METHOD AND APPARATUS FOR OPTIMIZING PERFORMANCE PARAMETERS OF DATABASE, AND ELECTRONIC DEVICE

[0001] This application claims priority to Chinese Patent Application No. 202411786864.0 filed with the China National Intellectual Property Administration (CNIPA) on Dec. 6, 2024, the disclosure of which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present application relates to the field of distributed databases, for example, to a method and apparatus for optimizing performance parameters of a database, and an electronic device.

BACKGROUND

[0003] Distributed databases are widely adopted in modern businesses and can primarily achieve efficient operation through a layered design including a compute engine layer and a data storage layer. In this architecture, each layer is further composed of multiple nodes that can process business statements concurrently, thereby enabling near-linear performance improvement as the business volume increases. This scalability makes distributed databases the preferred solution for large-scale, high-concurrency scenarios. However, the complexity of the distributed architecture also brings significant challenges, particularly in database performance optimization. Since structured query language (SQL) execution involves multiple nodes and multiple layers, performance is constrained by numerous factors such as inter-node coordination, resource allocation, and hardware bottlenecks. Meanwhile, the interactions among components across layers further complicate troubleshooting and optimization processes, often requiring a comprehensive analysis of multi-dimensional performance indexes, including the central processing unit (CPU), memory, disk input/output (I/O), and network bandwidth. This multi-factor coupling relationship leads to extremely difficult database parameter tuning, which necessitates not only refinement down to specific execution stages but also comprehensive debugging and optimization in conjunction with server performance, thereby incurring substantial manual analysis costs. Therefore, how to effectively collect and analyze performance

consumption information during execution processes and accurately perform parameter tuning based on such data is an urgent problem to be solved in current distributed database optimization.

SUMMARY

[0004] The present application provides a method and apparatus for optimizing performance parameters of a database, and an electronic device to solve the problem of great difficulty in parameter tuning caused by the complexity of the distributed database architecture.

[0005] In a first aspect, the present application provides a method for optimizing performance parameters of a database. The method includes: each SQL statement in an SQL statement set is inputted into an initial database so that the initial database executes each SQL statement, and operation data of each SQL statement is outputted to obtain a first data set, where the SQL statement set includes multiple types of SQL statements, and the operation data includes performance consumption data of a server of each node of the initial database and execution duration data of each SQL statement at each node when the initial database executes each SQL statement; classification and aggregation are performed on the first data set to obtain operation data of each type of SQL statements, and a comprehensive evaluation index value of each type of SQL statements is calculated according to the operation data of each type of SQL statements; a complete parameter set of the initial database is analyzed according to the first data set to obtain a first parameter set requiring optimization; all the comprehensive evaluation index values and the first parameter set are inputted into a prediction model so that the prediction model performs an initial optimization on the first parameter set according to all the comprehensive evaluation index values to obtain a second parameter set after the initial optimization; and multiple iterative optimizations are performed on the second parameter set using the initial database and the prediction model to obtain an optimal parameter set of the initial database.

[0006] In a second aspect, the present application provides an apparatus for optimizing performance parameters of a database. The apparatus includes: an execution module, configured

to input each SQL statement in an SQL statement set into an initial database so that the initial database executes each SQL statement, and output operation data of each SQL statement to obtain a first data set, where the SQL statement set includes multiple types of SQL statements, and the operation data includes performance consumption data of a server of each node of the initial database and execution duration data of each SQL statement at each node when the initial database executes each SQL statement; a calculation module, configured to perform classification and aggregation on the first data set to obtain operation data of each type of SQL statements and calculate a comprehensive evaluation index value of each type of SQL statements according to the operation data of each type of SQL statements; an analysis module, configured to analyze a complete parameter set of the initial database according to the first data set to obtain a first parameter set requiring optimization; a first optimization module, configured to input all the comprehensive evaluation index values and the first parameter set into a prediction model so that the prediction model performs an initial optimization on the first parameter set according to all the comprehensive evaluation index values to obtain a second parameter set after the initial optimization; and a second optimization module, configured to perform multiple iterative optimizations on the second parameter set using the initial database and the prediction model to obtain an optimal parameter set of the initial database.

[0007] In a third aspect, the present application provides a computer-readable storage medium. The computer-readable storage medium stores a computer program which, when executed by a processor, causes the processor to perform the method for optimizing performance parameters of a database described above.

[0008] In a fourth aspect, the present application provides an electronic device. The electronic device includes a memory and a processor. The memory stores a computer program, and the processor is configured to perform the method for optimizing performance parameters of a database through the computer program.

BRIEF DESCRIPTION OF DRAWINGS

[0009] The drawings herein, which are incorporated into and constitute a part of the

specification, illustrate embodiments consistent with the present application, and together with the specification, serve to explain the present application.

[0010] The drawings that need to be used in the description of the embodiments or the related art are briefly described below. For those of ordinary skill in the art, other drawings may be obtained according to these drawings without creative efforts.

[0011] One or more embodiments are illustrated exemplarily by means of figures in the drawings corresponding thereto. Elements with the same reference numerals in the drawings are represented as like elements, and the figures in the drawings are not intended to constitute a scale limitation unless specifically stated otherwise.

[0012] FIG. 1 is a flowchart of a method for optimizing performance parameters of a database according to an embodiment of the present application.

[0013] FIG. 2 is an architecture diagram of the method for optimizing performance parameters of a database according to an embodiment of the present application.

[0014] FIG. 3 is an implementation flowchart of the method for optimizing performance parameters of a database according to an embodiment of the present application.

[0015] FIG. 4 is an analysis flowchart of a log analysis module of the method for optimizing performance parameters of a database according to an embodiment of the present application.

[0016] FIG. 5 is an analysis flowchart of an analysis processing module of the method for optimizing performance parameters of a database according to an embodiment of the present application.

[0017] FIG. 6 is a structure diagram of an apparatus for optimizing performance parameters of a database according to an embodiment of the present application.

[0018] FIG. 7 is a schematic diagram of an electronic device according to an embodiment of the present application.

