

DESCRIPTION

TITLE OF INVENTION: DATA PROCESSING METHOD, SYSTEM, APPARATUS, DEVICE, AND STORAGE MEDIUM

The present disclosure claims priority of Chinese Patent Application No. 202310595005.2, filed to China National Intellectual Property Administration on May 23, 2023 and titled “DATA PROCESSING METHOD, SYSTEM, APPARATUS, DEVICE, AND STORAGE MEDIUM”, the content of which is hereby incorporated by reference in its entirety.

TECHNICAL FIELD

The present disclosure relates to the technical field of big data, and particularly, to a data processing method, a system, an apparatus, a device, and a storage medium.

BACKGROUND OF THE INVENTION

In the field of big data processing, a disaggregated storage and computing architecture is widely employed, where data computation and storage are deployed on different nodes, and data is commonly stored in a columnar storage format such as Parquet.

In practical application, there is often an actual requirement to read large volumes of data and perform computations (such as aggregation operations including calculating averages and sums, or filtering operations including numerical filtering and text filtering). However, due to the limited network bandwidth between a computing side and a storage side, both transmission efficiency and computation efficiency are often low.

SUMMARY OF THE INVENTION

Embodiments of the present disclosure provide a data processing method, a system, an apparatus, a device, and a storage medium, to improve data transmission efficiency and computational performance between a computing node and a storage node.

In a first aspect, the embodiments of the present disclosure provide a data processing method, which is applied to an acceleration node located between a computing node and a storage node. The data processing method includes the following steps.

A query instruction sent by the computing node is received, the query instruction including: identification information of target data and a computation instruction to be executed on the target data.

The target data is acquired according to the identification information of the target data.

The target data is processed according to the computation instruction, to obtain a computation result.

The computation result is sent to the computing node.

In a second aspect, the embodiments of the present disclosure provide a data processing system. The data processing system includes:

a computing node cluster, an acceleration node cluster, and a storage node cluster, the acceleration node cluster being coupled between the computing node cluster and the storage node cluster; and

any acceleration node in the acceleration node cluster being configured to perform the data processing method in the first aspect.

In a third aspect, the embodiments of the present disclosure provide a data processing apparatus, which is applied to an acceleration node located between a computing node and a storage node. The data processing apparatus includes:

a receiving module, configured to receive a query instruction sent by the computing node, the query instruction including: identification information of target data and a computation instruction to be executed on the target data;

a processing module, configured to acquire the target data according to the identification information of the target data; and process the target data according to the computation instruction, to obtain a computation result; and

a sending module, configured to send the computation result to the computing node.

In a fourth aspect, the embodiments of the present disclosure provides an electronic device, which includes: a memory, a processor, and a communication interface, where the memory stores executable code, and when the executable code is executed by the processor, the

executable code causes the processor to at least implement the data processing method in the first aspect.

In a fifth aspect, the embodiments of the present disclosure provide a non-transitory machine-readable storage medium. The non-transitory machine-readable storage medium stores executable code, and when the executable code is executed by a processor of an electronic device, the executable code causes the processor to at least implement the data processing method in the first aspect.

According to the solution provided in the embodiments of the present disclosure, the acceleration node is deployed between the computing node and the storage node. When the computing node needs to query the storage node for the target data and perform a corresponding computation (such as filtering or aggregation) on the target data, the acceleration node first receives the query instruction sent by the computing node, where the query instruction includes: the identification information of the target data and the computation instruction to be executed on the target data; then, acquires the target data according to the identification information of the target data, and processes the target data according to the computation instruction, to obtain the computation result; and finally, sends the computation result to the computing node. With this solution, the computation instruction that originally needs to be executed on the computing node is pushed down to the acceleration node for execution, and the acceleration node transmits the computation result to the computing node. In this way, the volume of data transmitted to the computing node can be reduced to improve data transmission efficiency; and moreover, resource usage pressure on the computing node can be reduced to improve the response speed of query services.

BRIEF DESCRIPTION OF DRAWINGS

To describe the technical solutions in the embodiments of the present disclosure more clearly, the following briefly introduces the accompanying drawings required for describing the embodiments. Apparently, the accompanying drawings in the following description show some embodiments of the present disclosure, and a person having ordinary skill in the art may still derive other drawings from these accompanying drawings without creative efforts.

