

COORDINATION SERVER AND METHOD FOR CONTROLLING TRAFFIC BUFFERING DURING DATABASE MIGRATION

TECHNICAL FIELD

[0001] Various aspects of this disclosure relate to coordination servers and methods for controlling traffic buffering during database migration.

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. For example, there may be a need to synchronize data between a source database and a destination database. This makes the synchronization a complex task.

[0003] Traditionally, for data migration systems, traffic buffering occurs at the source or destination database or virtual machine. However, this requires users to understand migration functionality and dependencies of the database software. Accordingly, efficient approaches for synchronizing data between databases are desirable.

SUMMARY

[0004] Various embodiments concern a coordination server for controlling traffic buffering during database migration between a source database and a destination database. The coordination server is configured to control traffic buffering of data by coordinating migration of data between the source database and the destination database by identifying overlap data and non-overlap data between the source database and the destination database, buffer the non-overlap data at one or more clients to obtain buffered traffic data and release the buffered traffic data to the destination database.

[0005] .

[0006] According to one embodiment, the coordination server is configured to migrate the non-overlap data between the source database and the destination database (or not to migrate the overlap data).

[0007] According to one embodiment, the coordination server is configured to check whether the destination database is synchronized with the source database by checking a unique Global transaction identifier (GTID) of the source database.

[0008] According to one embodiment, the GTID comprises a latest data replication state of the source database.

[0009] According to one embodiment, wherein the buffered traffic data release to the destination database is rate limited.

[0010] According to one embodiment, the coordination server is configured to instruct the one or more clients to buffer the non-overlap data for data migration.

[0011] According to one embodiment, the one or more clients is configured to check for existing traffic buffering before buffering the non-overlap data.

[0012] According to one embodiment, wherein the one or more clients is configured to send a traffic buffered acknowledgement to the coordination server after buffering the non-overlap data.

[0013] According to one embodiment, the system is developed using Golang programming language.

[0014] Various embodiments concern a method for using a coordination server to control traffic buffering during database migration between a source database and a destination database comprising: controlling traffic buffering of data using the coordination server by: coordinating migration of data between the source database and the destination database by identifying overlap data and non-overlap data between the source database and the destination database; buffering the non-overlap data at one or more clients to obtain buffered traffic data; release the buffered traffic data to the destination database.

[0015] According to one embodiment, the method further comprising using the coordination server to migrate the non-overlap data between the source database and the destination database.

[0016] According to one embodiment, the method further comprising using the coordination server to check whether the destination database is synchronized with the source database by checking a unique Global transaction identifier (GTID) of the source database.

[0017] According to one embodiment, the GTID comprises a latest data replication state of the source database.

[0018] According to one embodiment, the buffered traffic data release to the destination database is rate limited.

[0019] According to one embodiment, the method further comprising using the coordination server to instruct the one or more clients to buffer the non-overlap data for data migration.

[0020] According to one embodiment, the method further comprising using one or more clients to check for existing traffic buffering before buffering the non-overlap data.

[0021] According to one embodiment, the method further comprising using the one or more clients to send a traffic buffered acknowledgement to the coordination server after buffering the non-overlap data.

[0022] According to one embodiment, a data synchronization device is provided comprising a communication interface, a memory interface and a processing unit configured to perform the method for using a system to control traffic buffering during database migration described above.

[0023] 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 using a system to control traffic buffering during database migration described above.

[0024] 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 using a system to control traffic buffering during database migration described above.

[0025] It should be noted that embodiments described in context of the method for using a system to control traffic buffering during database migration are analogously valid for the system and vice versa.

BRIEF DESCRIPTION OF THE DRAWINGS

[0026] 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 an exemplary system for controlling traffic buffering during database migration according to an embodiment.

- FIG. 2 shows an exemplary method for controlling traffic buffering during database migration according to an embodiment.
- FIG. 3 shows a flow diagram for controlling traffic buffering during database migration according to an embodiment.

DETAILED DESCRIPTION

[0027] 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.

[0028] Embodiments described in the context of one of the systems or methods are analogously valid for the other systems or methods. Similarly, embodiments described in the context of a system are analogously valid for a method, and vice-versa.

[0029] 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.

[0030] 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.

[0031] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0032] In the following, embodiments will be described in detail.

[0033] FIG. 1 shows an exemplary system for controlling traffic buffering during database migration according to an embodiment.

[0034] The system 100 includes a coordination server 104 which may also be referred to as the TopoPilotTM. A user 102 may use the coordination server 104 to control traffic buffering

of data during database migration between a source database 108 and a destination database 110.

[0035] The coordination server 104 is a coordination service to orchestrate the traffic stopping and/or buffering and/or releasing of traffic.