DETAILED DESCRIPTION

[0019] A number of different embodiments or examples are given below for implementing different structures of the present application. To simplify the disclosure of the present application, the components and arrangements of specific examples are described below. Definitely, the above are only examples. In addition, the present application may repeat reference numerals and/or reference letters in different examples. Such repetition is for the sake of simplicity and clarity, and does not itself indicate the relationship between the various embodiments and/or arrangements discussed.

[0020] According to a first aspect of the embodiments of the present application, a method for optimizing performance parameters of a database is provided. Optionally, as shown in FIG. 1, the method includes S102 to S110 below.

[0021] In S102, each SQL statement in an SQL statement set is inputted into an initial database so that the initial database executes each SQL statement, and operation data of each SQL statement is outputted to obtain a first data set; where the SQL statement set includes multiple types of SQL statements, and the operation data includes performance consumption data of a server of each node of the initial database and execution duration data of each SQL statement at each node when the initial database executes each SQL statement.

[0022] In S104, classification and aggregation are performed on the first data set to obtain operation data of each type of SQL statements, and a comprehensive evaluation index value of each type of SQL statements is calculated according to the operation data of each type of SQL statements.

[0023] In S106, a complete parameter set of the initial database is analyzed according to the first data set to obtain a first parameter set requiring optimization.

[0024] In S108, all the comprehensive evaluation index values and the first parameter set are inputted into a prediction model so that the prediction model performs an initial optimization on the first parameter set according to all the comprehensive evaluation index values to obtain a

second parameter set after the initial optimization.

[0025] In S110, multiple iterative optimizations are performed on the second parameter set using the initial database and the prediction model to obtain an optimal parameter set of the initial database.

[0026] Optionally, in this embodiment, the architecture diagram shown in FIG. 2 includes an application (APP), a database, a log analysis module, and an analysis processing module. The APP is used for issuing business statements or SQL statements. The database is responsible for storing business data, executing business SQL statements, and returning execution results. The distributed database is generally divided into a compute engine layer and a data storage layer. Each layer is composed of multiple nodes that can process business statements concurrently, thereby achieving significant near-linear improvement in business performance. The log analysis module is configured to receive, from the database, performance consumption data of servers of each node of the database and detailed execution duration data of SQL statements at each node, parse the SQL statements to generate execution plan trees, classify and process the SQL statements according to the fingerprints of the execution plan trees, analyze the duration bottlenecks and performance consumption of each type of SQL statements as reference indexes, aggregate and calculate the duration and performance consumption at each execution stage for each type of SQL statements, and visually display the duration bottlenecks and performance consumption of the SQL statements. The analysis processing module is configured to form a preliminary tuning scheme and related parameter sets according to the duration bottlenecks and performance consumption of various types of SQL statements transmitted from the log analysis module, continuously adjust the values of inputted parameter sets, verify the execution results, collect duration and performance consumption data, evaluate optimization effects, and update the parameter prediction model. After multiple comparative tests, the parameter prediction model becomes increasingly refined, and an optimal parameter tuning scheme and an optimal parameter set are ultimately outputted.

[0027] Optionally, in this embodiment, as shown in the implementation flowchart of FIG. 3,

illustratively, the APP inputs each SQL statement of various types of business SQL statements in an SQL statement set one by one into an initial database that has been pre-loaded with baseline data, causing the SQL statements to be executed one by one, to cover as many business scenarios as possible, and repeatedly issues the SQL statements to make the statistical information more accurate. During this process, the database records operation data of each SQL statement, including execution duration, server performance consumption (for example, the CPU, memory, disk I/O, network bandwidth, etc.) of each node in the initial database, and detailed execution duration of the SQL statements at each node, with refinement down to each execution stage. These data constitute a first data set, which is promptly uploaded to the log analysis module. The log analysis module parses each SQL statement to generate an execution plan tree, classifies and aggregates the operation data of the SQL statements according to the fingerprint of the execution plan tree of each SQL statement, and visually displays the duration bottlenecks and performance consumption of the SQL statements. The analysis processing module receives the classified and aggregated operation data of each type of SQL statements and calculates a comprehensive evaluation index value of each type of SQL statements. The index value combines execution duration and performance consumption, providing a quantitative basis for performance evaluation. According to the first data set, the database parameters that have a significant impact on performance bottlenecks are analyzed, and a first parameter set requiring optimization is selected from the complete parameter set. These parameters are usually related to key stages of SQL execution, such as index configuration, cache settings, parallelism, etc. The first parameter set and the comprehensive evaluation index value of each type of SQL statements are inputted into the prediction model. The prediction model optimizes the first parameter set through preliminary analysis and generates a second parameter set after the initial optimization, that is, improvement suggestions based on current performance data. The second parameter set is subjected to multiple rounds of optimization using the prediction model and the initial database to gradually adjust parameter values, and the optimization effects are verified. After each iteration, the prediction model adjusts the optimization direction according to new performance data and finally generates an optimal parameter set.

[0028] Optionally, in this embodiment, based on the comprehensive analysis of SQL execution data and node performance consumption, performance bottlenecks can be accurately identified, for example, high resource consumption of a particular node or execution delays of specific SQL types. By selecting and optimizing the first parameter set, the optimization process focuses on key parameters that have the greatest impact on performance, thereby avoiding the blindness and inefficiency of tuning all parameters. In combination with automated optimization and iterative analysis of the prediction model, the time and workload of manual parameter adjustment are greatly reduced, thereby improving tuning efficiency. Multiple rounds of iterative optimization ensure that the parameter configuration of the database is close to optimal performance to adapt to diverse business scenario requirements, thereby enhancing the overall operational efficiency of the distributed database.

[0029] In some embodiments, the operation where classification and aggregation are performed on the first data set to obtain the operation data of each type of SQL statements includes the following operations.

[0030] An execution plan tree generated by the initial database when executing each SQL statement is acquired.

[0031] All the SQL statements are classified according to the fingerprint of the execution plan tree of each SQL statement to obtain multiple types of SQL statements.

[0032] Operation data of each SQL statement in a target type of SQL statements is aggregated to obtain operation data of the target type of SQL statements, where the target type of SQL statements is any type of SQL statements among the multiple types of SQL statements.