FIG. 1 is an architectural schematic diagram of a data processing system according to some embodiments of the present disclosure.

FIG. 2 is an architectural schematic diagram of another data processing system according to some embodiments of the present disclosure.

FIG. 3 is a flowchart of a data processing method according to some embodiments of the present disclosure.

FIG. 4 is a flowchart of another data processing method according to some embodiments of the present disclosure.

FIG. 5 is a flowchart of still another data processing method according to some embodiments of the present disclosure.

FIG. 6 is a structural schematic diagram of an acceleration node according to some embodiments of the present disclosure.

FIG. 7 is a structural schematic diagram of a data processing apparatus according to some embodiments of the present disclosure.

FIG. 8 is a structural schematic diagram of an electronic device corresponding to the data processing apparatus according to some embodiments shown in FIG. 7.

DETAILED DESCRIPTION OF THE INVENTION

To make the objectives, technical solutions, and advantages of the present disclosure clearer, the technical solutions in the embodiments of the present disclosure will be described clearly and completely below with reference to the drawings in the embodiments of the present disclosure. Apparently, the described embodiments are some of but not all of the embodiments of the present disclosure. All other embodiments derived by a person having ordinary skill in the art from the embodiments in the present disclosure without creative effort still fall within the scope of protection of the present disclosure.

It should be noted that user information (including but not limited to user device information, user personal information, and the like) and data (including but not limited to data for analysis, stored data, displayed data, and the like) that are involved in the embodiments of the present disclosure are all information and data that are authorized by users or fully authorized by all parties. Furthermore, collection, use, and processing of relevant data need to comply with the relevant laws and regulations and standards in the relevant countries and regions, and corresponding operation interfaces are provided to allow users to grant or refuse authorization.

In addition, the sequence of steps in the following method embodiments is an example, and should not be construed as limitations.

FIG. 1 is an architectural schematic diagram of a data processing system according to some embodiments of the present disclosure. As shown in FIG. 1, the data processing system includes: a computing node cluster, an acceleration node cluster, and a storage node cluster. The acceleration node cluster is coupled between the computing node cluster and the storage node cluster.

The computing node cluster includes N (N is greater than or equal to 1) different computing nodes. Computing resources, such as a central processing unit (CPU), and certain storage resources are primarily deployed in each computing node, and the computing node is configured to execute various computation tasks such as filtering and aggregation of data.

The storage node cluster includes P (P is greater than or equal to 1) different storage nodes. Storage resources, such as a disk, and certain computing resources are primarily deployed in each storage node, and the storage node is configured to store data persistently. When stored, data is stored in a columnar storage format or is stored in a row storage format.

The acceleration node cluster includes M (M is greater than or equal to 1) acceleration nodes. Each acceleration node correspondingly has a computing resource and a storage resource, and is configured to perform data pre-processing on target data queried by a computing node based on a query instruction sent by the computing node, and feed back an obtained processing result with a small volume of data to the computing node, to accelerate data transmission. The data pre-processing process will be described in detail in the following embodiments, and the architecture of the data processing system is primarily described here.

Values of N , P , and M are the same or are different. For ease of description, the following embodiments are described from the perspective of a computing node, a storage node, and an acceleration node, which are respectively any node in a corresponding cluster.

It should be noted that in some embodiments, the acceleration node is located between the computing node and the storage node, which is intended to emphasize that data queries between the computing node and the storage node pass through the acceleration node, rather

than to define corresponding hardware deployment of the computing node, the acceleration node, and the storage node.

In a specific implementation process, the acceleration node in some embodiments is regarded as a software component that can be deployed on a node having a storage resource and a computing resource and that is deployed in a variety of manners.

For example, in a disaggregated storage and computing architecture, the acceleration node is deployed in a manner shown in FIG. 1, that is, deployed on a node cluster other than the computing node cluster and the storage node cluster, and accelerates data transmission by using a storage resource and a computing resource of the node cluster other than the computing node cluster and the storage node cluster.