[0036] The system 100 also includes one or more clients 106 such as a first client (Client A) 106A and a second client (Client B) 106B. The clients may also be referred to as Hardy or Hardy clients. The clients may also be database drivers. In various embodiments, the first client 106A and the second client 106B may be controlled by the coordination server 104. The first client 106A and the second client 106B may be responsible for creating migration buffer, blocking traffic, and releasing traffic data for data migration between the source database 108 and the destination database 110.

[0037] Clients 106A and 106B may retrieve data from the source database 108 and the destination database 110. The coordination server 104, the source database 108, the destination database 110 and the clients 106A and 106B may be connected by a communication network (e.g. the Internet) which may be implemented by multiple sub-networks.

[0038] In various embodiments, Clients 106A and 106B may be database clients which may accept traffic and/or read/write to source database 108.

[0039] In various embodiments, the migration of data from the source database 108 to the destination database 110 may be performed by a data migration program 109. The Clients 106A and 106B will buffer the incoming non-overlap traffic until the migrated data in destination database 110 catches up with source database 108. Then Clients 106A and 106B will release the buffered traffic to the destination database 110, by reading or writing to the destination database 110. All the subsequent traffic to Clients 106A and 106B will also go to the destination database 110 and not the source database 108, which completes the migration of data from the source database 108 to the destination database 110.

[0040] In various embodiments, the coordination server 104 is configured to coordinate migration of data between the source database 108 and the destination database 110 by identifying overlap data and non-overlap data between the source database 108 and the destination database 110.

[0041] In various embodiments, the coordination server 104 is configured to buffer the non-overlap data at the first client 106A and/or the second client 106B to obtain buffered traffic data and release the buffered traffic data to the destination database 110.

[0042] In various embodiments, the coordination server 104 is configured to migrate the non-overlap data between the source database and the destination database, or not to migrate the overlap data between the source database and the destination database.

[0043] In various embodiments, buffering occurs at the client 106 rather than at the source 108 or the destination database 110. This allows developers to use third-party or off-the-shelf database software without needing to change the database code or fully understand the migration functionality/dependencies of the database software. Instead, the implementation occurs on the client side where developers have more control and understanding of the code.

[0044] In various embodiments, migration traffic buffer is created between the source 108 and destination database 110 are at client 106 (database driver) level, allowing zero code changes and minimal impacts to business.

[0045] During migration, only traffic that needs to be buffered will be buffered. This means if there are overlapping between source and destination databases routing, traffic hit the overlapped parts will not be buffered, minimizing impacts on irrelevant traffic. For example, in some cases, one or more portions of a database is to be migrated rather than the entire database. In such cases, traffic to the portions of the database to be migrated will be buffered while traffic to portions of the database that will not be migrated may continue to flow. As a result, the database may continue to operate as usual on portions that are not migrating.

[0046] In various embodiments, to minimize the impact on “innocent” traffic, the traffic control gracefully is at table level. This means that migration of a first data table A would not affect traffic on a second data table B. To identify this, the library will generate routing plans (at table level) on both source and destination to identify one or more overlappings.

[0047] In various embodiments, the coordination server 104 is configured to check whether the destination database is synchronized with the source database by checking a unique Global transaction identifier (GTID) of the source database.

[0048] In various embodiments, the GTID comprises a latest data replication state of the source database.

[0049] Internally, there is a data replication pipeline from the source database to the destination. To check whether the destination database is fully synchronized with the source database, the GTID of the source database is queried and then check whether the data replication pipeline has processed the GTID.

[0050] If the GTID has been processed, and no traffic is hitting the source database, all data in the source database is fully synchronized to the destination.

[0051] In various embodiments, the buffered traffic data release to the destination database is rate limited. The buffered traffic release rate subjects to the performance of destination database.

[0052] In various embodiments, the coordination server 104 is configured to instruct the first client and/or the second client to buffer the non-overlap data for data migration.

[0053] In various embodiments, the one or more clients is configured to check for existing traffic buffering before buffering the non-overlap data.

[0054] In various embodiments, the one or more clients is configured to send a traffic buffered acknowledgement to the coordination server after buffering the non-overlap data.

[0055] In various embodiments, wherein the system is developed using Golang programming language. The system may also be developed with any suitable programming language.

[0056] In various embodiments, the system and method disclosed herein relate to the technology domain for database accessing, migration, and scaling with Golang as the programming language. The intended usage is for application maintainers to focus more on developing business logic, spending minimum effort on database scaling and migration.

[0057] According to one embodiment, a data synchronization device is provided comprising a communication interface, a memory interface and a processing unit configured to perform the method for using a system to control traffic buffering during database migration described above.

[0058] 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 using a system to control traffic buffering during database migration described above.

[0059] 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 using a system to control traffic buffering during database migration described above.

