

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 needs to be synchronized between the data sources. However, for example to support different clients, data sources may operate according to different data storage schemas. This makes the synchronization a complex task. While efficient synchronization frameworks like FlinkX exist, the configuration of these frameworks is, in accordance with the complexity of the synchronization, complex and therefore requires a lot of effort, in particular to avoid errors.

[0003] Accordingly, efficient approaches for synchronizing data between data sources are desirable.

SUMMARY

[0004] Various embodiments concern a method for synchronizing data between data sources, comprising presenting a user with a graphical user interface to input values for configuration parameters for a synchronization task between a first data source and a second data source, determining values of the configuration parameters at least partially based on user input to the graphical user interface, generating a synchronization task configuration file by including each determined configuration parameter value into a section of the configuration file assigned to the configuration parameter according to a pre-defined configuration file structure and starting a synchronization task in accordance with the generated configuration file.

[0005] According to one embodiment, starting the synchronization task in accordance with the generated configuration file comprises supplying the generated configuration file to a data synchronization framework.

[0006] According to one embodiment, the synchronization task comprises a transmission of data from the first data source to the second data source.

[0007] According to one embodiment, the values of the configuration parameters include the information that the first data source is an origin data source of the transmission and that the second data source is a destination data source of the transmission.

[0008] According to one embodiment, the values of the configuration parameters include information about formatting functions to perform on the data for transmitting data from the first data source to the second data source.

[0009] According to one embodiment, the values of the configuration parameters include information about the first data source and the second data source.

[0010] According to one embodiment, the information about the first data source includes an identification of the first data source and the information about the second data source includes an identification of the second data source.

[0011] According to one embodiment, the synchronization task configuration file comprises a section for parameters related to the first data source, a section for parameters related to the second data source and a general section and generating the synchronization task configuration file comprises classifying the determined configuration parameter values according to whether they relate to the first data source, relate to the second data source or are general configuration parameters and putting each configuration parameter value in the section of the synchronization task configuration file to which it relates.

[0012] According to one embodiment, determining values of the configuration parameters comprises presenting possible values of the configuration parameters via the graphical user interface for selection and confirmation.

[0013] According to one embodiment, determining values of the configuration parameters comprises receiving an original synchronization task configuration file and parsing the original synchronization task configuration file to determine possible values of the configuration parameters.

[0014] According to one embodiment, generating the synchronization task configuration file includes keeping configuration parameter values of the original configuration file which have been confirmed by user input to the graphical user interface and updating configuration parameter values which have been changed by user input to the graphical user interface.

[0015] According to one embodiment, the pre-defined configuration file structure is a FlinkX configuration file structure.

[0016] According to one embodiment, the method comprises performing the synchronization task using FlinkX.

[0017] According to one embodiment, the pre-defined configuration file structure is a JSON format.

[0018] According to one embodiment, a data synchronization device comprising a communication interface, a memory interface and a processing unit configured to perform the method for synchronizing data between data sources as described above.

[0019] According to one embodiment, a computer program element is provided comprising 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.

[0020] According to one embodiment, a computer-readable medium is provided comprising 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.

[0021] 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

[0022] 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 the generation of a configuration file for a synchronization task.
- FIG. 4 illustrates a configuration system.
- FIG. 5 shows a graphical user interface presenting a drop-down menu to select an origin data source.

- FIG. 6 shows a graphical user interface presenting a drop-down menu to select a destination data source.
- FIG. 7 shows a graphical user interface for inputting base information.
- FIG. 8 shows a graphical user interface for inputting resource information.
- FIG. 9 shows a graphical user interface presenting a drop-down menu to select parameters to configure a data formatting function.
- FIG. 10 shows a graphical user interface presenting a window to input a JSON configuration file.
- FIG. 11 shows a flow diagram illustrating a method for synchronizing data between data sources.
- FIG. 12 shows a data synchronization device according to an embodiment.

DETAILED DESCRIPTION

[0023] 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.

