

DEVICE AND METHOD FOR SYNCHRONIZING DATA BETWEEN DATA SOURCES

TECHNICAL FIELD

[0001] Various aspects of this disclosure relate to devices and methods for synchronizing data between data sources.

BACKGROUND

[0002] In a distributed storage system where there are multiple data sources storing the same data, data is synchronized between the data sources such that clients may access the same data on all data sources. This, however, means, that also data which has low quality and is therefore useless, e.g. because it is erroneous, inaccurate or incomplete, is propagated between data sources and wastes transmission bandwidth and storage space

[0003] Accordingly, efficient approaches for removing low-quality data from a distributed storage system are desirable.

SUMMARY

[0004] Various embodiments concern a method for synchronizing data between data sources, including generating a synchronization task configuration file for a synchronization framework, which includes including information about one or more filtering conditions into the synchronization task configuration file which causes the synchronization framework to filter data in accordance with the one or more filtering conditions when performing a synchronization task in accordance with the synchronization task configuration file and controlling the synchronization framework to perform a synchronization task between a first data source and a second data source in accordance with the synchronization task configuration file.

[0005] According to one embodiment, filtering data in accordance with the one or more filtering conditions including checking whether the one or more filtering conditions are fulfilled by the data and performing a filtering action if the data fulfils the one or more filtering conditions.

[0006] According to one embodiment, the data includes multiple data units and filtering data in accordance with the one or more filtering conditions including checking, for each data unit of the data, whether the one or more filtering conditions are fulfilled by the data unit and filtering out the data unit if the data unit fulfils the one or more filtering conditions.

[0007] According to one embodiment, the synchronization task is a transmission of data from the first data source to the second data source and filtering out the data unit includes omitting transmitting the data unit to the second data source.

[0008] According to one embodiment, each data unit includes a plurality of data fields and checking whether the one or more filtering conditions are fulfilled by the data unit includes checking whether a data field of the data unit fulfils the one or more filtering conditions.

[0009] According to one embodiment, the one or more filtering conditions include a filtering condition which is fulfilled by a data field if the value of the data field has a value in a set of enumerated values or in a range of values.

[0010] According to one embodiment, the method includes including a filter type of one or more filtering conditions into the synchronization task configuration file which causes the synchronization framework to filter data in accordance with the filter type when performing a synchronization task in accordance with the synchronization task configuration file.

[0011] According to one embodiment, the filter type is an enumerated type or a range type.

[0012] According to one embodiment, the one or more filtering conditions are filtering conditions according to one or more quality criteria.

[0013] According to one embodiment, the one or more quality criteria include one or more of: a criterion which is violated if the data is erroneous, a criterion which is violated if the data is redundant, a criterion which is violated if the data is incomplete and a criterion which is violated if the data is inaccurate.

[0014] According to one embodiment, the method further includes providing the synchronization framework with a filter for filtering data in accordance with the one or more filtering conditions.

[0015] According to one embodiment, the synchronization framework parses the configuration file to determine the one or more filtering criteria.

[0016] According to one embodiment, the synchronization framework parses the data to determine one or more data fields for which to check whether the one or more filtering criteria are fulfilled.

[0017] According to one embodiment, the synchronization framework is FlinkX.

[0018] According to one embodiment, the first data source is implemented as one or more Kafka brokers.

[0019] According to one embodiment, the method includes generating the synchronization task configuration file in JSON format.

[0020] According to one embodiment, generating a synchronization task configuration file includes presenting a user with a graphical user interface to input the one or more filtering conditions and including information about one or more filtering conditions into the synchronization task configuration file according to user input.

[0021] According to one embodiment, a data synchronization device is provided including a communication interface, a memory interface and a processing unit configured to perform the method for synchronizing data between data sources as described above.

[0022] According to one embodiment, a computer program element is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method for synchronizing data between data sources described above.

[0023] According to one embodiment, a computer-readable medium is provided including program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method for synchronizing data between data sources described above.

[0024] It should be noted that embodiments described in context of the method for synchronizing data between data sources are analogously valid for the data synchronization device and vice versa.

BRIEF DESCRIPTION OF THE DRAWINGS

[0025] The invention will be better understood with reference to the detailed description when considered in conjunction with the non-limiting examples and the accompanying drawings, in which:

- FIG. 1 shows a data distribution arrangement.