If there is no another node cluster available for deploying the acceleration node, the acceleration node is deployed on the node cluster to which the storage node belongs, that is, the acceleration node and the storage node are deployed on the same node cluster. As shown in FIG. 2, FIG. 2 is an architectural schematic diagram of another data processing system according to some embodiments of the present disclosure. In a case shown in FIG. 2, the storage node and the acceleration node are both regarded as software components. For example, in practical application, the storage node is a Hadoop Distributed File System (HDFS). The hybrid deployment of the acceleration node and the storage node can not only fully utilize the storage resource of the node, but also fully utilize the computing resource of the node to provide an acceleration service. In this way, data transmission between the storage node and the computing node is accelerated.

Optionally, the acceleration node is alternatively deployed on the storage node, that is, acceleration and storage are deployed on the same node. For example, for the P acceleration nodes included in the storage node cluster, an acceleration node 1 is deployed on a storage node 1, and an acceleration node 2 is deployed on a storage node 2.

The acceleration node deployed in any of the foregoing manners can accelerate transmission by reducing the volume of data transmitted to the computing node. FIG. 1 is taken as an example. After the acceleration node provides an acceleration service, a data query process between a computing node 1 and a storage node 1 is changed from “computing node 1 → storage node 1 → computing node 1” in the absence of an acceleration node 1 to “computing

node 1 → acceleration node 1 → storage node 1 → acceleration node 1 → computing node 1”. The acceleration node 1 is configured to pre-process target data acquired from the storage node 1, to reduce the volume of data that needs to be transmitted to the computing node 1.

In practical application, different computing nodes correspond to different computing frameworks. For example, a computing framework corresponding to a computing node 1 is Spark, a computing framework corresponding to a computing node 2 is Hadoop, and a computing framework corresponding to a computing node 3 is Presto. According to different computing frameworks corresponding to computing nodes, the computing nodes are classified into different types. To ensure compatibility between different types of computing nodes and the acceleration node, data reading plugins corresponding to a plurality of types of computing nodes are provided.

Optionally, the data processing system shown in FIG. 1 further includes: a control node, configured to determine, according to a type of a target computing node in the computing node cluster, a data reading plugin matching the type of the target computing node, and deploy the data reading plugin in the target computing node. The acceleration nodes in the acceleration node cluster provide a unified service invocation interface. The service invocation interface supports a preset data transmission protocol, where the set data transmission protocol defines a data format for inputs and outputs. The computing node communicates with the acceleration node by invoking the service invocation interface in the acceleration node via the data reading plugin deployed in the computing node, such as issuing a query instruction.

In practical application, if there is no data reading plugin matching the type of the computing node, a user develops a data reading plugin matching a type of a computing node of the user according to the unified service invocation interface in the acceleration node.

Optionally, a corresponding load balancing algorithm is set between the computing node cluster and the acceleration node cluster. The query instruction sent by the computing node is sent to any acceleration node in the acceleration node cluster based on the load balancing algorithm. Communication between the computing node 1 and the acceleration node 1 in FIG. 1 is merely an example, and in practical application, communication between the computing node and the acceleration node is not fixed. For example, if an acceleration node x is in an idle state at time t1, a query instruction 1 sent by a target computing node at the time t1 is sent to

the acceleration node x based on the load balancing algorithm, if an acceleration node y is in an idle state at time t2, a query instruction 2 sent by the target computing node at the time t2 is sent to the acceleration node y based on the load balancing algorithm, and so on.

The foregoing embodiments only provide a brief introduction to the basic architecture of the data processing system. The following describes, from the perspective of an acceleration node, a specific execution logic of the acceleration node in implementing a data processing method provided in the embodiments of the present disclosure.

It should be noted that the data processing method provided in some embodiments is employed to process data stored in a columnar format.

FIG. 3 is a flowchart of a data processing method according to some embodiments of the present disclosure, which is applied to an acceleration node located between a computing node and a storage node. As shown in FIG. 3, the data processing method includes the following steps.

In step 301, a query instruction sent by the computing node is received, the query instruction including: identification information of target data and a computation instruction to be executed on the target data.

In step 302, the target data is acquired according to the identification information of the target data.

In step 303, the target data is processed according to the computation instruction, to obtain a computation result.

In step 304, the computation result is sent to the computing node.

