

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. However, a data source may itself be implemented in a distributed manner, i.e. include multiple components, each storing a part of the data that the data source stores. Therefore, when data is transmitted from an origin data source to a destination data source for synchronization and the destination data source includes multiple components, the data needs to be correctly distributed to the components.

[0003] Efficient approaches for distributing data in the synchronization of data between data sources 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, wherein the generation includes including values of configuration parameters from which a distribution rule is derivable which specifies how data from a first data source should be distributed to components of a second data source 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, wherein the synchronization framework derives the distribution rule and distributes data to the components of the second data source in accordance with the distribution rule.

[0005] According to one embodiment, the values of the configuration parameters specify one or more data fields of the data depending on which, according to the distribution rule, the data should be distributed to the components of the second data source.

[0006] According to one embodiment, the data includes a plurality of data units according to a predefined data unit format and wherein the values of the configuration parameters specify the one or more data fields by specifying one or more positions within the predefined data unit format.

[0007] According to one embodiment, the data unit format is a database table row.

[0008] According to one embodiment, the synchronization framework parses the data to determine the values of the one or more data fields and distributes the data according to the values.

[0009] According to one embodiment, the data includes a plurality of data units and distributing the data to the components of the second data source in accordance with the distribution rule includes determining, for each data unit, one of the components of the second data source to which the data unit should be distributed and storing the data unit in the determined component.

[0010] According to one embodiment, the values of the configuration parameters specify one or more data fields of the data unit depending on which the data should be distributed to the components of the second data source and determining, for the data unit, the component of the second data source to which the data unit should be distributed includes determining the value or values of the one or more data fields of the data unit and determining the component of the second data source to which the data unit should be distributed in accordance with the determined value or values.

[0011] According to one embodiment, the components of the second data source include data bases of the second data source.

[0012] According to one embodiment, the data bases are relational data bases.

[0013] According to one embodiment, the components of the second data source include data base tables of the second data source.

[0014] According to one embodiment, the data includes a plurality of data units and distributing the data to the components of the second data source in accordance with the distribution rule includes determining, for each data unit, a data base of the second data source and a data base table of the second data source and storing the data unit in the determined data base table of the determined data base.

[0015] According to one embodiment, determining the data base includes identifying one or more first fields of the data unit whose value or values specify the data base in which the

data unit should be stored and determining the data base according to the value or values of the first field and the distribution rule and determining the data base table includes identifying one or more second fields of the data unit whose value or values specify the data base table in which the data unit should be stored and determining the data base table according to the value and values of the second field and the distribution rule.

[0016] According to one embodiment, the value of one of the first fields specifies a country and the value of one of the second fields specifies a time.

[0017] According to one embodiment, the configuration parameters include one or more first configuration parameters which specify, for each data unit, the one or more first fields of the data unit and includes one or more second configuration parameters which specify, for each data unit, the one or more second fields of the data unit and wherein the synchronization framework identifies the one or more first fields according to the one or more first configuration parameters and identifies the one or more second fields according to the one or more second configuration parameters.

[0018] According to one embodiment, the one or more second configuration parameters specify a field of the data unit and intervals, wherein the interval of the intervals in which the value of the field of the data unit lies gives the database table in which the data unit should be stored.

[0019] According to one embodiment, the synchronization framework parses the configuration file to determine the configuration parameters.

[0020] According to one embodiment, the synchronization framework is FlinkX.

[0021] According to one embodiment, the first data source is implemented as one or more Kafka brokers.

[0022] According to one embodiment, the method includes generating the synchronization task configuration file in JSON format.

[0023] According to one embodiment, generating a synchronization task configuration file includes presenting a user with a graphical user interface to input information about the distribution rule and setting the values of the configuration parameters according to user input.

[0024] 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.

[0025] 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.

[0026] 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

[0027] 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 illustrates synchronization where the destination data source is implemented in a distributed manner.
- FIG. 3 shows excerpts of an exemplary configuration file.
- FIG. 4 shows a flow diagram for performing a distribution operation in the transmission from a first data source to a second data source.
- FIG. 5 illustrates further details of data distribution according to an embodiment .
- FIG. 6 shows a graphical user interface including a drop-down menu allowing a user to select information about a data distribution rule.
- FIG. 7 shows a flow diagram illustrating a method for synchronizing data between data sources, for example performed by a synchronization engine.
- FIG. 8 shows a data synchronization device according to an embodiment.

DETAILED DESCRIPTION

[0028] 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.

[0029] 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.

[0030] 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.

[0031] 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.

[0032] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0033] In the following, embodiments will be described in detail.