- FIG. 2 shows excerpts of an exemplary configuration file.
- FIG. 3 shows a flow diagram for performing a filtering operation in a data transmission.
- FIG. 4 illustrates the operation of a filter.
- FIG. 5 shows a graphical user interface including a drop-down menu allowing a user to select information about a filtering condition.
- FIG. 6 shows a flow diagram illustrating a method for synchronizing data between data sources.
- FIG. 7 shows a data synchronization device according to an embodiment.

DETAILED DESCRIPTION

[0026] The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details and embodiments in which the disclosure may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure. Other embodiments may be utilized and structural, and logical changes may be made without departing from the scope of the disclosure. The various embodiments are not necessarily mutually exclusive, as some embodiments can be combined with one or more other embodiments to form new embodiments.

[0027] Embodiments described in the context of one of the devices or methods are analogously valid for the other devices or methods. Similarly, embodiments described in the context of a device are analogously valid for a method, and vice-versa.

[0028] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly described in these other embodiments. Furthermore, additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0029] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0030] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0031] In the following, embodiments will be described in detail.

[0032] FIG. 1 shows a data distribution arrangement 100.

[0033] The data distribution arrangement 100 includes multiple data sources 101, 102, 103 forming a distributed data storage system. Clients 104 may retrieve data from the data sources 101, 102, 103. The data sources 101, 102, 103 and the clients 104 are connected by a communication network 109 (e.g. the Internet) which may be implemented by multiple sub-networks.

[0034] The data sources 101, 102, 103 may be heterogeneous, i.e. operate according to different data storage (e.g. data base) systems but may also operate according to the same data storage system. For example, data sources 101, 102, 103 may operate according to Hdfs, Hbase, SQLServer, MySQL, Oracle, Kafka, TiDB, Apache Hive etc. A client 104 may (depending on its type) contact a suitable data source 101, 102, 103 to retrieve data. To ensure that a client 104 may retrieve data from any one of the data sources 101, 102, 103, the data sources 101, 102, 103 are synchronized.

[0035] According to various embodiments, data between two data sources 101, 102 is synchronized using FlinkX. FlinkX is a distributed offline data synchronization framework based on Flink that enables efficient data migration between multiple heterogeneous data sources. FlinkX provides a simplification and encapsulation of the configuration of synchronization (e.g. streaming) jobs, and can be applied in the field of big data to synchronize various heterogeneous data sources. It should be noted that FlinkX is only an example and other synchronization frameworks or tools may be used like for example DataX.

[0036] In FlinkX, when a data transmission 105 from a first data source 101 to a second data source 102 is performed for synchronization, the first data source 101 acts as a “reader” and is handled by a corresponding reader-plugin-in (e.g. a Kafka reader) and the second data source 102 acts as a “writer” and is handled by a corresponding writer-plugin-in (e.g. a MySQL writer).

[0037] In FlinkX, different data sources 101, 102 are abstracted into different reader plug-ins and different data destinations are abstracted into different writer plug-ins. Theoretically, the FlinkX framework can support data synchronization for any data source type.

[0038] A data synchronisation task, i.e. data transmission 105, between two data sources 101, 102 is denoted as a job.

[0039] A job is defined by a configuration file which is, in FlinkX, a JOSN file with a certain pre-defined structure.

[0040] The configuration file has the following structure:

1	{
2	"job": {
3	"setting": {...},
4	"content": [...]
5	}
6	}

Table 1: top-level configuration file structure

[0041] The data synchronization task contains one job element, and this element contains two parts of setting and content.

[0042] Setting is for example used to configure the speed limit and error control.

1	"speed": {
2	"channel": 3,
3	"bytes": 0
4	}

Table 2: example of speed information element

1	"errorLimit": {
2	"record": 10000,
3	"percentage": 100
4	}

Table 3: example of error limit information element

[0043] Setting may also be used to configure managing of dirty data, logging and restoring.

[0044] The content part is used to configure certain task information, including the origin data source 101 (reader plug-in information) and destination data source 102 (writer plug-in information).

1	"content": [
2	{
3	"reader": {

4	"name": "...",
5	"parameter": {
6	...
7	}
8	},
9	"writer": {
10	"name": "...",
11	"parameter": {
12	...
13	}
14	}
15	}
16	]

Table 4: example of content part structure of configuration file

[0045] So, a synchronization engine 106 controls the data transmission 105 in accordance with a configuration file 107. The synchronization engine 106 may run on a separate computer or on a computer which implements (possibly together with one or more other computers, e.g. in a cloud implementation) one of the data sources 101, 102.