In some embodiments, the acceleration node is any node in an acceleration node cluster and provides a unified service invocation interface. The service invocation interface supports a preset data transmission protocol. A data reading plugin adapted to the computing node is deployed in the computing node. The computing node can communicate with the acceleration node by invoking the service invocation interface in the acceleration node via the data reading plugin, such as receiving a query instruction issued by the computing node, or sending a computation result to the computing node.

The identification information of the target data in the query instruction is storage path information or the like of the target data. The computation instruction to be executed on the target data in the query instruction is understood as a computation instruction corresponding to an operator that originally needs to be executed on the target data on the computing node in the absence of the acceleration node.

Optionally, the computation instruction to be executed on the target data in the query instruction is all or part of computation instructions that originally need to be executed on the target data on the computing node.

For example, it is assumed that the target data includes data A, data B, and data C, and an average value of data A, data B, data C originally needs to be calculated on the computing node, corresponding computation instructions include: a computation instruction 1 for calculating a sum of data A, data B, and data C to obtain a computation result D, and a computation instruction 2 for equally dividing the computation result D by 3 to obtain a computation result E. Based on this assumption, the computation instruction to be executed on the target data in the query instruction includes the computation instruction 1 only, or includes both the computation instruction 1 and the computation instruction 2.

In practical application, the computing node determines, according to an association relationship between the computation instructions to be executed on the target data, the computation instruction to be executed on the target data, which is included in the query instruction.

Optionally, the computing node alternatively adds, according to a type of an operator that needs to be executed on the target data, a computation instruction corresponding to an operator of a preset category to the query instruction, and takes the added computation instruction as the computation instruction to be executed on the target data. For example, the operator of the preset category is an essential operator such as a scanning operator, a filtering operator, an aggregation operator, or a column pruning operator. If the type of the operator that needs to be executed by the computing node on the target data includes filtering and aggregation, computation instructions corresponding to the filtering operator and the aggregation operator are added to the query instruction.

After acquiring the target data according to the identification information of the target data, the acceleration node performs a corresponding computation on the target data according to the computation instruction included in the query instruction. It is assumed that the computation instruction included in the query instruction is the foregoing computation instruction 1, the acceleration node calculates the sum of data A, data B, and data C to obtain the computation result D, and sends the computation result D to the computing node.

It is understood that the large volume of target data that is acquired based on the identification information of the target data can be effectively reduced after certain computations. In the foregoing example, if the computation instruction 1 is not executed on the target data on the acceleration node, a total of three data items, namely, data A, data B, and data C, need to be sent to the computing node. However, if the computation instruction 1 is executed on the target data on the acceleration node, only one data item, namely, the computation result D, needs to be sent to the computing node. In this way, the volume of data that needs to be transmitted is effectively reduced. Furthermore, after receiving the computation result D, the computing node needs to execute the computation instruction 2 only, and then the average value of data A, data B, and data C can be determined. In this way, computational load on the computing node can be effectively reduced, and consumption of resources, such as a CPU and an internal memory, of the computing node can also be reduced.

In some embodiments, the computation instruction that originally needs to be executed on the computing node is pushed down to the acceleration node, and the acceleration node performs pre-computation on the target data, that is, executes the computation instruction to be executed on the target data, which is included in the query instruction. In this way, the volume of data that needs to be transmitted to the computing node over a network is effectively reduced, thereby improving data transmission efficiency, and relieving computation pressure on the computing node.

FIG. 4 is a flowchart of another data processing method according to some embodiments of the present disclosure, which is applied to an acceleration node located between a computing node and a storage node. As shown in FIG. 4, the data processing method includes the following steps.

In step 401, a query instruction sent by the computing node is received, the query instruction including: identification information of target data and a computation instruction to be executed on the target data.

In step 402, the target data is acquired locally when it is determined, according to the identification information of the target data, that the target data is cached locally in the acceleration node; the target data is acquired from another acceleration node, in response to the target data being not cached locally in the acceleration node but being queried from the other acceleration node belonging to the same cluster as the acceleration node; and the target data is read from the storage node, in response to the target data being not cached locally in the acceleration node and being not queried from any other acceleration node belonging to the same cluster as the acceleration node, the storage node being configured to store the target data persistently.

In step 403, the target data is processed according to the computation instruction, to obtain a computation result.

In step 404, the computation result is sent to the computing node.

For specific implementation processes of step 401, step 403, and step 404, refer to the foregoing embodiments. Details are not described again here.

