device may be a camera on the game table, and the real-time scene data may be an acquired image of the game table. The game state may be any state of a game on the game table. For example, the game state may be a game mode adopted by each player in a game process (for example, an operating mode adopted by the player for each link of the game, a game region of the player, and an operating sequence adopted by the player for each link of the game), or may be an operation of a game prop operator, etc. No limits are made thereto in the embodiments of the disclosure.

[0092] In S2, the real-time state is compared with a preset state to obtain a comparison result, and the comparison result is recorded in real time.

[0093] In S3, when the comparison result represents that the real-time state is abnormal, prompt information is output.

[0094] In the embodiments of the disclosure, when determining the real-time state of the scene, the server may compare it and the preset state to judge whether the determined real-time state of the scene is abnormal to obtain the comparison result representing whether the determined real-time state of the scene is abnormal, and record the comparison result for subsequent data analysis. Moreover, when the comparison result represents that the real-time state is abnormal, the server may further send the prompt information to a related device such that a related maintainer may timely intervene in the device related to the scene state, for example, the acquisition device, to enable that the device may run normally.

[0095] In the embodiments of the disclosure, the preset state represents a correct state. For example, when the scene state is the game state of the game table, the preset state

may be a correct game mode adopted by each player in a game (for example, a correct operating mode for operations in each link of the game, a correct game region, and a correct operating sequence for each link of the game), or may be a standard operation of the game prop operator, etc. No limits are made thereto in the embodiments of the disclosure.

[0096] Still taking the condition that the scene state is the game state of the game table as an example, the server, when the determined real-time state of the scene is the game region of each player, may compare the determined game region of each player and the correct game region of each player to further judge whether the game region of each player is correct to finally obtain a comparison result indicating the player corresponding to a wrong game region, and when determining that the game region of a certain player is wrong, sends prompt information to a related display device to give an information prompt to a related maintainer operating the display device.

[0097] In the embodiments of the disclosure, FIG. 3 is an optional flowchart of a logging method according to an embodiment of the disclosure. As shown in FIG. 3, after S102 in FIG. 2, S201 may further be executed. Descriptions will be made below in combination with the operations shown in FIG. 3.

[0098] In S201, the log data in the log storage queue is filtered based on a preset first filtering parameter, and filtered log data is stored in the database in batches.

[0099] In the embodiments of the disclosure, the server may filter the log data stored in the log storage queue using the preset first filtering parameter to remove the log data not required to be stored, and store the log data obtained by filtering and required to be stored in the database to implement recording of the log data.

[00100] In some embodiments of the disclosure, through the first asynchronous thread, the server may filter the log data in the log storage queue based on the preset first filtering parameter and store the filtered log data in the database in batches. In some other embodiments of the disclosure, through the thread, for example, a batch processing thread, different from the first asynchronous thread, the server may also filter the log data in the log storage queue based on the preset first filtering parameter and store the filtered

log data in the database in batches. Here, the two different threads are used to implement generation of the log data and filtering and storage of the generated log data respectively, so that the recording efficiency of the log data may be improved.

[00101] In some embodiments of the disclosure, the first filtering parameter may include at least one of a method parameter, a resource path parameter, a resource path prefix parameter, a parameter blacklist, or a parameter whitelist.

[00102] It is to be noted that the parameter whitelist may include multiple target parameters, and the parameter blacklist may include multiple non-target parameters. In the embodiments of the disclosure, the target parameter or the non-target parameter may be an access time period, an Internet Protocol (IP) address of the client, etc. No limits are made thereto in the embodiments of the disclosure. Exemplarily, when the parameter blacklist includes multiple different access time periods, it indicates that log data corresponding to the multiple different access time periods is not required to be recorded in the database, and is required to be filtered. When the parameter whitelist includes IP addresses of multiple clients, it indicates that log data corresponding to the IP addresses of the multiple clients is required to be recorded in the database, and is required to be filtered.

[00103] Exemplarily, when the first filtering parameter is the method parameter, through the method parameter, log data that satisfies the method parameter in the log data stored in the log storage queue may be determined as the filtered log data and stored in the database. For another example, when the first filtering parameter is at least one target resource path parameter in the parameter whitelist, log data that satisfies the at least one target resource path parameter in the log data stored in the log storage queue may be determined as the filtered log data and stored in the database. Therefore, the redundancy of the log data in the database may be reduced effectively through the first filtering parameter, not only may the query efficiency of the log data be improved when a user queries the log data in the database, but also occupation of a disk may be reduced.

[00104] Here, the filtered log data is stored in the database in a batch processing manner, so that I/O operations over the database may be reduced, and the log storage efficiency may be improved.

[00105] It is to be noted that, as shown in FIG. 3, S201 may be executed together with S103. In another embodiment, S201 may also be executed after S103. No limits are made thereto in the embodiments of the disclosure.

[00106] In some embodiments of the disclosure, FIG. 4 is an optional flowchart of a logging method according to an embodiment of the disclosure. As shown in FIG. 4, after S101 and before S102, S301 may further be executed. S102 may further be implemented by S1021. Descriptions will be made below in combination with the operations shown in FIG. 4.

[00107] In S301, user permission information is obtained using the first asynchronous thread according to a request header obtained by parsing the API calling request, and the user permission information is authenticated to obtain an authentication result.