[0024] 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.

[0025] 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.

[0026] 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.

[0027] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0028] In the following, embodiments will be described in detail.

[0029] FIG. 1 shows a data distribution arrangement 100.

[0030] The data distribution arrangement 100 comprises 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.

[0031] In the present example, it is assumed that the data sources 101, 102, 103 are heterogeneous. For example a data source 101, 102 may operate according to Hdfs, Hbase, SQLServer, MySQL, Oracle, Kafka, 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.

[0032] 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.

[0033] 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 (e.g. a Kafka reader) and the second data source 102 acts as a “writer” and is handled by a corresponding writer-plugin (e.g. a MySQL writer).

[0034] In FlinkX, different data sources 101, 102 are abstracted into different reader plugins and different data destinations are abstracted into different writer plug-ins. Theoretically, the FlinkX framework can support data synchronization for any data source type. As a set of

ecosystems, each new set of data sources can be connected to the existing data source with the newly added data source.

[0035] A data synchronisation task, i.e. transmission 105, between two data sources 101, 102 is denoted as a job.

[0036] A job is defined by a configuration file which is, in FlinkX, a JOSN file with a certain pre-defined structure. 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.

[0037] The configuration file has the following structure:

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

Table 1: top-level configuration file structure

[0038] The data synchronization task contains one job element, and this element contains two parts of setting and content.

[0039] 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

[0040] Setting may also be used to configure managing of dirty data, logging and restoring.

[0041] 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

[0042] FIG. 2 shows excerpts of an exemplary configuration file 200 as an example for a configuration file 107 in JOSN for illustration (written in columns from left to right, omissions are indicated by "...").

[0043] In the example of FIG. 2, the first data source (reader) is a Kafka data source and the second data source (writer) is a MySQL data source.

[0044] In the configuration file 107, all parameter that the synchronization engine 106 needs to carry out the synchronization task 105 are defined.

[0045] In the exemplary configuration file 200, this for example includes core configuration parameters such as

[0046] "channel": 32

[0047] "splitTableRange": 6

[0048] "splitTableIndex": 8

[0049] "splitDBIndex": 4

[0050] etc.

[0051] As can be seen in the example of FIG. 2, these core configuration parameters are mixed at various positions into the configuration file 107, e.g. partially in context of a data source (i.e. in the content part) and partially in the setting part.

[0052] Thus, the configuration of a job is inconvenient and difficult to handle and the maintenance effort for a job is high.

[0053] Therefore, according to various embodiments, an approach is provided which allows management of the core parameters for synchronization tasks which in the end allows reliable synchronization between data sources 101, 102, 103 and thus maintenance of a distributed data storage system. Performing a synchronization task, e.g. a transmission from the first data source 101 to the second data source 102, may in particular include the determination of core parameters related to the second data source 102. Thus, these core parameters need to be extracted and managed. According to various embodiments, corresponding functions for this are provided.

[0054] These functions allow, e.g. when using FlinkX for a synchronization job, to simplify the synchronization (e.g. possibly large-scale streaming) job configuration, in particular extract streaming job core parameters (e.g. according to a data source schema), controllable management of core parameters and thus in the end increase reliability of synchronization and reduce workload for the operation.

[0055] FIG. 3 shows a flow diagram 300 for the generation of a configuration file for a synchronization task.

[0056] In 301, a synchronization event is triggered. For example, the synchronization engine 106 is configured to regularly synchronize data between the data sources 101, 102 and in the present examples determines that data needs to be transferred from the first data source 101 to the second data source 102. The synchronization event may include the creation of an original (base) configuration file (e.g. a template, possibly taken from a previous synchronization task).

[0057] In 302, the synchronization engine 106 determines whether it should perform the synchronization task based on FlinkX. If that is not the case (e.g. because FlinkX is not configured to be used for the present synchronization task), the synchronization engine 106 skips the processing described in the following (and e.g. continues in a conventional manner using another synchronization framework).