In some embodiments, the acceleration node correspondingly has a storage resource configured to cache the target data. It is understood that acquisition of the target data from a local cache of the acceleration node is quicker than acquisition of the target data from the storage node. In addition, a plurality of acceleration nodes are simultaneously deployed in the same acceleration node cluster. The plurality of acceleration nodes transmit locally cached target data to each other over an internal network, and target data transmission between the acceleration nodes in the same acceleration node cluster is quicker than acquisition of the target data from the storage node.

Therefore, to improve the speed at which the acceleration node acquires the target data according to the identification information of the target data, in some embodiments, when the target data is acquired, a corresponding determination condition is set to quickly acquire the target data. A process of acquiring the target data by the acceleration node is described by

using an example in which the computing node communicates with an acceleration node x in the acceleration node cluster.

During acquisition of the target data, first, whether the target data is cached in a local cache of an acceleration node x is determined according to the identification information of the target data, and if the target data is cached in the local cache of the acceleration node x, the target data is acquired directly from the local cache of the acceleration node x.

If the target data does not exist in the local cache of the acceleration node x, whether the target data is cached in another acceleration node in the same acceleration node cluster as the acceleration node x is determined next. If the target data is cached in a local cache of the other acceleration node such as an acceleration node y, the target data in the acceleration node y is transmitted to the acceleration node x over an internal network between the acceleration node y and the acceleration node x. In this way, the target data is acquired. The target data in the local cache of the acceleration node y is data that has been cached previously when the acceleration node y provides an acceleration service and that has not been cleared.

If the target data is not queried from any other acceleration node in the same cluster as the acceleration node x, finally, the target data is read from the storage node configured to store the target data persistently, and the read target data is stored in the local cache of the acceleration node x.

It is understood that the data in the local cache of the acceleration node has a limited validity period, such that data cached in a previous time period will be overwritten by data cached in a subsequent time period. In a case that before the acceleration node acquires the target data, the target data has been loaded into the local cache of the acceleration node or loaded into the local cache of another acceleration node in the same acceleration node cluster as the acceleration node and has not been overwritten, the target data acquisition method provided in the embodiments of the present disclosure can effectively reduce time for acquiring the target data and improve the acceleration performance of the acceleration node. Furthermore, the target data is processed according to the computation instruction, to reduce the volume of data transmitted between the local cache and the computing node, thereby improving data transmission efficiency.

FIG. 5 is a flowchart of still another data processing method according to some embodiments of the present disclosure, which is applied to an acceleration node located between a computing node and a storage node, where the acceleration node includes a processor and a hardware accelerator. As shown in FIG. 5, the data processing method includes the following steps.

In step 501, a query instruction sent by the computing node is received, the query instruction including: identification information of target data and a computation instruction to be executed on the target data.

In step 502, the target data is acquired according to the identification information of the target data.

In step 503, according to a data type of the target data and an instruction type of the computation instruction, whether the hardware accelerator is employed to process the target data according to the computation instruction is determined; in response to determining that the hardware accelerator is employed to process the target data according to the computation instruction, the target data is processed through the processor and the hardware accelerator according to the computation instruction, to obtain a computation result; and in response to determining that the hardware accelerator is not employed to process the target data according to the computation instruction, the target data is processed through the processor according to the computation instruction, to obtain the computation result.

In step 504, the computation result is sent to the computing node.

For specific implementation processes of step 501, step 502, and step 504, refer to the foregoing embodiments. Details are not described again here.

In some embodiments, to enhance the computational performance corresponding to an acceleration service of the acceleration node, in addition to the general-purpose processor (such as a CPU) that supports a plurality of data types and a plurality of instruction types of computation instructions, the acceleration node also provides the hardware accelerator such as a field-programmable gate array (FPGA), a graphics processing unit (GPU), or Advanced Vector Extensions (AVX). The acceleration computing capability of the hardware accelerator is reflected in high computational performance of the hardware accelerator for specific data types or computation instruction types. For example, the FPGA excels at processing operations

such as string matching and regular expression processing, the GPU excels at precise floating-point comparisons and arithmetic operations. Compared with the CPU, the FPGA, the GPU, and the like demonstrate higher processing efficiency for data types or operator types that they are optimized to handle.