[00108] In the embodiments of the disclosure, the request header of the API calling request usually includes a plurality of attributes, and for example, may include a browser type and host address that generate the request, and an accepted coding manner and compression manner. For each received API calling request, the server may parse the API calling request, obtain user permission information of the client according to a plurality of attributes in an HTTP header obtained by parsing the API calling request, for example, at least one of a browser type or a host address that generates the request, and judge whether the obtained user permission information is legal according to preset user permission information, thereby implementing authentication of the user permission information.

[00109] In the embodiments of the disclosure, the preset user permission information may be set as practically required. No limits are made thereto in the embodiments of the disclosure.

[00110] In S1021, when the authentication result represents successful authentication, the log data is generated using the first asynchronous thread based on the API calling request, and the log data is stored in the log storage queue, the log storage queue being configured to store the log data to be filtered for storage in the database.

[00111] In the embodiments of the disclosure, for each API calling request, when the

server determines through the first asynchronous thread that the user permission information passes authentication, the corresponding log data may be continued to be generated according to the API calling request through the first asynchronous thread.

[00112] Here, the user permission information obtained by parsing each API calling request is authenticated, and when the obtained authentication result represents successful authentication, the corresponding log data is generated according to each API calling request for subsequent filtering and storage, so that the condition that the client makes a request of faking an operating user or user behavior data may be found timely, and the generated log data is more real and effective.

[00113] In some embodiments of the disclosure, when S301 is further included after S101 and before S102, S103 in FIG. 3 may further be implemented by S1033.

[00114] In S1033, when the authentication result represents successful authentication, real-time recording of the scene state is implemented using the service thread in response to the modification request in the API calling request.

[00115] In the embodiments of the disclosure, for each API calling request, when the user permission information in the API calling request passes authentication and the API calling request includes a modification request, the server may implement real-time recording of the scene state using the service thread in response to the modification request in the API calling request.

[00116] In some embodiments of the disclosure, for each API calling request including no modification request, when user permission information in the API calling request passes authentication, the server may directly respond to the API calling request using the service thread. Exemplarily, when an API calling request including no modification request is a request of logging in to an operation and maintenance management system, the server directly executes a corresponding login operation using the service thread.

[00117] In some embodiments of the disclosure, for each intercepted API calling request, when the API calling request passes authentication of the first asynchronous thread, the API calling request may be stored in another queue different from the log storage queue, and the service thread may regularly acquire the API calling requests that

pass authentication from the other queue to respond to the API calling requests that pass authentication.

[00118] In some other embodiments of the disclosure, for each intercepted API calling request, when the API calling request passes authentication of the first asynchronous thread, the API calling request may be directly sent to the service thread such that the service thread may respond to the received API calling request that passes authentication.

[00119] It can be understood that, when an API calling request including a modification request passes authentication, the own working parameter may be modified in response to the modification request to timely find the condition that the client makes a request of faking an operating user or user behavior data, so that the data security may be improved.

[00120] In some embodiments of the disclosure, before S201 in FIG. 3, S11 to S12 may further be executed.

[00121] In S11, a filtering parameter setting instruction containing a preset first filtering parameter is received.

[00122] In S12, a preset first initial filtering parameter is modified to the preset first filtering parameter based on the filtering parameter setting instruction.

[00123] In the embodiments of the disclosure, before filtering the log data stored in the log storage queue, the server may receive the filtering parameter setting instruction containing the preset first filtering parameter from the user and modify its own preset first initial filtering parameter to the preset first filtering parameter according to the filtering parameter setting instruction to subsequently filter the log data stored in the log storage queue based on the preset first filtering parameter.

[00124] Here, the filtering parameter is dynamically modified to control a filtering logic for the log data, so that the diversity of the filtering logic may be improved, the diversity of the recorded log data is improved, and the data recording flexibility is improved.

[00125] In some embodiments of the disclosure, the operation in S102 in FIG. 2 that the log data is generated based on the API calling request may be implemented by S21 to S23.

[00126] In S21, first data is extracted from a request header obtained by parsing each calling request in the API calling request, the first data at least including user identity data and resource path data.

[00127] In S22, second data is extracted from a request body and request line obtained by parsing each calling request, the second data at least including request method data and request content data.

[00128] In S23, the first data and the second data are combined to obtain a piece of log data corresponding to each calling request.

[00129] Here, the API calling request in the disclosure is an HTTP request. An HTTP request usually includes an HTTP header, a request line, and an HTTP request body. The request line usually includes a request method field, a Uniform Resource Identifier (URI) field, and an HTTP version. A request method usually includes GET, POST, HEAD, PUT, DELETE, OPTIONS, TRACE, CONNECT, etc. The HTTP header usually includes a browser type and host address that generate the request, an accepted coding manner and compression manner, etc. The HTTP request body usually includes request content data, etc. In the embodiments of the disclosure, the server may intercept at least one API calling request every time. When the at least one intercepted API calling request, for each API calling request in the at least one API calling request, first data may be obtained by extracting user permission information, for example, a browser type and host address that generate the request from an HTTP header and extracting a resource path (also called a URI), etc., and a request method, request content data, etc., may be extracted from an HTTP request body and a request line respectively to obtain second data. After the first data and the second data are obtained, the first data and the second data may be combined according to a preset data format, thereby obtaining a piece of structured log data corresponding to the API calling request.

[00130] Here, the preset data format may be set as practically required. No limits are

made thereto in the embodiments of the disclosure.