[0033] Optionally, in this embodiment, as shown in the analysis flowchart of the log analysis module in FIG. 4, illustratively, when the database executes each SQL statement, the system generates a corresponding execution plan tree. The execution plan tree describes in detail the execution flow of the SQL statement, including operation steps (for example, index scan, table scan, join operation, etc.) and resource allocation (for example, the CPU, memory, etc.). These plans are critical data structures during the SQL statement execution process and can reflect

execution efficiency and potential bottlenecks. The SQL statements are classified based on the fingerprint of the execution plan tree of each SQL statement. The execution plan tree fingerprint is a feature identifier that uniquely represents the core pattern of the execution logic structure of the SQL statement. Through classification based on fingerprints, SQL statements with similar execution logic may be grouped into one type, forming multiple types of SQL statements. For a target type of SQL statements (any one of the multiple types of SQL statements), the operation data (for example, node performance consumption, execution duration, etc.) of all SQL statements in this type is aggregated. The aggregated results include averages, peak values, distribution statistics, etc., providing an overall picture of the performance of the target type of SQL statements.

[0034] Optionally, in this embodiment, classification based on execution plan trees can precisely focus on the optimization requirements of different SQL types, thereby avoiding "one-size-fits-all" generalized tuning and improving specificity. Aggregation and analysis of SQL statements of the same type can identify patterned performance bottlenecks (for example, ineffective specific indexes, resource contention, etc.), thereby providing data support for optimization. The aggregation process covers multi-dimensional performance data, thereby making the results more representative and facilitating subsequent comprehensive evaluation index calculation and performance optimization scheme design. The classified and aggregated data may provide high-quality input for the subsequent prediction model, thereby enhancing the accuracy of parameter optimization and the effectiveness of iteration.

[0035] In some embodiments, before the comprehensive evaluation index value of each type of SQL statements is calculated according to the operation data of each type of SQL statements, the method further includes the following operation.

[0036] According to an adaptive algorithm, all abnormal values are removed from the first data set to obtain an updated first data set.

[0037] Optionally, in this embodiment, various abnormal values may exist in the first data set, such as glitches and exceptional data information caused by sudden server resource occupation

(interference from other tasks), network fluctuations, hardware failures, etc. These abnormal values may deviate significantly from the normal data distribution and mislead the analysis results. Therefore, before classifying and processing the first data set, the log analysis module uses an adaptive algorithm (for example, a least squares method, a least mean squares algorithm, Wiener filtering, etc.) to dynamically identify and remove abnormal values. The adaptive algorithm can automatically adjust parameters based on statistical features of the data (for example, the distribution and trend), thereby ensuring that abnormal values are removed while normal data is retained. After the abnormal values are removed, a cleaned first data set is obtained, and its data is more authentic and reliable, thereby accurately reflecting the actual performance of database operation.

[0038] Optionally, in this embodiment, after the abnormal values are removed, the data distribution becomes more reasonable, thereby avoiding the interference of abnormal points with statistical results and improving the accuracy of classification and aggregation of operation data. Erroneous decisions that may be caused by abnormal values are eliminated to ensure that comprehensive evaluation indexes and optimization strategies are based on normal operation data, thereby enhancing the overall analysis effect. Data cleaning reduces computational deviations caused by abnormal values, thereby making the direction of subsequent performance optimization clearer and reducing the probability of ineffective optimization.

[0039] In some embodiments, the operation where the comprehensive evaluation index value of each type of SQL statements is calculated according to the operation data of each type of SQL statements includes the following operations.

[0040] Each type of SQL statements is taken as a current type of SQL statements, and the following operations are performed on the current type of SQL statements.

[0041] Performance consumption data and execution duration data of the current type of SQL statements at each execution stage are acquired, and the performance consumption data and the execution duration data are converted into comparable units.

[0042] Weighting factors are assigned to the performance consumption data and the execution

duration data of the current type of SQL statements at each execution stage.

[0043] The performance consumption data and the execution duration data of the current type of SQL statements at each execution stage are respectively multiplied by the corresponding weighting factors to obtain a weighted result of the current type of SQL statements at each execution stage.

[0044] The weighted result of the current type of SQL statements at each execution stage is summed to obtain a comprehensive evaluation index value of the current type of SQL statements.

[0045] Optionally, in this embodiment, as shown in the analysis flowchart of the analysis processing module in FIG. 5, illustratively, for the operation data of each type of SQL statements, performance consumption data (for example, the CPU, memory, disk I/O, network bandwidth, etc.) and execution duration data generated at different execution stages (for example, parsing, optimization, plan generation, execution, etc.) are extracted. These data comprehensively reflect the resource usage and performance of SQL at each stage. The performance consumption data and the execution duration data are converted into comparable units (for example, converted into percentages or normalized to the [0, 1] interval) and standardized to the same dimension (for example, by converting memory consumption into MB/s and duration into milliseconds), thereby maintaining consistency in subsequent calculations and avoiding result distortion due to inconsistent units. According to business requirements or practical scenarios, different weighting factors are assigned to execution stages of the SQL statements. The weighting factors represent the importance of different stages to the comprehensive evaluation of the SQL statements. For example, if the business is more concerned with the execution speed, the weighting factor for execution duration may be higher, whereas for resource-constrained scenarios, the weighting factor for performance consumption may be higher. The performance consumption data and the execution duration data of each type of SQL statements at each execution stage are respectively multiplied by the corresponding weighting factors to obtain a weighted result corresponding to each execution stage. This

process incorporates the importance of different stages, thereby making the evaluation results more aligned with practical business needs. The weighted results of each type of SQL statements at the execution stages are summed to obtain a comprehensive evaluation index value of this type of SQL statements. This index is a quantitative evaluation of the overall performance of SQL and comprehensively reflects the efficiency of execution and the reasonableness of resource usage.