In practical application, a plurality of different hardware accelerators, such as an FPGA, a GPU, and AVX, are simultaneously deployed in the same acceleration node, to support a plurality of data types and a plurality of computation instruction types, thereby providing a good acceleration service.

Optionally, the data type of the target data is included in the query instruction sent by the computing node, or is determined based on a data type of historical data.

In a specific implementation process, after acquiring the target data, the acceleration node determines, according to the data type of the target data and the instruction type of the computation instruction included in the query instruction, whether the hardware accelerator is employed to process the target data according to the computation instruction, that is, determines whether a hardware accelerator matching both the data type of the target data and the instruction type of the computation instruction exists in the acceleration node.

When the hardware accelerator matching both the data type of the target data and the instruction type of the computation instruction does not exist in the acceleration node, no hardware accelerator is employed to process the target data according to the computation instruction, and the target data is processed through the processor according to the computation instruction, to obtain the computation result.

For ease of understanding, a description is made with reference to FIG. 6. FIG. 6 is a structural schematic diagram of an acceleration node according to some embodiments of the present disclosure. As shown in FIG. 6, the acceleration node receives a query instruction from a computing node through a processor, and reads target data through the processor. Then, the processor processes the target data according to a computation instruction, to obtain a computation result, and sends the computation result to the computing node.

In practical application, the storage node is self-built by a user, or is a storage service purchased from a cloud provider. Although data is stored in columnar storage formats in the storage node,

the columnar storage formats are different. For example, a storage node 1 stores the target data in a columnar storage format Parquet, and a storage node 2 stores the target data in a columnar storage format ORC.

In some embodiments, after reading the target data, the processor converts the columnar storage formats of the target data into a unified columnar storage format, which is referred to as a first columnar storage format. The first columnar storage format is the same as or different from a second columnar storage format corresponding to the target data in the storage node. After executing the computation instruction on the target data, the processor still sends the computation result in the first columnar storage format to the computing node.

In some embodiments, the first columnar storage format is a columnar storage format Arrow. Based on zero-copy characteristics of Arrow, when the processor sends the computation result to the computing node and the computing node reads the computation result, the computation result can be directly transmitted without being serialized or deserialized. In response to the first columnar storage format being Arrow, a preset data transmission protocol that is supported by the service invocation interface provided by the acceleration node is an Arrow Flight protocol. This protocol defines a data format for inputs and outputs, to achieve a unified service invocation interface.

When the hardware accelerator matching both the data type of the target data and the instruction type of the computation instruction exists in the acceleration node, the hardware accelerator is employed to process the target data according to the computation instruction. Specifically, the target data is processed through the processor and the hardware accelerator according to the computation instruction, to obtain the computation result.

As shown in FIG. 6, when the target data is processed through the processor and the hardware accelerator according to the computation instruction, first, the processor stores the read target data in the first columnar storage format at a first storage address in an internal memory of the acceleration node. Then, the first storage address and the computation instruction to be executed on the target data, which is included in the query instruction, are sent to the hardware accelerator. The hardware accelerator acquires the target data from the first storage address, and processes the target data according to the computation instruction to obtain the computation result. Next, the hardware accelerator stores the computation result in the first

columnar storage format at a second storage address in the internal memory, and sends the second storage address to the processor. Finally, the processor acquires the computation result from the second storage address, and sends the computation result in the first columnar storage format to the computing node.

In some embodiments, because the hardware accelerator having the acceleration computing capability is deployed in the acceleration node, the acceleration node can perform rapid computation on the data type and the computation instruction type that match the hardware accelerator. In this way, the computational performance of the acceleration node is improved. Since the data is transmitted between the processor and the hardware accelerator of the acceleration node, and between the processor and the computing node in the unified columnar storage format such as Arrow, the data format conversion process can be effectively reduced, and high-efficient data transmission is achieved.

The following will describe a data processing apparatus according to at least one embodiment of the present disclosure in detail. A person having ordinary skill in the art may understand that these apparatuses are all configured by using commercially available hardware components and following the steps instructed in this solution.

FIG. 7 is a structural schematic diagram of a data processing apparatus according to some embodiments of the present disclosure. The data processing apparatus is applied to an acceleration node located between a computing node and a storage node. As shown in FIG. 7, the data processing apparatus includes: a receiving module 11, a processing module 12, and a sending module 13.