[0058] If the synchronization task is to be carried out using FlinkX, the synchronization engine 106 starts using a (synchronization) configuration system 108 for gathering configuration information in 303.

[0059] FIG. 4 illustrates a configuration system 400.

[0060] The configuration system 400 includes a data source information retrieval module 401. It reads information about the read data source (i.e. the data source acting as reader) and information about the write data source (i.e. the data source acting as writer) to ensure the uniqueness of the read data source and write data source.

[0061] The data source information retrieval module 401 may for example retrieve this information from an original (e.g. base) configuration file that was created (and triggers the job e.g. in 301) or from another file indicating the data sources 101, 102 which should be synchronized, e.g. provided or input by a user.

[0062] The configuration system 400 further includes a unified management module 402 which gets the information about the data sources 101, 102 and in 304 confirms the information about the data sources (e.g. upon confirmation input from a user). These may be seen as the main system parameters.

[0063] The configuration system 400 further comprises a core parameter module 403 which, in 305, configures and confirms core parameters. These may for example include parameters controlling functions required for transmitting data from one data source to another such as reformatting (e.g. table splitting).

[0064] Further, in 305, one or more additional modules 404 of the configuration system 400 may set the values of additional fields of the configuration file and confirm configuration values (e.g. upon confirmation input from a user).

[0065] If, in 306, the configuration file is to be created (e.g. when a user inputs a corresponding instruction, e.g. by pressing a corresponding button), the configuration system 400 generates the configuration file 107 according to the set parameters and fields and stores it in 307.

[0066] The synchronization engine 106 then runs the synchronization task according to the configuration file 107 using FlinkX.

[0067] The setting of configuration parameters may include querying a user for corresponding information.

[0068] This may include presenting a graphical user interface to the user for inputting the information. For example, the configuration system 108 may communicate with one of the clients 104 to cause the client 104 to display a corresponding graphical user interface on a screen.

[0069] The user may for example be presented with a drop down menu to select the origin data source 101 and similarly with a drop down menu to select the destination data source 102.

[0070] The user may further be presented with a graphical user interface to enter base information (such as channel), resource info (such as information about the origin data source such as a Kafka group ID), parameters to configure certain functions controlling how data is synchronized (e.g. formatted)

[0071] The user may also be presented with the opportunity (e.g. a button) to input a JSON configuration file from which the configuration system parses the various configuration parameters.

[0072] Examples for corresponding GUIs are shown in FIG. 5 to FIG. 10.

[0073] FIG. 5 shows a graphical user interface 500 presenting a drop-down menu to select the origin data source.

[0074] FIG. 6 shows a graphical user interface 600 presenting a drop-down menu to select the destination data source.

[0075] FIG. 7 shows a graphical user interface 700 for inputting base information.

[0076] FIG. 8 shows a graphical user interface 800 for inputting resource information.

[0077] FIG. 9 shows a graphical user interface 900 presenting a drop-down menu to select parameters to configure a data formatting function.

[0078] FIG. 10 shows a graphical user interface 1000 presenting a window to input a JSON configuration file.

[0079] According to various embodiments, the configuration system 400 provides a centralized management of data synchronization and controllable configuration of core parameters of synchronization jobs. This may include the management according to the various data source schema, such as different data formats etc. used by the different data sources, i.e. setting configuration parameters to correctly synchronize data between data sources of different types (e.g. Kafka and MySQL etc.).

[0080] In summary, according to various embodiments, a method is provided as illustrated in FIG. 11.

[0081] FIG. 11 shows a flow diagram 1100 illustrating a method for synchronizing data between data sources.

[0082] In 1101, a user is presented with a graphical user interface to input values for configuration parameters for a synchronization task between a first data source and a second data source.

[0083] In 1102, values of the configuration parameters are determined at least partially based on user input to the graphical user interface.

[0084] In 1103, a synchronization task configuration file is generated by including each determined configuration parameter value into a section of the configuration file assigned to the configuration parameter according to a pre-defined configuration file structure.