[0046] Optionally, in this embodiment, through unit conversion and weighted calculation, various types of performance data and duration indexes are unified into comparable values so that the evaluation results have standardized features, thereby facilitating horizontal comparison among SQL types. The setting of weighting factors may be flexibly adjusted according to different business scenarios to make the evaluation indexes more suitable for practical requirements, for example, by increasing the emphasis on execution efficiency or resource consumption. The comprehensive evaluation index value can help quickly identify the main performance bottlenecks of a certain type of SQL statements (for example, excessively high resource consumption or excessive duration at a particular stage), thereby providing a clear direction for subsequent optimization. The evaluation index provides a quantitative basis for SQL classification-based optimization and parameter tuning, thereby simplifying the optimization process and improving tuning efficiency and effectiveness.

[0047] In some embodiments, the operation where the complete parameter set of the initial database is analyzed according to the first data set to obtain the first parameter set requiring optimization includes the following operations.

[0048] A bottleneck stage is determined according to the first data set.

[0049] A third parameter set related to the bottleneck stage is selected from the complete parameter set.

[0050] An optimization range is determined for each parameter in the third parameter set.

[0051] The third parameter set and the optimization range for each parameter in the third

parameter set are aggregated to obtain the first parameter set.

[0052] Optionally, in this embodiment, as shown in the analysis flowchart of the analysis processing module in FIG. 5, illustratively, according to the first data set (including performance consumption data and execution duration data of SQL execution), the distribution of time consumption and resource consumption at each execution stage (for example, parsing, optimization, plan generation, execution, etc.) of the database when processing SQL statements is analyzed. By comparing the proportion of time consumption and resource consumption at each stage, the bottleneck stage that has a significant impact on overall performance is identified. For example, if the time consumption of the execution stage is significantly higher than that of other stages or a preset value, the execution stage may be determined as the bottleneck stage. Parameters directly related to the bottleneck stage are selected from the complete parameter set of the initial database to form the third parameter set. For each parameter in the third parameter set, a reasonable optimization range is determined based on the operation data of the first data set and the database parameter design document. For example, the size of the thread pool may have an effective range (for example, 4 to 64 threads). If the adjustment is too small, it may cause insufficient concurrency; if the adjustment is too large, it may consume additional resources. The optimization range may be narrowed according to the actual bottleneck and database load conditions. The selected third parameter set and the optimization range for each parameter in the third parameter set are aggregated to obtain the first parameter set. The first parameter set is a subset of the initial database parameters, is used for iterative parameter optimization, and focuses on parameters related to the bottleneck stage, thereby narrowing the scope of optimization and improving the efficiency of parameter adjustment.

[0053] Optionally, in this embodiment, the analysis of the bottleneck stage identifies the key area of performance issues, thereby focusing on the most critical parameters and avoiding the complexity of complete-parameter adjustment. By selecting related parameters and defining optimization ranges, the parameter set requiring optimization is reduced, thereby significantly lowering the time and computational cost of the tuning process. The first parameter set, which is centered on the bottleneck stage, ensures targeted and effective optimization, thereby achieving

rapid performance improvement. The first parameter set is dynamically generated according to different bottleneck stages so that the optimization process can adapt to different SQL types and operational scenarios, thereby enhancing the tuning flexibility of the distributed database.

[0054] In some embodiments, the operation where multiple iterative optimizations are performed on the second parameter set using the initial database and the prediction model to obtain the optimal parameter set of the initial database includes the following operations.

[0055] A second parameter set and a database used for an M-th optimization ARE acquired, where M is a positive integer greater than 1; when M is 2, the second parameter set used for the M-th optimization is the second parameter set after the initial optimization, and the database used for the M-th optimization is the initial database; when M is greater than 2, the second parameter set used for the M-th optimization is a second parameter set after an (M−1)-th optimization, and the database used for the M-th optimization is a database after the (M−1)-th optimization.

[0056] The database used for the M-th optimization is optimized according to the second parameter set used for the M-th optimization to obtain a database after the M-th optimization.

[0057] The comprehensive evaluation index value of each type of SQL statements is recalculated based on the SQL statement set and the database after the M-th optimization to obtain a comprehensive evaluation index value of each type of SQL statements after the M-th optimization.

[0058] All the comprehensive evaluation index values after the M-th optimization and the second parameter set used for the M-th optimization inputted into the prediction model so that the prediction model optimizes the second parameter set used for the M-th optimization according to all the comprehensive evaluation index values after the M-th optimization to obtain a second parameter set after the M-th optimization, M is updated to (M+1), and an (M+1)-th optimization is performed until M is equal to a target threshold to obtain the optimal parameter set.

[0059] Optionally, in this embodiment, the input for the first optimization is the initially optimized second parameter set and the initial database. As the iterative optimizations proceed, the input for the M-th optimization includes the second parameter set generated by the (M−1)-th optimization and the database instance generated by the (M−1)-th optimization. According to the second parameter set used for the M-th optimization, the database for the M-th optimization is adjusted, including modifying configuration parameters (for example, parallelism, caching strategies, etc.) and restarting or hot-updating the database, to generate a database instance after the M-th optimization. The database after the M-th optimization is tested using the SQL statement set, multiple types of SQL statements are executed, performance data (time consumption, CPU, memory, disk I/O, etc.) of each type of SQL statements is collected, and the comprehensive evaluation index value after the M-th optimization is recalculated according to the weighting factors. The comprehensive evaluation index value after the M-th optimization and the current second parameter set are inputted into the prediction model. The prediction model analyzes the trend of changes in the new index value, further adjusts the parameter set, and generates a second parameter set for the next round of optimization. After each optimization is completed, the iteration number M is incremented by 1, and the next round of optimization proceeds with the new parameter set and database instance. The reasonableness of the current parameter tuning is manually monitored and analyzed. If some parameters are deemed unreasonable, manual intervention may be performed to adjust the parameters, and then the parameter prediction model is modified to make the prediction model better suit application requirements. When M reaches a target threshold (for example, when performance stabilizes or no significant improvement occurs), the iteration is terminated, and the final optimal parameter set is outputted.

[0060] Optionally, in this embodiment, through multiple iterative optimizations, each iteration uses more accurate index values to guide the next round of parameter adjustment, thereby achieving a progressive approximation to the optimal parameters. The prediction model can adjust strategies based on historical optimization data to adapt to different SQL workload features and changes in database architecture. The finally outputted optimal parameter set can

balance resource utilization and SQL execution efficiency, thereby significantly improving database performance while ensuring stability. Automated iterative optimization reduces the complexity of manual parameter analysis, thereby improving optimization efficiency and precision.