[0034] FIG. 1 shows a data distribution arrangement 100.

[0035] 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.

[0036] 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.

[0037] 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.

[0038] 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). 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.

[0039] A data synchronisation task, i.e. data transmission 105, between two data sources 101, 102 is denoted as a job.

[0040] While, as explained above, the data sources 101, 102, 103 are synchronized to store the same data, it is possible that a data source (e.g. here the destination data source 102 of the transmission 105) is itself implemented in a distributed manner, i.e. includes multiple data bases which do not store all the complete data of the second data source 101, but together store the data of the second data source 102.

[0041] FIG. 2 illustrates a synchronization where the destination data source 202 (corresponding to data source 102) is implemented in a distributed manner.

[0042] Similarly to the example of FIG 1. There is a transmission 215 for synchronization of data from an origin data source 201 (corresponding to the first data source 101) to the destination data source 202.

[0043] However, the destination data source 202 is implemented in a distributed manner, i.e. is implemented by multiple data bases (e.g. in different geographical locations), in this example three (sub-)data bases 203, 204, 205. This may for example be desirable to support a high number of queries in different geographical locations. Therefore, data transmitted in the data transmission 215 needs to be properly forwarded (i.e. routed) to the correct sub-data base 203, 204, 205.

[0044] Moreover, each data base 203, 204, 205 may be a relational data base and may have multiple tables 206, 207, 208 (in this example three but there may be much more) within each data base 203, 204, 205. Therefore, data transmitted in the data transmission 215 needs also to be put in the correct table 206, 207, 208 within each data base 203, 204, 205.

[0045] According to various embodiments, e.g. in a big data application, (e.g. stream) synchronization of data to a relational databases (i.e. where the destination data source includes a plurality of relational data bases), in particular the distribution to the relational data bases (and possibly tables within the relational data bases) is performed on the synchronization task configuration file for the synchronization framework.

[0046] For example, in FlinkX, a synchronization job is defined by a configuration file which is, in FlinkX, a JOSN file with a certain pre-defined structure.

[0047] The configuration file has the following structure:

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

Table 1: top-level configuration file structure

[0048] The data synchronization task contains one job element, and this element contains two parts of setting and content.

[0049] 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

[0050] Setting may also be used to configure managing of dirty data, logging and restoring.

[0051] 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

[0052] 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.

[0053] In particular, according to various embodiments, the synchronization engine 106 determines on the basis of information given in the configuration file 107 how data should be distributed to data bases 203, 204, 205 and/or data base tables 206, 207, 208. Since a synchronization framework like FlinkX already allows control of a synchronization task based on configuration files, the synchronization of data to a destination data source having multiple data bases and multiple tables may be supported with little additional implementation effort to the case of a destination data source having only a single data base and a single table. This means that the data base and/or table where the data should be stored is configured by means of a configuration file (or script file) for synchronization and no separate task code is required, i.e. the user can define the distribution of data to data bases and tables by adapting the configuration file and does not need to write specific code.

[0054] According to various embodiments, the synchronization engine 106 obtains information about the data base (or library) and the database (or library) table where to store data by means of an inverse function (e.g. internal to the synchronization engine) from information in the configuration file. This means that the data base and the database table where to store the data (also referred to as destination data base and destination (database) table) are not directly defined in the configuration file but the synchronization engine 106 (in context of operating according to the synchronization framework) determines them from synchronization parameter values contained in the configuration file using the inverse function and stores the data in the destination data source accordingly. In other words, the synchronization engine 106 determines destination data base and/or table information from parameter values in the configuration file from which this information is derivable by means of the inverse function.

[0055] FIG. 3 shows excerpts of a configuration file 300 according to an embodiment.

[0056] The configuration file 300 is an example of the configuration file 107 according to an embodiment.

[0057] In the configuration file 107, all parameter that the synchronization engine 106 needs to carry out the synchronization task 105 are defined.

[0058] 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 which includes multiple data bases, wherein each data base has multiple data base tables.

[0059] The configuration file 300 includes configuration parameters 201 from which the destination data base and the destination table of transmitted data is derivable.

[0060] Specifically, the configuration file specifies

"splitTableRange": 6,

"splitTableIndex": 8,

"splitDBIndex": 4,

[0061] This means that in a data unit received from the first data source 201, the value of the field at the fourth position (splitDBIndex) defines the data base 203, 204, 205 to which the content of the data unit should be written and the value of the field at the eighth position (splitTableIndex) defines the table 206, 207, 208 to which the content of the data unit should be written wherein the "splitTableRange" parameter defines the interval after which a new table 206, 207, 208 should be started.