[0085] In 1104, a synchronization task is started in accordance with the generated configuration file (e.g. a synchronization framework is instructed or executed to perform a synchronization task in accordance with the generated configuration file, according to the synchronization framework, control messages may then be sent to the data sources).

[0086] According to various embodiments, in other words, a configuration method (e.g. performed by a configuration tool running on a configuration or synchronization system or device) is provided which allows managing configuration parameters of a synchronization task between two data sources (e.g. a data base synchronization job between two data bases).

[0087] The approach of FIG. 11 for example allows addressing issues of configuration confusion and complexity in synchronization of data based on the configuration file of FlinkX.

[0088] The method of FIG. 11 is for example carried out by a server computer as illustrated in FIG. 12.

[0089] FIG. 12 shows a data synchronization device 1200 according to an embodiment.

[0090] The data synchronization device 1200 is implemented by one or more computers and includes a communication interface 1201 (e.g. configured to transmit data for displaying the graphical user interface on a display, receiving user input, sending control data to the data sources for starting the synchronization task etc.)

[0091] The data synchronization device 1200 further includes a processing unit 1202 and a memory 1203. The memory 1203 may be used by the processing unit 1202 to store, for example, values of the configuration parameters and the generated synchronization task configuration file. The data synchronization device is configured to perform the method of FIG. 11.

[0092] 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.

[0093] 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:
presenting a user with a graphical user interface to input values for configuration parameters for a synchronization task between a first data source and a second data source;
determining values of the configuration parameters at least partially based on user input to the graphical user interface;
generating a synchronization task configuration file by including each determined configuration parameter value into a section of the configuration file assigned to the configuration parameter according to a pre-defined configuration file structure; and
starting a synchronization task in accordance with the generated configuration file, wherein the synchronization task configuration file comprises a section for parameters related to the first data source, a section for parameters related to the second data source and a general section and generating the synchronization task configuration file comprises classifying the determined configuration parameter values according to whether they relate to the first data source, relate to the second data source or are general configuration parameters and putting each configuration parameter value in the section of the synchronization task configuration file to which it relates.
2. The method of claim 1, wherein starting the synchronization task in accordance with the generated configuration file comprises supplying the generated configuration file to a data synchronization framework.
3. The method of claim 1 or 2, wherein the synchronization task comprises a transmission of data from the first data source to the second data source.
4. The method of claim 3, wherein the values of the configuration parameters include the information that the first data source is an origin data source of the transmission and that the second data source is a destination data source of the transmission.

5. The method of any one of claims 1 to 4, wherein the values of the configuration parameters include information about formatting functions to perform on the data for transmitting data from the first data source to the second data source.
6. The method of any one of claims 1 to 5, wherein the values of the configuration parameters include information about the first data source and the second data source.
7. The method of claim 6, wherein the information about the first data source includes an identification of the first data source and wherein the information about the second data source includes an identification of the second data source.
8. The method of any one of claims 1 to 7, wherein determining values of the configuration parameters comprises presenting possible values of the configuration parameters via the graphical user interface for selection and confirmation.
9. The method of one of claim 8, wherein determining values of the configuration parameters comprises receiving an original synchronization task configuration file and parsing the original synchronization task configuration file to determine possible values of the configuration parameters.
10. The method of claim 9, wherein generating the synchronization task configuration file includes keeping configuration parameter values of the original configuration file which have been confirmed by user input to the graphical user interface and updating configuration parameter values which have been changed by user input to the graphical user interface.
11. The method of any one of claims 1 to 10, wherein the pre-defined configuration file structure is a FlinkX configuration file structure.
12. The method of any one of claims 1 to 11, comprising performing the synchronization task using FlinkX.

13. The method of any one of claims 1 to 12, wherein the pre-defined configuration file structure is a JSON format.
14. 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 13.
15. 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 13.
16. 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 13.