[0061] In some embodiments, the operation where the comprehensive evaluation index value of each type of SQL statements is recalculated based on the SQL statement set and the database after the M-th optimization to obtain the comprehensive evaluation index value of each type of SQL statements after the M-th optimization includes the following operations.

[0062] Each SQL statement in the SQL statement set is inputted into the database after the M-th optimization so that the database after the M-th optimization executes each SQL statement, and operation data of each SQL statement is outputted to obtain a first data set after the M-th optimization.

[0063] Classification and aggregation are performed on the first data set after the M-th optimization to obtain operation data of each type of SQL statements, and the comprehensive evaluation index value of each type of SQL statements after the M-th optimization is calculated according to the operation data of each type of SQL statements.

[0064] Optionally, in this embodiment, each SQL statement in the SQL statement set is inputted one by one into the database after the M-th optimization for execution. During the execution of the SQL statements, the database records operation data, including performance consumption (the CPU, memory, disk I/O, etc.) of each node and execution duration of the SQL statements at each node, and outputs the operation data to form the first data set after the M-th optimization. The first data set after the M-th optimization is subjected to classification according to the fingerprint of the execution plan tree of each SQL statement, all the SQL statements are classified into multiple types, and the operation data (duration and performance consumption) of each type of SQL statements is aggregated to obtain operation data of each type of SQL statements. For the operation data of each type of SQL statements, the performance consumption data and execution duration data at execution stages are extracted and subjected to

unit normalization (for example, converted to milliseconds or percentages). The performance consumption data and the execution duration data at each execution stage are respectively multiplied by corresponding weighting factors to obtain a weighted result corresponding to each execution stage, and weighted results at all the execution stages are summed to calculate the comprehensive evaluation index value of the current type of SQL statements. The above process is repeated to calculate the comprehensive evaluation index values of all types of SQL statements, and then a complete set of comprehensive evaluation indexes after the M-th optimization is obtained.

[0065] Optionally, the core of the present application lies in utilizing the duration logs and server performance consumption information of the database for data preprocessing, classification, aggregation, computation, and iterative optimization to finally obtain an optimal parameter set and parameter values suitable for the database. The SQL feature classification may also be flexibly adjusted according to other rules, such as specified HINT information (that is, SQL hints, which are special instructions used to guide the database in generating SQL execution plans), pre-compiled templating features, and the like. The prediction model may adopt artificial intelligence algorithms. There are two alternative approaches: model types: supervised learning (for example, regression analysis), unsupervised learning (for example, clustering), semi-supervised learning, reinforcement learning, etc.; and training algorithms: combining different algorithms such as regression algorithms, decision trees, deep learning, association rule learning, etc., to select a suitable optimization method.

[0066] For the foregoing method embodiments, for ease of description, the method embodiments are all expressed as a series of action combinations, but those skilled in the art should know that the present application is not limited to the described action sequence because certain steps may adopt other sequences or be performed simultaneously according to the present application. Furthermore, those skilled in the art should also appreciate that all the embodiments described in the specification are example embodiments, and the related actions and modules are not necessarily mandatory to the present application.

[0067] Compared with the related art, the above solutions provided by the embodiments of the present application have the features described below.

[0068] The present application adopts a method, which includes: each SQL statement in an SQL statement set is inputted into an initial database so that the initial database executes each SQL statement, and operation data of each SQL statement is outputted to obtain a first data set, where the SQL statement set includes multiple types of SQL statements, and the operation data includes performance consumption data of a server of each node of the initial database and execution duration data of each SQL statement at each node when the initial database executes each SQL statement; classification and aggregation are performed on the first data set to obtain operation data of each type of SQL statements, and a comprehensive evaluation index value of each type of SQL statements is calculated according to the operation data of each type of SQL statements; a complete parameter set of the initial database is analyzed according to the first data set to obtain a first parameter set requiring optimization; all the comprehensive evaluation index values and the first parameter set are inputted into a prediction model so that the prediction model performs an initial optimization on the first parameter set according to all the comprehensive evaluation index values to obtain a second parameter set after the initial optimization; multiple iterative optimizations are performed on the second parameter set using the initial database and the prediction model to obtain an optimal parameter set of the initial database. In the above-described method for optimizing performance parameters of a database, each SQL statement in the SQL statement set is inputted into the initial database for execution, and operation data including detailed execution duration and server performance consumption at each node is collected to form a first data set. After the data is subjected to classification and aggregation, a comprehensive evaluation index value of each type of SQL statements is calculated, and a first parameter set requiring optimization is selected from the complete parameter set. Subsequently, all the comprehensive evaluation index values and the first parameter set are inputted into a prediction model to generate a second parameter set after initial optimization, and through multiple rounds of iterative adjustment, an optimal parameter set is finally obtained. Therefore, a comprehensive collection of performance data and SQL duration

is achieved, performance bottlenecks are accurately identified, critical parameters are focused on, blind tuning is avoided, and, in combination with automated parameter adjustment by the prediction model, tuning efficiency is improved. Multiple rounds of iterative optimization ensure continuous improvement of database performance, meet the requirements of complex business scenarios, and significantly reduce the tuning difficulty under the distributed architecture. Ultimately, the problem of great difficulty in parameter tuning caused by the complexity of the distributed database architecture is solved.

[0069] According to another aspect of the embodiments of the present application, an apparatus for optimizing performance parameters of a database is further provided. As shown in FIG. 6, the apparatus includes an execution module 602, a calculation module 604, an analysis module 606, a first optimization module 608, and a second optimization module 610.

[0070] The execution module 602 is configured to input each SQL statement in an SQL statement set into an initial database so that the initial database executes each SQL statement, and output operation data of each SQL statement to obtain a first data set, where the SQL statement set includes multiple types of SQL statements, and the operation data includes performance consumption data of a server of each node of the initial database and execution duration data of each SQL statement at each node when the initial database executes each SQL statement.