[00131] For the abovementioned operations, descriptions will be made below taking FIG. 5 as an example. As shown in FIG. 5, the interceptors are arranged at all the APIs of the server. When clients send HTTP requests to the APIs of the server, all the HTTP requests may pass through the interceptors, so the interceptors may obtain all the HTTP requests sent by the clients. When the HTTP requests are intercepted, the server obtains permission tokens of users from HTTP headers of the HTTP requests through the interceptors, and authenticates the users according to the permission tokens of the users to determine whether the users are legal users. When a user is determined as a legal user, a resource path is obtained from the corresponding HTTP header, and an HTTP request body and request line of the corresponding HTTP request are read to obtain a request method and request content data from the HTTP request body and request line of the HTTP request respectively. Then, the permission token of the user, the path, the method, and the request content data are combined into a piece of structured log data for the HTTP request according to a preset format, and the structured log data is stored in the log storage queue. Meanwhile, the server also performs corresponding processing using the service thread according to an HTTP request that passes authentication in the HTTP requests obtained by the interceptors to respond to the HTTP request that passes authentication to implement a service logic. In this manner, the server may asynchronously implement generation of the structured log data corresponding to the HTTP request and responding to the HTTP request.

[00132] In some embodiments of the disclosure, FIG. 6 is an optional flowchart of a logging method according to an embodiment of the disclosure. As shown in FIG. 6, S201 in FIG. 3 may further be implemented by S401 to S404. Descriptions will be made below in combination with the operations shown in FIG. 6.

[00133] In S401, at least one piece of log data is acquired from the log storage queue, the at least one piece of log data being stored in the log storage queue of the server, and each piece of log data including a log parameter.

[00134] In S402, the log parameter of each piece of log data in the at least one piece of log data is matched with the first filtering parameter to obtain a matched matching

parameter.

[00135] In the embodiments of the disclosure, the server may use the abovementioned batch processing thread to acquire the at least one piece of log data stored in the log storage queue and compare the log parameter of each piece of log data in the acquired at least one piece of log data and the first filtering parameter, thereby determining the matching parameter correspondingly the same as the first filtering parameter in the log parameter of the log data.

[00136] Exemplarily, when the first filtering parameter is at least one of the method parameter, the resource path parameter, or the resource path prefix parameter, the first filtering parameter includes N filtering sub parameters, and the log parameter of each piece of log data includes M sub parameters, if the M sub parameters include N sub parameters correspondingly the same as the N filtering sub parameters, the N sub parameters may be determined as matching parameters of the log data. If the M sub parameters do not include N sub parameters correspondingly the same as the N filtering sub parameters (for example, the M sub parameters only include N-1 sub parameters correspondingly the same as N-1 filtering sub parameters, or, the M sub parameters include no sub parameter correspondingly the same as any filtering sub parameter in the N filtering sub parameters), the matching parameter of the log data is null.

[00137] Exemplarily, when the first filtering parameter is the parameter whitelist, the first filtering parameter includes N filtering sub parameters (namely the parameter whitelist includes N target parameters), and the log parameter of each piece of log data includes M sub parameters, if the M sub parameters include at least one sub parameter the same as the corresponding filtering sub parameter in the N filtering sub parameters, the at least one sub parameter may be determined as the matching parameter of the log data. If the M sub parameters do not include any N sub parameters correspondingly the same as the N filtering sub parameters, the matching parameter of the log data is null.

[00138] Exemplarily, when the first filtering parameter is the parameter blacklist, the first filtering parameter includes N filtering sub parameters (namely the parameter blacklist includes N non-target parameters), and the log parameter of each piece of log data includes M sub parameters, if the M sub parameters include at least one sub

parameter the same as the corresponding filtering sub parameter in the N filtering sub parameters, the M sub parameters may be determined as matching parameters of the log data. If the M sub parameters do not include any N sub parameters correspondingly the same as the N filtering sub parameters, the matching parameter of the log data is null.

[00139] Here, both N and M are integers more than or equal to 1. It is to be noted that the matching parameter of each piece of log data may be called a set of matching parameters. A set of matching parameters may include at least one matching parameter.

[00140] In the embodiments of the disclosure, the log parameter of each piece of log data may include at least one of the method parameter (method (array)), the resource path parameter (path (array)), the resource path prefix parameter (prefix (array)), an access time period, an IP address of the client, etc.

[00141] In S403, the filtered log data is determined according to at least one piece of target log data corresponding to the matching parameter.

[00142] In the embodiments of the disclosure, for each piece of log data, when the matching parameter is null, the matching parameter does not correspond to the log data, namely the matching parameter does not correspond to any piece of target log data. When the matching parameter is not null, the matching parameter corresponds to the log data, and the log data is a piece of target log data. For example, for log data A, when a matching parameter matched with the first filtering parameter in a log parameter of log data A is null, log data A is not a piece of target log data, and when the matching parameter matched with the first filtering parameter in the log parameter of log data A is not null, log data A is a piece of target log data. It is to be noted that, since each piece of log data corresponds to a set of matching parameters, and there is at least one piece of log data at present, for each piece of log data, multiple sets of matching parameters may be obtained, the at least one piece of target log data may be obtained according to the multiple matching parameters, and the filtered log data may be determined according to the obtained at least one piece of target log data.

[00143] In S404, when the amount of the filtered log data reaches a preset amount, the filtered log data is stored in the database.

[00144] In the embodiments of the disclosure, when the server determines all the filtered log data through the batch processing thread, and the amount of the filtered log data reaches the preset amount, all the filtered log data is stored in the database.

[00145] In some embodiments of the disclosure, S401 in FIG. 6 may be implemented by S4011 to S4012.

[00146] In S4011, the log data is read from the log storage queue according to a preset frequency.