[0046] According to various embodiments, the synchronization engine 106 increases the quality of the data when reading the data from the origin data source 101 (e.g. Kafka data source) by filtering out dirty data. Thus overhead for synchronization and storage overhead is reduced.

[0047] According to various embodiments, this is achieved by the synchronization engine 106 implementing or controlling a filter 108 configured to filter the data transmitted in the transmission 105 in accordance with one or more filtering criteria specified in the configuration file 107.

[0048] FIG. 2 shows excerpts of a configuration file 200 according to an embodiment.

[0049] The configuration file 200 is an example of the configuration file 107 according to an embodiment.

[0050] In the configuration file 107, all parameter that the synchronization engine 106 needs to carry out the synchronization task 105 are defined.

[0051] In the example of FIG. 2, the first data source 101 (reader) is a Kafka data source and the second data source 102 (writer) is a MySQL data source.

[0052] The configuration file 200 includes an indication of a filtering criterion (i.e. a filtering condition) 201 and an indication of the type of the filter criterion. Specifically, the configuration file specifies

"filter": "ID,BR",

"filterType": 1

[0053] This means that if the data contains a data field equal to ID or BR, the function specified by filterType will be performed. In other words, the filter criterion is whether the data contains a data field equal to ID or BR. filterType '1' means that the data is saved (i.e. not filtered out), filterType '2' means that the data is filtered out.

[0054] In accordance with the specified filtering criterion, the filter 108 checks the values of data fields of the data transmitted from the first data source 101 to the second data source 102 (e.g. Kafka data fields).

[0055] For example, the filter 108 checks all or one or more specific data fields of data units of data provided by the first data source 101. A data unit is for example a file, a table, an array, a transmission frame, a data packet, a set of one or more data records or any other predefined chunk of data. If for a unit of data the filter condition is fulfilled (depending on the filter condition e.g. a field has a certain value or lies in a range of values or certain fields have a certain combination of values) a filtering action is applied to the data unit, e.g. the data unit is filtered out (discarded), i.e. not forwarded to and not stored in the second data source 102.

[0056] The filtering condition may for example be fulfilled if the data unit does not fulfil a predefined quality criterion, e.g. the values of one or more data fields may indicate that the data unit is erroneous, redundant, inaccurate, incomplete etc. For example, the data unit may be a table entry which is incomplete and therefore it is filtered out.

[0057] The filter 108 may also check whether the values of data fields have a predefined type (e.g. integer, string, float, double) and may filter out entries or data units which do not have the predefined type.

[0058] The filter condition can be of an enumeration type (i.e. the filtering criterion is fulfilled if the data field has a value equal to one or more values of a list of values) or of range (i.e. interval) type (i.e. the filtering criterion is fulfilled if the data field has a value in a range of values).

[0059] By defining suitable filtering conditions, it can be ensured that the data transmitted to and stored in the second the source 102 fulfils its requirements, e.g. regarding type and function. The filtering operation can thus be seen as a quality inspection.

[0060] By filtering out "dirty" data (which does not fulfils requirements, e.g. is erroneous), in particular, waste of storage space at the second data source 102 can be avoided.

[0061] The filter 108 may be implemented at the output of the first data source 101, at the input of the second data source 102 or by an intermediate component.

[0062] FIG. 3 shows a flow diagram for performing a filtering operation in the transmission 105 from the first data source 101 to the second data source 102.

[0063] The filtering operation is performed by the synchronization engine 106 when performing synchronization according to the synchronization framework, i.e. is operating according to the synchronization framework.

[0064] In 301, the synchronization engine 106 receives data about a request to perform the transmission 105 of data from the first data source 101 to the second data source 102. The data about the request includes the configuration file 107 (e.g. configuration file 200). It should be noted that the synchronization engine 106 may itself have generated the configuration file 300, e.g. when executing a configuration tool. The configuration file can thus be seen as the interface between the configuration tool and the synchronization framework (which may both be implemented by the synchronization engine 106).

[0065] In 302, the synchronization engine 106 parses the data about the request (e.g. the configuration file 200) and determines, in 303, based on the result of the parsing, whether a filtering function (e.g. a quality inspecting function) should be performed. If not, filtering is skipped.

[0066] If yes, the synchronization engine 106 checks, in 304, whether the configuration file 107 indicates one of a predefined set of types of filtering conditions (for which the filter 108 is implemented). In the present example, the filtering condition types are an enumeration type and a range type. If the configuration file 107 does not indicate one of the predefined set of filtering condition types, filtering is skipped.