[0071] The calculation module 604 is configured to perform classification and aggregation on the first data set to obtain operation data of each type of SQL statements, and calculate a comprehensive evaluation index value of each type of SQL statements according to the operation data of each type of SQL statements.

[0072] The analysis module 606 is configured to analyze a complete parameter set of the initial database according to the first data set to obtain a first parameter set requiring optimization.

[0073] The first optimization module 608 is configured to input all the comprehensive evaluation index values and the first parameter set into a prediction model so that the prediction

model performs an initial optimization on the first parameter set according to all the comprehensive evaluation index values to obtain a second parameter set after the initial optimization.

[0074] The second optimization module 610 is configured to perform multiple iterative optimizations on the second parameter set using the initial database and the prediction model to obtain an optimal parameter set of the initial database.

[0075] The execution module 602 in this embodiment may be configured to execute step S102 in the embodiments of the present application, the calculation module 604 in this embodiment may be configured to execute step S104 in the embodiments of the present application, the analysis module 606 in this embodiment may be configured to execute step S106 in the embodiments of the present application, the first optimization module 608 in this embodiment may be configured to execute step S108 in the embodiments of the present application, and the second optimization module 610 in this embodiment may be configured to execute step S110 in the embodiments of the present application.

[0076] In some embodiments, the calculation module includes a first acquisition unit, a classification unit, and a first aggregation unit.

[0077] The first acquisition unit is configured to acquire an execution plan tree generated by the initial database when executing each SQL statement.

[0078] The classification unit is configured to classify all the SQL statements according to the fingerprint of the execution plan tree of each SQL statement to obtain multiple types of SQL statements.

[0079] The first aggregation unit is configured to aggregate operation data of each SQL statement in a target type of SQL statements to obtain operation data of the target type of SQL statements, where the target type of SQL statements is any type of SQL statements among the multiple types of SQL statements.

[0080] In some embodiments, the apparatus further includes a removal module.

[0081] The removal module is configured to, before the comprehensive evaluation index value of each type of SQL statements is calculated according to the operation data of each type of SQL statements, remove, according to an adaptive algorithm, all abnormal values from the first data set to obtain an updated first data set.

[0082] In some embodiments, the calculation module includes a processing unit.

[0083] The processing unit is configured to take each type of SQL statements as a current type of SQL statements and perform the following operations on the current type of SQL statements.

[0084] Performance consumption data and execution duration data of the current type of SQL statements at each execution stage are acquired, and the performance consumption data and the execution duration data are converted into comparable units.

[0085] Weighting factors are assigned to the performance consumption data and the execution duration data of the current type of SQL statements at each execution stage.

[0086] The performance consumption data and the execution duration data of the current type of SQL statements at each execution stage are respectively multiplied by the corresponding weighting factors to obtain a weighted result of the current type of SQL statements at each execution stage.

[0087] The weighted result of the current type of SQL statements at each execution stage is summed to obtain a comprehensive evaluation index value of the current type of SQL statements.

[0088] In some embodiments, the analysis module includes a first determination unit, a selection unit, a second determination unit, and a second aggregation unit.

[0089] The first determination unit is configured to determine a bottleneck stage according to the first data set.

[0090] The selection unit is configured to select a third parameter set related to the bottleneck

stage from the complete parameter set.

[0091] The second determination unit is configured to determine an optimization range for each parameter in the third parameter set.

[0092] The second aggregation unit is configured to aggregate the third parameter set and the optimization range for each parameter in the third parameter set to obtain the first parameter set.

[0093] In some embodiments, the second optimization module includes a second acquisition unit, a first optimization unit, a calculation unit, and a second optimization unit.

[0094] The second acquisition unit is configured to acquire a second parameter set and a database used for an M-th optimization, where M is a positive integer greater than 1; when M is 2, the second parameter set used for the M-th optimization is the second parameter set after the initial optimization, and the database used for the M-th optimization is the initial database; when M is greater than 2, the second parameter set used for the M-th optimization is a second parameter set after an (M−1)-th optimization, and the database used for the M-th optimization is a database after the (M−1)-th optimization.

[0095] The first optimization unit is configured to optimize the database used for the M-th optimization according to the second parameter set used for the M-th optimization to obtain a database after the M-th optimization.

[0096] The calculation unit is configured to recalculate the comprehensive evaluation index value of each type of SQL statements based on the SQL statement set and the database after the M-th optimization to obtain a comprehensive evaluation index value of each type of SQL statements after the M-th optimization.

[0097] The second optimization unit is configured to input all the comprehensive evaluation index values after the M-th optimization and the second parameter set used for the M-th optimization into the prediction model so that the prediction model optimizes the second parameter set used for the M-th optimization according to all the comprehensive evaluation index values after the M-th optimization to obtain a second parameter set after the M-th

optimization, update M to (M+1), and perform an (M+1)-th optimization until M is equal to a target threshold to obtain the optimal parameter set.

[0098] In some embodiments, the calculation unit includes an execution sub-unit and a calculation sub-unit.

[0099] The execution sub-unit is configured to input each SQL statement in the SQL statement set into the database after the M-th optimization so that the database after the M-th optimization executes each SQL statement, and output operation data of each SQL statement to obtain a first data set after the M-th optimization.

[0100] The calculation sub-unit is configured to perform classification and aggregation on the first data set after the M-th optimization to obtain operation data of each type of SQL statements, and calculate the comprehensive evaluation index value of each type of SQL statements after the M-th optimization according to the operation data of each type of SQL statements.

[0101] For other examples in this embodiment, reference may be made to the foregoing examples, and the details are not repeated here.

[0102] FIG. 7 is a schematic diagram of an electronic device according to an embodiment of the present application. As shown in FIG. 7, the electronic device includes a processor 702, a communication interface 704, a memory 706, and a communication bus 708, where the processor 702, the communication interface 704, and the memory 706 communicate with each other through the communication bus 708.

[0103] The memory 706 is configured to store a computer program.

[0104] The processor 702 is configured to perform the following operations when executing the computer program stored in the memory.

[0105] Each SQL statement in an SQL statement set is inputted into an initial database so that the initial database executes each SQL statement, and operation data of each SQL statement is outputted to obtain a first data set, where the SQL statement set includes multiple types of SQL

statements, and the operation data includes performance consumption data of a server of each node of the initial database and execution duration data of each SQL statement at each node when the initial database executes each SQL statement.