[00147] In S4012, when present log data is read, a next piece of log data is continued to be read from the log storage queue until a last piece of log data stored in the log storage queue is read, thereby acquiring the at least one piece of log data.

[00148] In the embodiments of the disclosure, the server may read the log data from the log storage queue using the batch processing thread according to the preset frequency. When a piece of log data is read at a certain time, the log data is continuously read from the log storage queue until all the log data stored in the log storage queue is read, thereby acquiring the at least one piece of log data. For example, the server reads the log data from the log storage queue every 10ms using the processing thread. When no log data is read at the first time, namely the read data is null, log data is read from the log storage queue 10ms later at the second time. When a piece of log data is read from the log storage queue at the second time, log data is continuously read from the log storage queue until ten pieces of log data stored in the log storage queue are read, to obtain the ten pieces of log data.

[00149] It is to be noted that, after the ten pieces of log data stored in the log storage queue are read, the server may continue to read log data from the log storage queue 10ms later through the batch processing thread and judge whether data is read. When no log data is read, log data is continued to be read from the log storage queue 10ms later. When log data is read, log data is continuously read from the log storage queue until all log data stored in the log storage queue is read. The operations are cyclically performed according to the method, and reading is stopped when a reading stopping instruction is received.

[00150] In the embodiments of the disclosure, the preset frequency may be set as

practically, and for example, may be 1s, 10ms or 30ms. No limits are made thereto in the embodiments of the disclosure.

[00151] In some embodiments of the disclosure, the matching parameter includes a method parameter, a resource path resource, a resource path prefix parameter, a non-target parameter in a parameter blacklist, or a target parameter in a parameter whitelist.

[00152] In some embodiments of the disclosure, S403 may be implemented by S4031 to S4032.

[00153] In S4031, candidate log data is determined according to the at least one piece of target log data corresponding to the matching parameter.

[00154] In S4032, the filtered log data is determined based on the candidate log data.

[00155] In some embodiments of the disclosure, the server may determine the candidate log data according to the at least one piece of target log data corresponding to the matching parameter at first, and after obtaining the matching parameters of all the log data, determines the filtered log data according to the candidate log data.

[00156] In some embodiments of the disclosure, the operation that the candidate log data is determined according to the at least one piece of target log data corresponding to the matching parameter may be implemented by directly determining the at least one piece of target log data corresponding to the matching parameter as the candidate log data. In some other embodiments of the disclosure, the operation that the candidate log data is determined according to the obtained at least one piece of target log data may also be implemented by determining the other log data except the at least one piece of target log data corresponding to the matching parameter as the candidate log data.

[00157] In some embodiments of the disclosure, S4031 may further be implemented by at least one of S501 to S505.

[00158] In S501, when the matching parameter is the method parameter, at least one piece of target log data corresponding to the method parameter in the at least one piece of log data is determined as the candidate log data.

[00159] In some embodiments of the disclosure, for each piece of log data in the acquired at least one piece of log data, when the matching parameter of the log data is not null and is the method parameter, the log data is determined as a piece of target log data. Therefore, the acquired at least one piece of log data may include at least one piece of target log data, and the server may determine the at least one piece of target log data determined from the acquired at least one piece of log data as the candidate log data.

[00160] In S502, when the matching parameter is the resource path parameter, at least one piece of target log data corresponding to the resource path parameter is determined as the candidate log data.

[00161] In some embodiments of the disclosure, for each piece of log data in the acquired at least one piece of log data, when the matching parameter of the log data is not null and is the resource path parameter, the log data is determined as a piece of target log data. Therefore, the acquired at least one piece of log data may include at least one piece of target log data, and the server may determine the at least one piece of target log data determined from the acquired at least one piece of log data as the candidate log data.

[00162] In S503, when the matching parameter is the resource path prefix parameter, at least one piece of target log data corresponding to the resource path prefix parameter in the at least one piece of log data is determined as the candidate log data.

[00163] In some embodiments of the disclosure, for each piece of log data in the acquired at least one piece of log data, when the matching parameter of the log data is not null and is the resource path prefix parameter, the log data is determined as a piece of target log data. Therefore, the acquired at least one piece of log data may include at least one piece of target log data, and the server may determine the at least one piece of target log data determined from the acquired at least one piece of log data as the candidate log data.

[00164] From the above, when the matching parameter is the method parameter, the resource path parameter, and the resource path prefix parameter, the operation in S4031 that the candidate log data is determined according to the at least one piece of target log data corresponding to the matching parameter is implemented by directly determining the

at least one piece of target log data corresponding to the matching parameter as the candidate log data.

[00165] In S504, when the matching parameter is the at least one non-target parameter in the parameter blacklist, log data, except at least one piece of target log data corresponding to the at least one non-target parameter, in the at least one piece of log data is determined as the candidate log data.

[00166] In some other embodiments of the disclosure, for each piece of log data in the acquired at least one piece of log data, when the matching parameter of the log data is not null and is at least one non-target parameter in the parameter blacklist, the log data is determined as a piece of target log data. Therefore, the acquired at least one piece of log data may include at least one piece of target log data, and the server may determine the log data, except the at least one piece of target log data, in the acquired at least one piece of log data as the candidate log data. Exemplarily, when the acquired at least one piece of log data includes five pieces, and three pieces of target log data are determined from the five pieces of log data, the other two pieces of log data in the five pieces of log data are determined as candidate log data. That is, when the matching parameter is the at least one non-target parameter in the parameter blacklist, the resource path parameter, and the resource path prefix parameter, the operation in S4031 that the candidate log data is determined according to the at least one piece of target log data corresponding to the matching parameter is implemented by determining the other log data except the at least one piece of target log data corresponding to the matching parameter as the candidate log data.