[0062] For example, the fourth field (splitDBIndex) of each data unit defines a country (e.g. the data unit corresponds to an online transaction and the fourth field indicates the country to which a good should be delivered) and each country is associated with one of the data bases 203, 204, 205. Each data base 203, 204, 205 stores data units which contain the associated country at the fourth field.

[0063] The eighth field (splitTableIndex) of each data unit defines a time (e.g. the data unit corresponds to an online transaction and the eighth field indicates the day on which an order was placed) and within each data base 203, 204, 205, data units are stored in tables depending on the time, wherein a new table is begun every six (splitTableRange) months.

[0064] Thus, by parsing the data unit and with knowledge of the parameters splitTableRange, splitTableIndex and splitDBIndex, the synchronization engine 106 can write the data unit to the correct table 206, 207, 208 in the correct data base 203, 204, 205.

[0065] In other words, the configuration file contains values for configuration parameters which specify rules according to which the synchronization engine 106 (when executing the synchronization framework) can correctly route data units (by checking corresponding fields (e.g. request fields) in the data units) to a destination data source having multiple data bases and multiple tables.

[0066] It should be noted that each database 203, 204, 205 may be located in the associated country (such that in particular data is written to data bases in multiple countries) but that is not necessary.

[0067] FIG. 4 shows a flow diagram for performing a distribution operation in the transmission 105 from the first data source 101 to the second data source 102.

[0068] In 401, the synchronization engine 106 (when executing the synchronization framework) 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 300). 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).

[0069] In 402, the synchronization engine 106 parses the data about the request (e.g. the configuration file 300) and determines, in 403, based on the result of the parsing, whether it

needs to distribute data received from the first data source 101 to different data bases and/or tables. If no, it does not perform a distribution but simply supplies the data to the second data source 202.

[0070] If yes, the synchronization engine 106 determines the distribution rules and, in 404, checks, for data of the first data source, whether an initialized data base exists to which (according to the distribution rules) the data should be stored. If no, the synchronization engine 106 performs some error handling, e.g. filters out the data.

[0071] If yes, the synchronization engine 106 checks, in 405, for data of the first data source, whether an initialized data base table exists to which (according to the distribution rules) the data should be stored. If no, the synchronization engine 106 performs some error handling, e.g. filters out the data.

[0072] If yes, it stores the data in the database and data base table according to the distribution rules in 406.

[0073] Instead of filtering out data in case there exists no initialized destination data base and/or destination data base table, the synchronization engine 106 may trigger their initialization. An example is described in the following with reference to FIG. 5.

[0074] FIG. 5 illustrates further details of the data distribution, in particular initialization of data base and data base tables.

[0075] In 501, the configuration file 502 is supplied to the synchronization framework executed on the synchronization engine 503 (e.g. FlinkX).

[0076] From parsing the configuration file in 504 and from parsing received data units in 505 the synchronization engine 503 determines (when executing the synchronisation framework) that certain data bases and/or tables are needed (e.g. a data base for a certain country or a table for a certain range of six months). It may then trigger the data source 202 to initialize corresponding data bases and tables in 506. In other words, it constructs a preparedState of a relational data base to prepare for the data requested to be written. As explained above, the parsing of a received data unit includes analyzing the data unit according to the distribution rules, i.e. obtaining values of data fields from which the distribution depend and determine a data base and data base table from the values in accordance with the distribution rules.

[0077] Then, when the synchronization engine 503 has determined the destination data base 509, 510, 511 in 507 and the destination table 512-520 in 508, it may write the data to the destination data base and the destination table.

[0078] As mentioned above, a user may define the distribution rules (i.e. declare the distribution function) by including suitable configuration parameters into the configuration file 107.

[0079] For this, the synchronization engine 106 (e.g. when executing a configuration tool, i.e. implementing a configuration system) may present the user with a graphical user interface. The user may then enter values for configuration parameters and the synchronization engine 106 generates a corresponding configuration file including the input values.

[0080] FIG. 6 shows a graphical user interface 600 including a drop-down menu 601 allowing a user to select information about a data distribution rule.

[0081] When the user has input the information about the distribution rule, 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 data distribution.

[0082] In summary, according to various embodiments, a method is provided as illustrated in FIG. 7.

[0083] FIG. 7 shows a flow diagram 700 illustrating a method for synchronizing data between data sources, for example performed by a synchronization engine.

[0084] In 701, a synchronization task configuration file for a synchronization framework is generated. In the generation values of configuration parameters from which a distribution rule is derivable are included. The distribution rule specifies how data from a first data source should be distributed to components (i.e. storage components such as memories, storage devices, memory regions or storage data structures) of a second data source.