[0106] Classification and aggregation are performed on the first data set to obtain operation data of each type of SQL statements, and a comprehensive evaluation index value of each type of SQL statements is calculated according to the operation data of each type of SQL statements.

[0107] A complete parameter set of the initial database is analyzed according to the first data set to obtain a first parameter set requiring optimization.

[0108] All the comprehensive evaluation index values and the first parameter set are inputted into a prediction model so that the prediction model performs an initial optimization on the first parameter set according to all the comprehensive evaluation index values to obtain a second parameter set after the initial optimization.

[0109] Multiple iterative optimizations are performed on the second parameter set using the initial database and the prediction model to obtain an optimal parameter set of the initial database.

[0110] Optionally, in this embodiment, the communication bus may be a peripheral component interconnect (PCI) bus or an extended industry standard architecture (EISA) bus. The communication bus may be divided into an address bus, a data bus, a control bus, and the like. For ease of representation, the communication bus is represented with only one thick line in FIG. 7, but this does not mean that there is only one bus or one type of bus. The communication interface is used for communication between the electronic device and other devices.

[0111] The memory may include a random-access memory (RAM) or a non-volatile memory, for example, at least one disk memory. Optionally, the memory may also be at least one storage apparatus located remotely from the processor.

[0112] As some examples, the memory 706 may include the execution module 602, the

calculation module 604, the analysis module 606, the first optimization module 608, and the second optimization module 610 of the apparatus for optimizing performance parameters of a database. In addition, the memory 706 may also include other module units in the apparatus for optimizing performance parameters of a database.

[0113] The processor may be a general-purpose computer, including, for example, a central processing unit (CPU), a network processor (NP), and the like; may also be a digital signal processor (DSP), an application-specific integrated circuit (ASIC), a field-programmable gate array (FPGA) or any other programmable logic device, a discrete gate or transistor logic device, or a discrete hardware component.

[0114] Optionally, for example, in this embodiment, reference may be made to the examples described in the foregoing embodiments, and the details are not repeated here.

[0115] Those of ordinary skill in the art may understand that the configuration shown in FIG. 7 is only illustrative, and the device for implementing the method for optimizing performance parameters of a database may be a terminal device, which may be a smart phone (such as an Android mobile phone, an iPhone Operating System (iOS) mobile phone, or the like), a tablet computer, a handheld computer, a mobile Internet device (MID) or a portable android device (PAD). FIG. 7 is not intended to limit the structure of the electronic device. For example, the electronic device may also include more or fewer components (for example, a network interface, a display device, etc.) than the components shown in FIG. 7, or have a configuration different from the configuration shown in FIG. 7.

[0116] Those of ordinary skill in the art may understand that all or part of the steps in the various methods in the foregoing embodiments may be accomplished by means of a program that instructs hardware associated with the terminal device, and the program may be stored in a computer-readable storage medium, which may include a flash memory disk, a read-only memory (ROM), a RAM, a magnetic disc, or an optical disc.

[0117] According to yet another aspect of the present application, a computer-readable storage medium is further provided. The computer-readable storage medium stores a computer

program which, when executed by a processor, causes the processor to perform the steps of the method for optimizing performance parameters of a database described above.

[0118] Optionally, in this embodiment, those of ordinary skill in the art may understand that all or part of the steps in the various methods in the foregoing embodiments may be accomplished by means of a program that instructs hardware associated with a terminal device, and the program may be stored in a computer-readable storage medium, which may include a flash memory disk, a ROM, a RAM, a magnetic disc, or an optical disc.

[0119] If the integrated units in the foregoing embodiments are implemented in the form of software functional units and sold or used as independent products, these units may be stored in the above computer-readable storage medium. Based on this understanding, the essence of the technical solutions of the present application, or the part that contributes to the related art, or all or part of the technical solutions, may be embodied in the form of a software product. The computer software product is stored in a storage medium and includes several instructions for causing one or more computer devices (which may be personal computers, servers, network devices, etc.) to perform all or part of the steps of the methods described in the various embodiments of the present application.

[0120] In the foregoing embodiments of the present application, the description of each embodiment has its own emphasis. For parts that are not described in detail in some embodiments, reference may be made to the relevant descriptions in other embodiments.

[0121] In the several embodiments provided in the present application, it should be understood that the disclosed client may be implemented in other ways. The apparatus embodiments described above are only illustrative. For example, the division of units is only a logical functional division, and in practical implementation, there may be other division manners. For example, multiple units or components may be combined or integrated into another system, or some features may be omitted or not performed. In addition, the mutual coupling or direct coupling or communication connections shown or discussed may be implemented by using some interfaces, and the indirect coupling or communication connections

between units or modules may be in an electrical form or another form.

[0122] The units described as separate components may or may not be physically separated, and the components shown as units may or may not be physical units, that is, they may be located in one place or distributed over multiple network units. Part or all of the units may be selected according to practical requirements to achieve the objects of the solutions of the embodiments.

[0123] In addition, the functional units in the various embodiments of the present application may be integrated into one processing unit, or each unit may exist separately physically, or two or more units may be integrated into one unit. The above integrated units may be implemented in the form of hardware or in the form of software functional units.

What is claimed is:

1. A method for optimizing performance parameters of a database, comprising:

inputting each structured query language (SQL) statement in an SQL statement set into an initial database so that the initial database executes each SQL statement and outputs operation data of each SQL statement to obtain a first data set; wherein the SQL statement set comprises a plurality of types of SQL statements, and the operation data comprises performance consumption data of a server of each node of the initial database and execution duration data of each SQL statement at each node of the initial database during the executing of each SQL statement by the initial database;

performing classification and aggregation on the first data set to obtain operation data of each type of SQL statements of the plurality of types of SQL statements, and calculating a comprehensive evaluation index value of each type of SQL statements according to the operation data of each type of SQL statements;

analyzing a complete parameter set of the initial database according to the first data set to obtain a first parameter set requiring optimization;

inputting comprehensive evaluation index values of the plurality of types of SQL statements and the first parameter set into a prediction model so that the prediction model performs an initial optimization on the first parameter set according to the comprehensive evaluation index values of the plurality of types of SQL statements to obtain a second parameter set after the initial optimization; and

performing a plurality of iterative optimizations on the second parameter set using the initial database and the prediction model to obtain an optimal parameter set of the initial database.