[00167] In S505, when the matching parameter is the at least one target parameter in the parameter whitelist, at least one piece of target log data corresponding to the at least one target parameter in the at least one piece of log data is determined as the candidate log data.

[00168] In some embodiments of the disclosure, for each piece of log data in the acquired at least one piece of log data, when the matching parameter of the log data is not null and is at least one target parameter in the parameter whitelist, the log data is determined as a piece of target log data. Therefore, the acquired at least one piece of log

data may include at least one piece of target log data, and the server may determine the sub parameter at least one piece of target log data determined from the acquired at least one piece of log data as the candidate log data.

[00169] In some embodiments of the disclosure, when S4031 is implemented by S501 to S505, S4032 may be implemented in the following manner: the server may determine all the log data corresponding to the method parameter, the resource path parameter, and the at least one target parameter but not corresponding to the at least one non-target parameter from all the candidate log data formed by the at least one piece of target log data corresponding to the method parameter, the at least one piece of target log data corresponding to the resource path parameter, the log data except the at least one piece of target log data corresponding to the at least one non-target parameter, and the at least one piece of target log data corresponding to the at least one target parameter, and determine the determined log data as the filtered log data. That is, when S4031 is implemented by S501 to S505, the filtered log data finally determined by S4032 corresponds to the method parameter, the resource parameter, and the at least one target parameter in the parameter whitelist, and does not correspond to the at least one non-target parameter in the parameter blacklist.

[00170] In some embodiments of the disclosure, when S4031 is implemented by S501, implemented by S502, implemented by S503, or implemented by S505, S4032 may be implemented in the following manner: the server may directly determine the determined candidate log data as the filtered log data through the batch processing thread. For example, when S4031 is implemented by S501, the server may directly determine the at least one piece of target log data corresponding to the method parameter as the filtered log data through the batch processing thread.

[00171] Exemplarily, when the matching parameter is a path value for logging in to the API, the server may store login log data in the database for recording as filtered log data.

[00172] Exemplarily, when the matching parameter is "post,put,delete", the server may store all scene modification log data corresponding to the modification request configured to modify its own working parameter in the database for recording as filtered log data.

[00173] In some other embodiments of the disclosure, when S4031 is implemented by S501 to S502, implemented by S501 to S503, implemented by S501 to S504, implemented by S502 to S503, implemented by S502 to S504, implemented by S503 to S504, or implemented by S504 to S505, etc., S4032 may be implemented in the following manner: the server may determine the log data that satisfy all corresponding filtering parameters in all the determined candidate log data as the filtered log data through the batch processing thread. For example, when S4031 is implemented by S501 to S502, S4032 may be implemented in the following manner: the server may determine the log data that corresponds to the method parameter and also corresponds to the resource path parameter in the at least one piece of target log data corresponding to the method parameter and the at least one piece of target log data corresponding to the resource path parameter as the filtered log data through the batch processing thread. For another example, when S4031 is implemented by S501 to S503, S4032 may be implemented in the following manner: the server may determine the log data that corresponds to both the method parameter and the resource path parameter and also corresponds to the resource path prefix parameter in all the candidate log data formed by the at least one piece of target log data corresponding to the method parameter, the at least one piece of target log data corresponding to the resource path parameter, and the at least one piece of target log data corresponding to the resource path prefix parameter as the filtered log data through the batch processing thread.

[00174] In some embodiments of the disclosure, as shown in FIG. 7, after S201 in FIG. 3, S601 may further be executed. Descriptions will be made below in combination with the operations shown in FIG. 7.

[00175] In S601, log data of which a time parameter does not satisfy a second filtering parameter in the database is deleted using a second asynchronous thread, the second filtering parameter representing a preset storage duration of each piece of log data in the database, each piece of log data stored in the database including a time parameter, and the time parameter representing a duration when each piece of log data has been stored in the database.

[00176] In the embodiments of the disclosure, each piece of log data stored in the

database corresponds to a time parameter. The server may extract multiple pieces of log data from the database using the second asynchronous thread different from the first asynchronous thread, compare the time parameter of each piece of log data and the second filtering parameter representing the preset storage duration of each piece of log data in the database, and when the time parameter of a certain piece of log data is greater than the second filtering parameter, delete the log data from the database.

[00177] In some embodiments of the disclosure, after storing each piece of log data in the database through the first asynchronous thread, the server may set a timer for the log data to count a storage duration of the log data in the database. A numerical value of the timer is the time parameter of the log data. In some embodiments of the disclosure, when comparing the time parameter of each piece of log data and the second filtering parameter through the second asynchronous thread, the server may extract multiple pieces of log data at one time from the database for comparison one by one, or may read one piece of log data at one time from the database for further comparison. No limits are made thereto in the embodiments of the disclosure.

[00178] Exemplarily, when the second filtering parameter is "history-remain (365d)", the server may clean historical log data of which a storage duration exceeds a year in the database.