The receiving module 11 is configured to receive a query instruction sent by the computing node, the query instruction including: identification information of target data and a computation instruction to be executed on the target data.

The processing module 12 is configured to acquire the target data according to the identification information of the target data; and process the target data according to the computation instruction, to obtain a computation result.

The sending module 13 is configured to send the computation result to the computing node.

Optionally, the acceleration node provides a unified service invocation interface. The service invocation interface supports a preset data transmission protocol. A data reading plugin adapted to the computing node is deployed in the computing node, so that the service invocation interface is invoked through the data reading plugin to send the query instruction.

Optionally, the acceleration node includes a processor and a hardware accelerator having the acceleration computing capability. The processing module 12 is specifically configured to determine, according to a data type of the target data and an instruction type of the computation instruction, whether the hardware accelerator is employed to process the target data according to the computation instruction; and process, in response to determining that the hardware accelerator is employed to process the target data according to the computation instruction, the target data through the processor and the hardware accelerator according to the computation instruction, to obtain the computation result; and process, in response to determining that the hardware accelerator is not employed to process the target data according to the computation instruction, the target data through the processor according to the computation instruction, to obtain the computation result.

Optionally, the processing module 12 is further specifically configured to store the read target data in a first columnar storage format at a first storage address in an internal memory of the acceleration node, the target data being read by the processor; send the first storage address and the computation instruction to the hardware accelerator, to allow the hardware accelerator to acquire the target data according to the first storage address, process the target data according to the computation instruction to obtain the computation result, store the computation result in the first columnar storage format at a second storage address in the internal memory, and send the second storage address to the processor; and acquire the computation result from the second storage address through the processor.

Optionally, the processing module 12 is further specifically configured to acquire the target data locally, when it is determined, according to the identification information of the target data, that the target data is cached locally in the acceleration node; acquire the target data from another acceleration node, in response to the target data being not cached locally in the acceleration node but being queried from the other acceleration node belonging to the same cluster as the acceleration node; and read the target data from the storage node, in response to

the target data being not cached locally in the acceleration node and being not queried from any other acceleration node belonging to the same cluster as the acceleration node, the storage node being configured to store the target data persistently.

Optionally, the target data is stored in a second columnar storage format in the storage node.

The data processing apparatus shown in FIG. 7 implements the steps introduced in the foregoing embodiments. For the detailed implementation process and technical effects, refer to the descriptions in the foregoing embodiments. Details are not described again here.

In some embodiments, the structure of the data processing apparatus shown in FIG. 7 is implemented into an electronic device. As shown in FIG. 8, the electronic device includes: a memory 21, a processor 22, and a communication interface 23. The memory 21 stores executable code, and when the executable code is executed by the processor 22, the executable code causes the processor 22 to at least implement the data processing method provided in the foregoing embodiments.

In addition, the embodiments of the present disclosure provide a non-transitory machine-readable storage medium. The non-transitory machine-readable storage medium stores executable code, and when the executable code is executed by a processor of an electronic device, the executable code causes the processor to at least implement the data processing method provided in the foregoing embodiments.

The apparatus embodiments described above are exemplary, and units described as separate components in the apparatus embodiments are or are not separated physically. Some or all of the modules are selected as needed to achieve the objective of the solution of some embodiments. A person having ordinary skill in the art may understand and implement the present disclosure without involving any creative effort.

Through the description of the foregoing implementations, a person having ordinary skill in the art may clearly understand that each implementation is implemented by adding a necessary general-purpose hardware platform, or is implemented through a combination of hardware and software. Based on such understanding, the essence of the foregoing technical solutions or the portion contributing to the related art is embodied in the form of a computer product. The present disclosure may adopt the form of a computer program product implemented on at least

one computer-usable storage medium (including but not limited to a disk memory, a compact disc read-only memory (CD-ROM), an optical memory, and the like) including computer-usable program code.

Finally, it should be noted that the foregoing embodiments are provided to describe the technical solutions of the present disclosure rather than to limit the present disclosure. Although the present disclosure has been described in detail with reference to the foregoing embodiments, a person having ordinary skill in the art will understand that modifications are made to the technical solutions described in the foregoing embodiments or equivalent substitutions are made to some of the technical features. Such modifications or substitutions do not cause the essence of the corresponding technical solutions to depart from the spirit and scope of the technical solutions of the embodiments of the present disclosure.