2. The method according to claim 1, wherein performing classification and aggregation on the first data set to obtain the operation data of each type of SQL statements of the plurality of types of SQL statements comprises:

for each SQL statement, acquiring an execution plan tree generated by the initial database during the executing of the SQL statement;

classifying all SQL statements in the SQL statement set according to a fingerprint of the execution plan tree of each SQL statement to obtain the plurality of types of SQL statements; and

aggregating operation data of each SQL statement in a target type of SQL statements to

obtain operation data of the target type of SQL statements, wherein the target type of SQL statements is any type of SQL statements of the plurality of types of SQL statements.

3. The method according to claim 1, before calculating the comprehensive evaluation index value of each type of SQL statements according to the operation data of each type of SQL statements, further comprising:

according to an adaptive algorithm, removing all abnormal values from the first data set to obtain an updated first data set.

4. The method according to claim 1, wherein calculating the comprehensive evaluation index value of each type of SQL statements according to the operation data of each type of SQL statements comprises:

taking each type of SQL statements as a current type of SQL statements, and performing following operations on the current type of SQL statements:

acquiring performance consumption data and execution duration data of the current type of SQL statements at each execution stage, and converting the performance consumption data and the execution duration data into comparable units;

assigning weighting factors to the performance consumption data and the execution duration data of the current type of SQL statements at each execution stage;

respectively multiplying the performance consumption data and the execution duration data of the current type of SQL statements at each execution stage by the corresponding weighting factors to obtain a weighted result of the current type of SQL statements at each execution stage; and

summing weighted results of the current type of SQL statements at all execution stages to obtain a comprehensive evaluation index value of the current type of SQL statements.

5. The method according to claim 1, wherein analyzing the complete parameter set of the initial database according to the first data set to obtain the first parameter set requiring optimization comprises:

determining a bottleneck stage according to the first data set;

selecting a third parameter set related to the bottleneck stage from the complete parameter set;

determining an optimization range for each parameter in the third parameter set; and

aggregating the third parameter set and the optimization range for each parameter in the third parameter set to obtain the first parameter set.

6. The method according to claim 1, wherein performing the plurality of iterative optimizations on the second parameter set using the initial database and the prediction model to obtain the optimal parameter set of the initial database comprises:

acquiring a second parameter set and a database used for an M-th optimization, wherein M is a positive integer greater than 1; when M is 2, the second parameter set used for the M-th optimization is the second parameter set after the initial optimization, and the database used for the M-th optimization is the initial database; when M is greater than 2, the second parameter set used for the M-th optimization is a second parameter set after an (M−1)-th optimization, and the database used for the M-th optimization is a database after the (M−1)-th optimization;

optimizing the database used for the M-th optimization according to the second parameter set used for the M-th optimization to obtain a database after the M-th optimization;

recalculating the comprehensive evaluation index value of each type of SQL statements based on the SQL statement set and the database after the M-th optimization to obtain a comprehensive evaluation index value of each type of SQL statements after the M-th optimization; and

inputting comprehensive evaluation index values of the plurality of types of SQL statements after the M-th optimization and the second parameter set used for the M-th optimization into the prediction model so that the prediction model optimizes the second parameter set used for the M-th optimization according to the comprehensive evaluation index values of the plurality of types of SQL statements after the M-th optimization to obtain a second parameter set after the M-th optimization, updating M to (M+1), and performing an (M+1)-th optimization until M is equal to a target threshold to obtain the optimal parameter set.

7. The method according to claim 6, wherein recalculating the comprehensive evaluation index value of each type of SQL statements based on the SQL statement set and the database after the M-th optimization to obtain the comprehensive evaluation index value of each type of SQL statements after the M-th optimization comprises:

inputting each SQL statement in the SQL statement set into the database after the M-th optimization so that the database after the M-th optimization executes each SQL statement, and outputting operation data of each SQL statement to obtain a first data set after the M-th

optimization; and

performing classification and aggregation on the first data set after the M-th optimization to obtain operation data of each type of SQL statements, and calculating the comprehensive evaluation index value of each type of SQL statements after the M-th optimization according to the operation data of each type of SQL statements.

8. An apparatus for optimizing performance parameters of a database, comprising:

an execution module, configured to input each structured query language (SQL) statement in an SQL statement set into an initial database so that the initial database executes each SQL statement and outputs operation data of each SQL statement to obtain a first data set; wherein the SQL statement set comprises a plurality of types of SQL statements, and the operation data comprises performance consumption data of a server of each node of the initial database and execution duration data of each SQL statement at each node of the initial database during the executing of each SQL statement by the initial database;

a calculation module, configured to perform classification and aggregation on the first data set to obtain operation data of each type of SQL statements of the plurality of types of SQL statements, and calculate a comprehensive evaluation index value of each type of SQL statements according to the operation data of each type of SQL statements;

an analysis module, configured to analyze a complete parameter set of the initial database according to the first data set to obtain a first parameter set requiring optimization;

a first optimization module, configured to input comprehensive evaluation index values of the plurality of types of SQL statements and the first parameter set into a prediction model so that the prediction model performs an initial optimization on the first parameter set according to the comprehensive evaluation index values of the plurality of types of SQL statements to obtain a second parameter set after the initial optimization; and

a second optimization module, configured to perform a plurality of iterative optimizations on the second parameter set using the initial database and the prediction model to obtain an optimal parameter set of the initial database.

9. A computer-readable storage medium, wherein the computer-readable storage medium stores a computer program which, when executed by a processor, causes the processor to perform the method for optimizing the performance parameters of the database according to any one of claims 1 to 7.

10. An electronic device, comprising a memory and a processor, wherein the memory stores a computer program, and the processor is configured to perform the method for optimizing the performance parameters of the database according to any one of claims 1 to 7 through the computer program.