[00179] For the abovementioned contents, descriptions will be made below taking FIG. 8 as an example. As shown in FIG. 8, the first asynchronous processing thread may include the batch processing thread, and the second asynchronous thread may be a cleaning thread. The batch processing thread may read multiple pieces of structured log data from the log storage queue according to the preset frequency. For each piece of structured log data that is read, the batch processing thread matches each parameter in a log parameter of the structured log data and the first filtering parameter, stores the structured log data in the database when the first filtering parameter includes two filtering sub parameters method and path and the log parameter of the log structure includes two sub parameters correspondingly the same as the two filtering sub parameters method and path, and discards the structured log data when the log parameter does not include the two sub parameters correspondingly the same as the two filtering sub parameters method

and path. Meanwhile, the cleaning thread may regularly extract each piece of structured log data from the database, compare a time parameter of each piece of structured log data and the second filtering parameter, determine whether the structured log data expires according to a comparison result, discards the structured log data when determining that the structured log data expires, and puts the structured log data back to the database for continuous storage when determining that the structured log data does not expire.

[00180] FIG. 9 is a structure composition diagram of a logging apparatus according to an embodiment of the disclosure. As shown in FIG. 9, the logging apparatus 20 includes an interceptor 2001, configured to intercept an API calling request sent by a client, generate log data based on the API calling request, and store the log data in a log storage queue, the log storage queue being configured to store log data to be filtered for storage in a database; and a service thread 2002, configured to implement real-time recording of a scene state in response to a modification request in the API calling request.

[00181] In some embodiments, the interceptor 2001 is further configured to filter the log data in the log storage queue based on a preset first filtering parameter and store filtered log data in the database in batches.

[00182] In some embodiments, the interceptor 2001 is further configured to: using a first asynchronous thread, obtain user permission information according to a request header obtained by parsing the API calling request, and authenticate the user permission information to obtain an authentication result, and when the authentication result represents successful authentication, generate the log data using the first asynchronous thread based on the API calling request.

[00183] In some embodiments, the interceptor 2001 is further configured to extract first data from a request header obtained by parsing each calling request in the API calling request, the first data at least including user identity data and resource path data, extract second data from a request body and request line obtained by parsing each calling request, the second data at least including request method data and request content data, and combine the first data and the second data to obtain a piece of log data corresponding to each calling request.

[00184] In some embodiments, at least one piece of log data is stored in the log storage queue, and each piece of log data includes a log parameter. The interceptor 2001 is further configured to acquire the at least one piece of log data from the log storage queue, match the log parameter of each piece of log data in the at least one piece of log data and the first filtering parameter to obtain a matched matching parameter, determine the filtered log data according to at least one piece of target log data corresponding to the matching parameter, and when the amount of the filtered log data reaches a preset amount, store the filtered log data in the database.

[00185] In some embodiments, the interceptor 2001 is further configured to read the log data from the log storage queue according to a preset frequency, and when present log data is read, continue to read a next piece of log data from the log storage queue until a last piece of log data stored in the log storage queue is read, thereby acquiring the at least one piece of log data.

[00186] In some embodiments, the matching parameter includes at least one of a method parameter, a resource path resource, a resource path prefix parameter, a non-target parameter in a parameter blacklist, or a target parameter in a parameter whitelist.

[00187] In some embodiments, the interceptor 2001 is further configured to determine candidate log data according to the at least one piece of target log data corresponding to the matching parameter and determine the filtered log data based on the candidate log data.

[00188] In some embodiments, the interceptor 2001 is further configured to implement at least one of the following operations:

[00189] when the matching parameter is the method parameter, determining at least one piece of target log data corresponding to the method parameter in the at least one piece of log data as the candidate log data; when the matching parameter is the resource path parameter, determining at least one piece of target log data corresponding to the resource path parameter in the at least one piece of log data as the candidate log data; when the matching parameter is the resource path prefix parameter, determining at least one piece of target log data corresponding to the resource path prefix parameter in the at

least one piece of log data as the candidate log data; when the matching parameter is the at least one non-target parameter in the parameter blacklist, determining log data, except at least one piece of target log data corresponding to the at least one non-target parameter, in the at least one piece of log data as the candidate log data; and when the matching parameter is the at least one target parameter in the parameter whitelist, determining at least one piece of target log data corresponding to the at least one target parameter in the at least one piece of log data as the candidate log data.

[00190] In some embodiments, the apparatus further includes a modification module, configured to receive a filtering parameter setting instruction containing the preset first filtering parameter and modify a preset first initial filtering parameter to the preset first filtering parameter in response to the filtering parameter setting instruction.

[00191] In some embodiments, the service thread 2002 is further configured to modify an own working parameter using the service thread in response to the modification request in the API calling request to obtain a modified working parameter, determine a real-time state of a scene based on the modified working parameter and received real-time scene data acquired by an acquisition device, compare the real-time state and a preset state to obtain a comparison result, and record the comparison result in real time.

[00192] In some embodiments, the service thread 2002 is further configured to, when the comparison result represents that the real-time state is abnormal, output prompt information.

[00193] In some embodiments, each piece of log data stored in the database includes a time parameter, the time parameter representing a duration when each piece of log data has been stored in the database. The apparatus further includes a second asynchronous thread, configured to delete log data of which the time parameter does not satisfy a second filtering parameter in the database using the second asynchronous thread, the second filtering parameter representing a preset storage duration of each piece of log data in the database.

[00194] It is to be noted that "module" in the embodiments of the disclosure represents a software module, or a module including a software part and a hardware part, etc.

[00195] FIG. 10 is a structure composition diagram of an electronic device according to an embodiment of the disclosure. As shown in FIG. 10, the electronic device 30 includes a memory 3001, a processor 3002, and a computer program stored in the memory 3001 and capable of running in the processor 3002. The processor 3002 is configured to run the computer program to execute the logging method in the abovementioned embodiments.