[0060] In various embodiments, the system disclosed requires strong client consensus and Coordination since the migration buffer is created per database client rather than in a

centralized place. To achieve migration consensus, the system is coordinated by the coordination server 104.

[0061] In various embodiments, the system requires planning to identify overlaps between the source and destination database routing. The system may generate routing plans on both source and destination to identify overlapping and unblock “innocent” traffic. Innocent traffic may refer to overlap data between the source and destination data. Since data overlaps between the source and the destination, there is no need to migrate the innocent traffic from the source to the destination. To minimize the impact on “innocent” traffic, the traffic control gracefully is at table level. This means that migration/scaling of table A won’t affect traffic on table B. To identify this, the library will generate routing plans (at table level) on both source and destination to identify overlappings.

[0062] In various embodiment, when user 102 issues traffic switch instructions, to minimize the impact on ongoing traffic, the library has to wait for ongoing traffic to finish between reporting traffic block signals. The graceful blocking process is complicated especially with semantic consideration of database transactions.

[0063] In various embodiment, when releasing traffic in buffer to destination database, one needs to rate limit the releasing process to not overwhelm the destination.

[0064] In various embodiment, since the routing plan on source and destination are different, a dry run on the database query on the destination may be needed to verify correctness before switching.

[0065] In various embodiments, one or more abnormal cases, which include, but is not limited to: client failure to buffer traffic, client failure to contact coordinator, unavailability of coordinator. In such abnormal cases, clients will inform the coordinator to abort the process or enter “kill-switch” mode (where all traffic is blocked); to make sure relevant source traffic is totally stopped before switching.

[0066] In various embodiments, transaction needs to be executed fully on source or destination. Executing some statements on source and other statements on destination will break the semantic and cause wrong results. When traffic is stopping, the library has to allow ongoing transactions to finish while blocking other types of traffic.

[0067] FIG. 2 shows an exemplary method for controlling traffic buffering during database migration according to an embodiment.

[0068] As shown in FIG. 2, there may be a method 200 for controlling traffic buffering during database migration. In the method 200, a first step 202 may include coordinating migration of data between a source database and a destination database by identifying overlap data and non-overlap data between the source database and the destination database. A second step 204 may include buffering the non-overlap data at one or more clients to obtain buffered traffic data. A third step 206 may include release the buffered traffic data to the destination database.

[0069] It will be understood that the above operations described above relating to FIG. 2 are not limited to this particular order. Any suitable, modified order of operations may be used.

[0070] FIG. 3 shows a flow diagram for controlling traffic buffering during database migration according to an embodiment.

[0071] The system 300 includes a coordination server 304 which may also be referred to as the TopoPilot. A user 302 may use the coordination server 304 to control traffic buffering of data during database migration between a source database 308 and a destination database 310.

[0072] The coordination server 304 is a coordination service to orchestrate the traffic stopping and/or buffering and/or releasing of data.

[0073] The system 300 also includes one or more clients such as a first client (Client A) 306A and a second client (Client B) 306B. The clients may also be referred to as Hardy or Hardy clients. The clients are also database drivers. In various embodiments, the first client 306A and the second client 306B may be controlled by the coordination server 304. The first client 306A and the second client 306B may be responsible for creating migration buffer, blocking traffic, and releasing traffic data for data migration between the source database 308 and the destination database 310.

[0074] At steps 1.1 to 1.4, the client A 306A and client B 306B query the source database 308 when entering the migration process. Internally, the clients (driver database) employ a counter to monitor each active query. Prior to executing a query, the counter increments, and upon completion of a query, it decrements. The counter serves the purpose of ensuring that there is no ongoing traffic during step 3.

[0075] In various embodiments, the migration buffer may be created at the client 306 (database driver) level. When user 302 instructs traffic switch (step 2), the coordination server 304 (Topo Pilot Coordinator (TPC)) will contact (step 3) and wait (step 4) for all Hardy clients 306 to buffer traffic and acknowledge.

[0076] In various embodiments, buffering occurs at the client 306 rather than at the source 308 or the destination database 310. This allows developers to use third-party or off-the-shelf database software without needing to change the database code or fully understand the migration functionality/dependencies of the database software. Instead, the implementation occurs on the client side where developers have more control and understanding of the code.

[0077] In various embodiments, migration traffic buffer is created between the source 308 and destination database 310 are at client 306 (database driver) level, allowing zero code changes and minimal impacts to business.

[0078] During migration, only traffic that needs to be buffered will be buffered. This means if there are overlapping between source and destination databases routing, traffic hit the overlapped parts will not be buffered, minimizing impacts on irrelevant traffic. For example, in some cases, one or more portions of a database is to be migrated rather than the entire database. In such cases, traffic to the portions of the database to be migrated will be buffered while traffic to portions of the database that will not be migrated may continue to flow. As a result, the database may continue to operate as usual on portions that are not migrating.