[0067] If the configuration file does not indicate one of the predefined set of filtering condition types, the filtering takes effect in 305 (e.g. the filter 108 is activated wherein it is configured with the filtering condition or conditions indicated in the configuration file 107).

[0068] In 306, the filter 108 filters the data transmitted in the data transmission 105 and in 307 the second data source 102 stores the filtered data (i.e. the transmitted data remaining after the filtering, i.e. not filtered out).

[0069] FIG. 4 illustrates the operation of a filter 400.

[0070] The filter 400 for example corresponds to the filter 108 and may be implemented as part of a (e.g. streaming) synchronization framework (e.g. FlinkX).

[0071] The filter 400 receives data from an origin data source 401 (e.g. corresponding to the first data source 101) in the form of data units.

[0072] In 402, the filter 400 parses the data units and locates fields of the data units for which a filtering condition should be checked.

[0073] In 403, for each field for which a filtering condition should be checked, the filter determines one or more filtering conditions which should be checked for the field.

[0074] In 404, for each field for which a filtering condition should be checked, the filter checks the determined one or more filtering conditions (e.g. an enumeration or interval filtering condition).

[0075] In 405, the filter 108 filters out the data unit if a filtering condition is fulfilled by the data unit. This is for example the case if any one of the fields for which a filtering condition should be checked fulfils the determined one or more filtering conditions. The fields for it should be checked whether a filtering condition is fulfilled may also be indicated in the configuration file and determined by the filter 400 in 402.

[0076] The filtered data (e.g. remaining data units) are then forwarded to a destination data source 406 (e.g. corresponding to the second data source 102).

[0077] As explained above, the information about the filtering function to be performed, in particular information about one or more filtering criteria or conditions and possibly also filtering types, value enumerations or ranges etc. are included in the synchronization task configuration file 107. The synchronization engine 106 may display a graphical user interface to a user (e.g. on one of the clients 104) to let the user define this information.

[0078] FIG. 5 shows a graphical user interface 500 including a drop-down menu allowing a user to select information about a filtering condition.

[0079] When the user has input the information about the filtering function, the synchronization engine 106 includes the information into the synchronization task configuration file 107 and supplies the synchronization task configuration file 107 such that the synchronization framework performs the synchronization task 105 in accordance with the synchronization task configuration file 107 and in particular in accordance with the information about the filtering function.

[0080] In summary, according to various embodiments, a method is provided as illustrated in FIG. 6.

[0081] FIG. 6 shows a flow diagram 600 illustrating a method for synchronizing data between data sources, for example performed by a synchronization engine.

[0082] In 601, a synchronization task configuration file for a synchronization framework is generated. This includes including information about one or more filtering conditions (e.g. a specification of the one or more filtering conditions) into the synchronization task configuration file which causes the synchronization framework to filter data in accordance with the one or more filtering conditions when performing a synchronization task in accordance with the synchronization task configuration file.

[0083] In 602, the synchronization framework is controlled (e.g. triggered) to perform a synchronization task between a first data source and a second data source in accordance with the synchronization task configuration file.

[0084] According to various embodiments, in other words, filtering of data transmitted from an origin data source to a destination data source is supported. For example, a filtering criterion may be fulfilled depending on the value of a data field. The filtering condition may be configured to be applied to a certain set of data fields to be transmitted. Thus, filtering depending on certain (e.g. internal) fields of a transmission (e.g. a synchronization stream) may be supported.

[0085] At least one of the data sources may operate according to Kafka such that the data may be seen as Kafka data. The transmission of data from the first data source to the second data source is for example performed using FlinkX. When the data of the first data source is transmitted to the second data source (e.g. in a big data context), the filter (configured by the FlinkX configuration file to perform a certain analysis, i.e. check one or more certain filtering criteria) can perform quality inspection and filtering on the data (FlinkX-)internally, thereby improving the data quality in the process of synchronization operation. By using the streaming configuration file for defining the filtering condition, the filtering functionality may be implemented with little programming effort.

[0086] The method of FIG. 6 is for example carried out by a server computer as illustrated in FIG. 7.

[0087] FIG. 7 shows a data synchronization device 700 according to an embodiment.

[0088] The data synchronization device 700 is implemented by one or more computers and includes a communication interface 701 (e.g. configured to receive configuration data (e.g. user input data), send data to a client for displaying a graphical user interface, receive

data to be filtered, send out data to be filtered or send a data synchronization task configuration file to another device for performing the filtering.)