[00196] It can be understood that the electronic device 200 further includes a communication bus 3003. Each device in the electronic device 30 is coupled together through the communication bus 3003. It can be understood that the communication bus 3003 is configured to implement connection communication between these devices. The communication bus 3003 includes a data bus, and further includes a power bus, a control bus, and a state signal bus.

[00197] The memory 3001 is configured to store a computer program and application executed by the processor 3002, may also cache data (for example, image data, video data, voice communication data, and video communication data) to be processed or having been processed by the processor 3002 and each module in the electronic device, and may be implemented through a flash or a Random Access Memory (RAM).

[00198] The processor 3002 executes the program to implement the steps of the logging method. The processor 3002 usually controls overall operations of the electronic device 30.

[00199] The processor may be at least one of an Application Specific Integrated Circuit (ASIC), a Digital Signal Processor (DSP), a Digital Signal Processing Device (DSPD), a Programmable Logic Device (PLD), a Field Programmable Gate Array (FPGA), a Central Processing Unit (CPU), a controller, a microcontroller, or a microprocessor. It can be understood that other electronic devices may also be configured to realize functions of the processor. No limits are made in the embodiments of the disclosure.

[00200] The computer-readable storage medium/memory may be a memory such as a Read Only Memory (ROM), a Programmable Read-Only Memory (PROM), an Erasable Programmable Read-Only Memory (EPROM), an Electrically Erasable Programmable

Read-Only Memory (EEPROM), a Ferromagnetic Random Access Memory (FRAM), a flash memory, a magnetic surface memory, an optical disk, or a Compact Disc Read-Only Memory (CD-ROM), or may be any terminal including one or any combination of the abovementioned memories, such as a mobile phone, a computer, a tablet device, and a personal digital assistant.

[00201] The embodiments of the disclosure provide a computer program product or a computer program, which includes a computer instruction stored in a computer-readable storage medium. A processor of a computer device reads the computer instruction from the computer-readable storage medium. The processor executes the computer instruction to enable the computer device to execute the logging method of the embodiments of the disclosure.

[00202] In some embodiments, the executable instruction may be compiled according to a programming language of any form (including a compiling or interpretive language, or a declarative or procedural language) in form of a program, software, a software module, a script, or a code, and may be deployed according to any form, including deployed as an independent program or deployed as a module, a component, a subroutine or another unit suitable to be used in a computing environment.

[00203] As an example, the executable instruction may but not always correspond to a file in a file system, and may be stored in a part of a file that stores another program or data, for example, stored in one or more scripts in a Hyper Text Markup Language (HTML) document, stored in a single file dedicated to a discussed program, or stored in multiple collaborative files (for example, files storing one or more modules, subprograms or code parts).

[00204] As an example, the executable instruction may be deployed in a computing device for execution, or executed in multiple computing devices at the same place, or executed in multiple computing devices that are interconnected through a communication network at multiple places.

[00205] It is to be pointed out here that the above descriptions about the storage medium and device embodiments are similar to the descriptions about the method

embodiment, and beneficial effects similar to those of the method embodiment are achieved. Technical details undisclosed in the storage medium and device embodiments of the disclosure are understood with reference to the descriptions about the method embodiment of the disclosure.

[00206] The above is only the embodiment of the disclosure and not intended to limit the scope of protection of the disclosure. Any modifications, equivalent replacements, improvements and the like made within the spirit and scope of the disclosure fall within the scope of protection of the disclosure.

CLAIMS

1. A logging method, comprising:

intercepting, using a first asynchronous thread, an Application Programming Interface (API) calling request sent by a client;

generating log data based on the API calling request, and storing the log data in a log storage queue, the log storage queue being configured to store log data to be filtered for storage in a database; and

implementing real-time recording of a scene state using a service thread in response to a modification request in the API calling request.

2. The logging method of claim 1, further comprising:

filtering the log data in the log storage queue based on a preset first filtering parameter, and storing the filtered log data in the database in batches.

3. The logging method of claim 1 or 2, further comprising:

obtaining user permission information using the first asynchronous thread according to a request header obtained by parsing the API calling request, and authenticating the user permission information to obtain an authentication result,

wherein generating the log data based on the API calling request comprises:

in a case where the authentication result represents successful authentication, generating the log data using the first asynchronous thread based on the API calling request.

4. The logging method of any one of claims 1-3, wherein generating the log data based on the API calling request comprises:

extracting first data from a request header obtained by parsing each calling request in the API calling request, the first data at least comprising user identity data and resource path data;

extracting second data from a request body and request line obtained by parsing each calling request, the second data at least comprising request method data and request content data; and

combining the first data and the second data to obtain a piece of log data corresponding to each calling request.

5. The logging method of claim 2, wherein at least one piece of log data is stored in the log storage queue, and each piece of log data comprises a log parameter; and

filtering the log data in the log storage queue based on the preset first filtering parameter and storing the filtered log data in the database in batches comprises:

acquiring the at least one piece of log data from the log storage queue,

matching the log parameter of the each piece of log data in the at least one piece of log data and the first filtering parameter to obtain a matched matching parameter,

determining the filtered log data according to at least one piece of target log data corresponding to the matching parameter, and

when the amount of the filtered log data reaches a preset amount, storing the filtered log data in the database.

6. The logging method of claim 5, wherein acquiring the at least one piece of log data from the log storage queue comprises:

reading the log data from the log storage queue according to a preset frequency; and

in a case where present log data is read, continuing to read a next piece of log data from the log storage queue until a last piece of log data stored in the log storage queue is read, thereby acquiring the at least one piece of log data.