Claims

1. A data processing method, applied to an acceleration node located between a computing node and a storage node, the data processing method comprising:
receiving a query instruction sent by the computing node, the query instruction comprising:
identification information of target data and a computation instruction to be executed on the target data;
acquiring the target data according to the identification information of the target data;
processing the target data according to the computation instruction, to obtain a computation result; and
sending the computation result to the computing node.
2. The data processing method according to claim 1, wherein the acceleration node provides a unified service invocation interface, and the service invocation interface supports a preset data transmission protocol; and a data reading plugin adapted to the computing node is deployed in the computing node, so that the service invocation interface is invoked through the data reading plugin to send the query instruction.
3. The data processing method according to claim 1, wherein the acceleration node comprises a processor and a hardware accelerator; and processing the target data according to the computation instruction, to obtain the computation result comprises:
determining, according to a data type of the target data and an instruction type of the computation instruction, whether the hardware accelerator is employed to process the target data according to the computation instruction;
processing, in response to determining that the hardware accelerator is employed to process the target data according to the computation instruction, the target data through the processor and the hardware accelerator according to the computation instruction, to obtain the computation result; and
processing, in response to determining that the hardware accelerator is not employed to process the target data according to the computation instruction, the target data through the processor according to the computation instruction, to obtain the computation result.
4. The data processing method according to claim 3, wherein processing the target data through the processor and the hardware accelerator according to the computation instruction, to obtain the computation result comprises:

storing the read target data in a first columnar storage format at a first storage address in an internal memory of the acceleration node, the target data being read by the processor; sending the first storage address and the computation instruction to the hardware accelerator, to allow the hardware accelerator to acquire the target data according to the first storage address, process the target data according to the computation instruction to obtain the computation result, store the computation result in the first columnar storage format at a second storage address in the internal memory, and send the second storage address to the processor; and

acquiring the computation result from the second storage address through the processor.

5. The data processing method according to any one of claims 1 to 4, wherein acquiring the target data according to the identification information of the target data comprises:
acquiring the target data locally, when it is determined, according to the identification information of the target data, that the target data is cached locally in the acceleration node;
acquiring the target data from another acceleration node, in response to the target data being not cached locally in the acceleration node but being queried from the other acceleration node belonging to the same cluster as the acceleration node; and
reading the target data from the storage node, in response to the target data being not cached locally in the acceleration node and being not queried from any other acceleration node belonging to the same cluster as the acceleration node, the storage node being configured to store the target data persistently.
6. The data processing method according to any one of claims 1 to 4, wherein the target data is stored in a second columnar storage format in the storage node.
7. A data processing system, the data processing system comprising:
a computing node cluster, an acceleration node cluster, and a storage node cluster, the acceleration node cluster being coupled between the computing node cluster and the storage node cluster; and
any acceleration node in the acceleration node cluster being configured to perform the data processing method as claimed in any one of claims 1 to 6.
8. The data processing system according to claim 7, further comprising: a control node, the control node being configured to determine, according to a type of a target computing node

in the computing node cluster, a data reading plugin matching the type of the target computing node, and deploy the data reading plugin in the target computing node.

9. A data processing apparatus, applied to an acceleration node located between a computing node and a storage node, the data processing apparatus comprising:
 - a receiving module, configured to receive a query instruction sent by the computing node, the query instruction comprising: identification information of target data and a computation instruction to be executed on the target data;
 - a processing module, configured to acquire the target data according to the identification information of the target data; and process the target data according to the computation instruction, to obtain a computation result; and
 - a sending module, configured to send the computation result to the computing node.
10. An electronic device, comprising: a memory, a processor, and a communication interface, wherein the memory stores executable code, and when the executable code is executed by the processor, the executable code causes the processor to implement the data processing method as claimed in any one of claims 1 to 6.
11. A non-transitory machine-readable storage medium, the non-transitory machine-readable storage medium storing executable code, and when the executable code is executed by a processor of an electronic device, the executable code causing the processor to implement the data processing method as claimed in any one of claims 1 to 6.