[0085] In 702, the synchronization framework is controlled to perform a synchronization task between a first data source and a second data source in accordance with the synchronization task configuration file, wherein the synchronization framework derives the

distribution rule and distributes data to the components of the second data source in accordance with the distribution rule.

[0086] According to various embodiments, in other words, distribution of data transmitted from an origin data source to a destination data source which has multiple components (e.g. data bases or data base tables) is supported by means of a synchronisation framework configuration file.

[0087] At least one of the data sources may operate according to Kafka such that the data may be, in one embodiment, seen as Kafka data. The transmission of data from the first data source to the second data source is for example performed using FlinkX.

[0088] The method of FIG. 7 is for example carried out by a server computer as illustrated in FIG. 8.

[0089] FIG. 8 shows a data synchronization device 800 according to an embodiment.

[0090] The data synchronization device 800 is implemented by one or more computers and includes a communication interface 801 (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 forwarded, send out data to be forwarded (while distributing it) or send a data synchronization task configuration file.)

[0091] The data synchronization device 800 further includes a processing unit 802 and a memory 803. The memory 803 may be used by the processing unit 802 to store, for example, the synchronization task configuration file. The data synchronization device is configured to perform the method of FIG. 7. 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).

[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:
generating a synchronization task configuration file for a synchronization framework, comprising including values of configuration parameters from which a distribution rule is derivable which specifies how data from a first data source should be distributed to components of a second data source; 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, wherein the synchronization framework derives the distribution rule and distributes data to the components of the second data source in accordance with the distribution rule.
2. The method of claim 1, wherein the values of the configuration parameters specify one or more data fields of the data depending on which, according to the distribution rule, the data should be distributed to the components of the second data source.
3. The method of claim 2, wherein the data comprises a plurality of data units according to a predefined data unit format and wherein the values of the configuration parameters specify the one or more data fields by specifying one or more positions within the predefined data unit format.
4. The method of claim 3, wherein the data unit format is a database table row.
5. The method of any one of claims 2 to 4, wherein the synchronization framework parses the data to determine the values of the one or more data fields and distributes the data according to the values.
6. The method of any one of claims 1 to 5, wherein the data comprises a plurality of data units and distributing the data to the components of the second data source in accordance with the distribution rule comprises determining, for each data unit, one of

the components of the second data source to which the data unit should be distributed and storing the data unit in the determined component.

7. The method of claim 6, wherein the values of the configuration parameters specify one or more data fields of the data unit depending on which the data should be distributed to the components of the second data source and wherein determining, for the data unit, the component of the second data source to which the data unit should be distributed comprises determining the value or values of the one or more data fields of the data unit and determining the component of the second data source to which the data unit should be distributed in accordance with the determined value or values.
8. The method of any one of claims 1 to 7, wherein the components of the second data source comprise data bases of the second data source.
9. The method of claim 8, wherein the data bases are relational data bases.
10. The method of any one of claims 1 to 9, wherein the components of the second data source comprise data base tables of the second data source.
11. The method of any one of claims 1 to 10, wherein the data comprises a plurality of data units and distributing the data to the components of the second data source in accordance with the distribution rule comprises determining, for each data unit, a data base of the second data source and a data base table of the second data source and storing the data unit in the determined data base table of the determined data base.
12. The method of claim 11, wherein determining the data base comprises identifying one or more first fields of the data unit whose value or values specify the data base in which the data unit should be stored and determining the data base according to the value or values of the first field and the distribution rule and wherein determining the data base table comprises identifying one or more second fields of the data unit whose value or values specify the data base table in which the data unit should be stored and

determining the data base table according to the value and values of the second field and the distribution rule.

13. The method of claim 12, wherein the value of one of the first fields specifies a country and the value of one of the second fields specifies a time.
14. The method of claim 12 or 13, wherein the configuration parameters comprise one or more first configuration parameters which specify, for each data unit, the one or more first fields of the data unit and comprises one or more second configuration parameters which specify, for each data unit, the one or more second fields of the data unit and wherein the synchronization framework identifies the one or more first fields according to the one or more first configuration parameters and identifies the one or more second fields according to the one or more second configuration parameters.
15. The method of claim 14, wherein the one or more second configuration parameters specify a field of the data unit and intervals, wherein the interval of the intervals in which the value of the field of the data unit lies gives the database table in which the data unit should be stored.
16. The method of any one of claims 1 to 13, wherein the synchronization framework is FlinkX.
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 information about the distribution rule and setting the values of the configuration parameters 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.