7. The logging method of claim 5, wherein the matching parameter comprises at least one of:

a method parameter, a resource path resource, a resource path prefix parameter, a non-target parameter in a parameter blacklist, or a target parameter in a parameter whitelist.

8. The logging method of claim 5, wherein determining the filtered log data according to the at least one piece of target log data corresponding to the matching parameter comprises:

determining candidate log data according to the at least one piece of target log data corresponding to the matching parameter; and

determining the filtered log data based on the candidate log data.

9. The logging method of claim 8, wherein determining the candidate log data according to the at least one piece of target log data corresponding to the matching parameter comprises at least one of:

in a case where the matching parameter is the method parameter, determining at least one piece of target log data corresponding to the method parameter in the at least one piece of log data as the candidate log data;

in a case where the matching parameter is the resource path parameter, determining at least one piece of target log data corresponding to the resource path parameter in the at least one piece of log data as the candidate log data;

in a case where the matching parameter is the resource path prefix parameter, determining at least one piece of target log data corresponding to the resource path prefix parameter in the at least one piece of log data as the candidate log data;

in a case where the matching parameter is the at least one non-target parameter in the parameter blacklist, determining log data, except at least one piece of target log data corresponding to the at least one non-target parameter, in the at least one piece of log data as the candidate log data; or

in a case where the matching parameter is the at least one target parameter in the parameter whitelist, determining at least one piece of target log data corresponding to the at least one target parameter in the at least one piece of log data as the candidate log data.

10. The logging method of claim 2 or any one of claims 5-8, further comprising:

receiving a filtering parameter setting instruction containing the preset first filtering parameter; and

modifying a preset first initial filtering parameter to the preset first filtering parameter in response to the filtering parameter setting instruction.

11. The logging method of any one of claims 1-10, wherein implementing real-time recording of the scene state using the service thread in response to the modification request in the API calling request comprises:

modifying a working parameter of the service thread using the service thread in response to the modification request in the API calling request to obtain a modified working parameter;

determining a real-time state of a scene based on the modified working parameter and received real-time scene data acquired by an acquisition device; and

comparing the real-time state and a preset state to obtain a comparison result, and

recording the comparison result in real time.

12. The logging method of claim 11, further comprising:

in a case where the comparison result represents that the real-time state is abnormal, outputting prompt information.

13. An electronic device, comprising:

a memory, configured to store an executable computer program; and

a processor, configured to execute the executable computer program stored in the memory to implement following operations:

intercepting, using a first asynchronous thread, an Application Programming Interface (API) calling request sent by a client;

generating log data based on the API calling request, and storing the log data in a log storage queue, the log storage queue being configured to store log data to be filtered for storage in a database; and

implementing real-time recording of a scene state using a service thread in response to a modification request in the API calling request.

14. The electronic device of claim 13, wherein the processor is further configured to implement following operations:

filtering the log data in the log storage queue based on a preset first filtering parameter, and storing the filtered log data in the database in batches.

15. The electronic device of claim 13 or 14, wherein the processor is further configured to implement following operations:

obtaining user permission information using the first asynchronous thread according to a request header obtained by parsing the API calling request, and authenticating the user permission information to obtain an authentication result,

wherein generating the log data based on the API calling request comprises:

in a case where the authentication result represents successful authentication, generating the log data using the first asynchronous thread based on the API calling request.

16. The electronic device of any one of claims 13-15, wherein the operation of generating the log data based on the API calling request comprises:

extracting first data from a request header obtained by parsing each calling request in the API calling request, the first data at least comprising user identity data and resource path data;

extracting second data from a request body and request line obtained by parsing each calling request, the second data at least comprising request method data and request content data; and

combining the first data and the second data to obtain a piece of log data corresponding to each calling request.

17. The electronic device of claim 14, wherein at least one piece of log data is stored in the log storage queue, and each piece of log data comprises a log parameter; and

the operations of filtering the log data in the log storage queue based on the preset first filtering parameter and storing the filtered log data in the database in batches comprise:

acquiring the at least one piece of log data from the log storage queue,

matching the log parameter of the each piece of log data in the at least one piece of log data and the first filtering parameter to obtain a matched matching parameter,

determining the filtered log data according to at least one piece of target log data corresponding to the matching parameter, and

when the amount of the filtered log data reaches a preset amount, storing the filtered log data in the database.

18. The electronic device of claim 14 or 17, wherein the processor is further configured to implement following operations:

receiving a filtering parameter setting instruction containing the preset first filtering parameter; and

modifying a preset first initial filtering parameter to the preset first filtering parameter in response to the filtering parameter setting instruction.

19. A computer-readable storage medium, storing a computer program, configured to be executed by a processor to implement following operations:

intercepting, using a first asynchronous thread, an Application Programming Interface (API) calling request sent by a client;

generating log data based on the API calling request, and storing the log data in a log storage queue, the log storage queue being configured to store log data to be filtered for storage in a database; and

implementing real-time recording of a scene state using a service thread in response to a modification request in the API calling request.

20. A computer program, comprising computer-readable codes which, when executed in an electronic device, cause a processor in the electronic device to implement following operations:

intercepting, using a first asynchronous thread, an Application Programming Interface (API) calling request sent by a client;

generating log data based on the API calling request, and storing the log data in a log storage queue, the log storage queue being configured to store log data to be filtered for storage in a database; and

implementing real-time recording of a scene state using a service thread in response to a modification request in the API calling request.