[0089] The data synchronization device 700 further includes a processing unit 702 and a memory 703. The memory 703 may be used by the processing unit 702 to store, for example, the synchronization task configuration file. The data synchronization device is configured to perform the method of FIG. 6. The data synchronization device may implement the data synchronization framework or may control another device implementing the data synchronization framework (e.g. by supplying the synchronization task configuration file to the other device).

[0090] The methods described herein may be performed and the various processing or computation units and the devices and computing entities described herein may be implemented by one or more circuits. In an embodiment, a "circuit" may be understood as any kind of a logic implementing entity, which may be hardware, software, firmware, or any combination thereof. Thus, in an embodiment, a "circuit" may be a hard-wired logic circuit or a programmable logic circuit such as a programmable processor, e.g. a microprocessor. A "circuit" may also be software being implemented or executed by a processor, e.g. any kind of computer program, e.g. a computer program using a virtual machine code. Any other kind of implementation of the respective functions which are described herein may also be understood as a "circuit" in accordance with an alternative embodiment.

[0091] While the disclosure has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

1. A method for synchronizing data between data sources, comprising:
generating a synchronization task configuration file for a synchronization framework, comprising including information about one or more filtering conditions into the synchronization task configuration file which causes the synchronization framework to filter data in accordance with the one or more filtering conditions when performing a synchronization task in accordance with the synchronization task configuration file;
and
controlling the synchronization framework to perform a synchronization task between a first data source and a second data source in accordance with the synchronization task configuration file.
2. The method of claim 1, wherein filtering data in accordance with the one or more filtering conditions comprising checking whether the one or more filtering conditions are fulfilled by the data and performing a filtering action if the data fulfils the one or more filtering conditions.
3. The method of claim 1 or 2, wherein the data comprises multiple data units and filtering data in accordance with the one or more filtering conditions comprising checking, for each data unit of the data, whether the one or more filtering conditions are fulfilled by the data unit and filtering out the data unit if the data unit fulfils the one or more filtering conditions.
4. The method of claim 3, wherein the synchronization task is a transmission of data from the first data source to the second data source and filtering out the data unit comprises omitting transmitting the data unit to the second data source.
5. The method of claim 3 or 4, wherein each data unit comprises a plurality of data fields and wherein checking whether the one or more filtering conditions are fulfilled by the data unit comprises checking whether a data field of the data unit fulfils the one or more filtering conditions.

6. The method of any one of claims 1 to 5, wherein the one or more filtering conditions comprise a filtering condition which is fulfilled by a data field if the value of the data field has a value in a set of enumerated values or in a range of values.
7. The method of any one of claims 1 to 6, comprising including a filter type of one or more filtering conditions into the synchronization task configuration file which causes the synchronization framework to filter data in accordance with the filter type when performing a synchronization task in accordance with the synchronization task configuration file.
8. The method of claim 7, wherein the filter type is an enumerated type or a range type.
9. The method of any one of claims 1 to 8, wherein the one or more filtering conditions are filtering conditions according to one or more quality criteria.
10. The method of claim 9, wherein the one or more quality criteria include one or more of: a criterion which is violated if the data is erroneous, a criterion which is violated if the data is redundant, a criterion which is violated if the data is incomplete and a criterion which is violated if the data is inaccurate.
11. The method of any one of claims 1 to 10, further comprising providing the synchronization framework with a filter for filtering data in accordance with the one or more filtering conditions.
12. The method of any one of claims 1 to 11, wherein the synchronization framework parses the configuration file to determine the one or more filtering criteria.
13. The method of any one of claims 1 to 12, wherein the synchronization framework parses the data to determine one or more data fields for which to check whether the one or more filtering criteria are fulfilled.

14. The method of any one of claims 1 to 13, wherein the synchronization framework is FlinkX.
15. The method of any one of claims 1 to 14, wherein the first data source is implemented as one or more Kafka brokers.
16. The method of any one of claims 1 to 15, comprising generating the synchronization task configuration file in JSON format.
17. The method of any one of claims 1 to 16, wherein generating a synchronization task configuration file comprises presenting a user with a graphical user interface to input the one or more filtering conditions and including information about one or more filtering conditions into the synchronization task configuration file according to user input.
18. A data synchronization device comprising a communication interface, a memory interface and a processing unit configured to perform the method of any one of claims 1 to 17.
19. A computer program element comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 1 to 17.
20. A computer-readable medium comprising program instructions, which, when executed by one or more processors, cause the one or more processors to perform the method of any one of claims 1 to 17.