[0079] In various embodiments, to minimize the impact on “innocent” traffic, the traffic control gracefully is at table level. This means that migration of a first data table A would not affect traffic on a second data table B. To identify this, the library will generate routing plans (at table level) on both source and destination to identify overlappings.

[0080] Prior to dispatching the TPC buffer traffic command (step 3.1 or step 3.4), TPC will record all participating Hardy clients 306. Step 3.1 and step 3.4 are executed in parallel.

[0081] Upon receiving the buffer traffic command (step 3.2 or step 3.5), the client 306 initiates the buffering of new traffic. Additionally, for existing traffic, the client 306 employs the counter in step 1 to keep track of them. It waits for the counter to reach zero, indicating the completion of all ongoing traffic.

[0082] After buffering the new traffic and waiting for the existing traffic to finish, the client 306 sends a traffic buffered acknowledgement back to TPC. Given that TPC has a record of all participating clients, it ensures that all acknowledgments are received from each client.

[0083] At step 3 when the client 306 buffers traffic, it will compare the execution plans between source and destination database and allow traffic not participated (overlap data) in the database migration.

[0084] When TPC receives signal (step 5) that destination database is fully synchronized with source database, it will instruct client 306 to release buffered traffic.

[0085] Internally, there is a data replication pipeline from the source database to the destination. The source database has a unique Global Transaction Identifier called GTID, which describes the latest data replication state of a database. To check whether the destination database is fully synchronized with the source database, the GTID of the source database is checked and then check whether the data replication pipeline has processed the GTID.

[0086] If the GTID has been processed, and no traffic is hitting the source database, all data in the source database is fully synchronized to the destination.

[0087] Therefore, the clients 306 are instructed to release buffered traffic to destination (step 6). Step 6.1 and step 6.3 are executed in parallel.

[0088] 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.

[0089] 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 coordination server for controlling traffic buffering during database migration between a source database and a destination database, :
wherein the coordination server configured to control traffic buffering of data by

coordinating migration of data between the source database and the destination database by identifying overlap data and non-overlap data between the source database and the destination database, buffer the non-overlap data at one or more clients to obtain buffered traffic data and release the buffered traffic data to the destination database.
2. The coordination server of claim 1, wherein the coordination server is configured to migrate the non-overlap data between the source database and the destination database.
3. The coordination server of claim 1 or claim 2, wherein the coordination server is configured to check whether the destination database is synchronized with the source database by checking a unique Global transaction identifier (GTID) of the source database.
4. The coordination server of claim 3, wherein the GTID comprises a latest data replication state of the source database.
5. The coordination server of claims 1 to 4, wherein the buffered traffic data release to the destination database is rate limited.
6. The coordination server of claims 1 to 5, wherein the coordination server is configured to instruct the one or more clients to buffer the non-overlap data for data migration.

7. The coordination server of claim 6, wherein the one or more clients is configured to check for existing traffic buffering before buffering the non-overlap data.
8. The coordination server of claim 6 or claim 7, wherein the one or more clients is configured to send a traffic buffered acknowledgement to the coordination server after buffering the non-overlap data.
9. The coordination server of claim 1 to 8, wherein the coordination server is developed using Golang programming language.
10. A method for using a coordination server to control traffic buffering during database migration between a source database and a destination database comprising:
controlling traffic buffering of data using the coordination server by:
coordinating the migration of data between the source database and the destination database by identifying overlap data and non-overlap data between the source database and the destination database;
buffering the non-overlap data at one or more clients to obtain buffered traffic data;
release the buffered traffic data to the destination database.
11. The method of claim 10, further comprising using the coordination server migrate the non-overlap data between the source database and the destination database.
12. The method of claim 10 or claim 11, further comprising using the coordination server to check whether the destination database is synchronized with the source database by checking a unique Global transaction identifier (GTID) of the source database.
13. The method of claim 12, wherein the GTID comprises a latest data replication state of the source database.

14. The method of claims 10 to 13, wherein the buffered traffic data release to the destination database is rate limited.
15. The method of claims 10 to 14, further comprising using the coordination server to instruct the one or more clients to buffer the non-overlap data for data migration.
16. The method of claim 15, further comprising using the one or more clients to check for existing traffic buffering before buffering the non-overlap data.
17. The method of claim 15 or claim 16, further comprising using the one or more clients to send a traffic buffered acknowledgement to the coordination server after buffering the non-overlap data.
18. A data migration device comprising a communication interface, a memory interface and a processing unit configured to perform the method of any one of claims 10 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 10 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 10 to 17.