一文了解TiDB的数据对比工具sync-diff-inspector

数据源的TiDB学习之路 发表于 2024-03-18

原创 # 实践案例

近年来诸多行业都开始进行数据库国产化替换的工作，如金融行业的银行、证券等。由于金融行业对数据一致性有严格的要求，在进行国产化替换时一般会选择较为有保障的主备库或多数据中心容灾方案，并且需要定期对上下游库进行数据一致性的比较。

TiDB作为国产化选型中一个十分常用的数据库，产品自带一款用于数据对比的工具sync-diff-inspector，它主要用来检验MySQL/TiDB中的数据一致性，并提供了修复数据的功能。下文具体介绍sync-diff-inspector数据对比工具的相关细节。

一． sync-diff-inspector有哪些能力

sync-diff-inspector是一款数据对比工具，它的主要能力包括：

1. 支持表结构和数据对比
2. 对比不一致时可生成用于数据修复的SQL，但如果表上没有主键或唯一索引时可能无法保证数据正确性
3. 支持上下游不同库名或表名的数据校验
4. 支持MySQL到TiDB的**历史**数据校验，如果是在线校验则需要配置校验某个数据范围
5. 支持TiDB主从集群的**在线**数据校验
6. 支持从TiDB DM拉取配置的数据校验

二． Sync-diff-inspector如何使用

要使用sync-diff-inspector进行数据一致性比较，前提是得有待校验的源库和目标库，如MySQL->TiDB或TiDB->TiDB，再者就是安装配置这个工具并运行对比命令。

安装sync-diff-inspector有两种方式，二进制文件或docker镜像，我们选择下载离线包的方式，在tidb的toolkit工具包中包含sync_diff_inspector可执行文件。

接着我们需要编辑一个配置文件config.toml，**这是最重要的一个部分**，根据具体要校验的内容，需要在配置文件中进行几部分的配置：

1. **通用配置。**如校验并发数、是否输出修复SQL、仅对比表结构还是同时对比数据、是否跳过检验不存在的表。
2. **数据源配置。**包括上下游的数据源实例。
3. **路由规则。**当上下游表名不同时或合库合表场景比对时需要配置一些映射规则。
4. **配置要校验的表。**指定哪些表需要校验。
5. **对具体表的特殊配置。**如指定对比范围、忽略哪些个对比字段等。

具体的配置示例可参考官网 sync-diff-inspector 用户文档 | PingCAP 文档中心

完成配置文件编辑后，便可以通过指定配置文件来运行sync_diff_inspector可执行文件进行数据对比了，具体的命令为 **./sync_diff_inspector --config=./config.toml**

三． Sync-diff-inspector使用实践

1. Mysql->TiDB静态数据校验

首先构造一个MySQL->TiDB的数据对比测试，对比的表在源库和目标库中表结构一致，数据均有5条记录，其中有1条记录不一致。

对比配置文件内容如下，此配置表示对比MySQL和TiDB中表名为test.test1的表结构及数据是否一致、如果不一致输出修复SQL语句。

```
Diff Configuration.

######################### Global config #########################

# 检查数据的线程数量，上下游数据库的连接数会略大于该值
check-thread-count = 4

# 如果开启，若表存在不一致，则输出用于修复的 SQL 语句。
export-fix-sql = true

# 只对比表结构而不对比数据
check-struct-only = false

# 如果开启，会跳过校验上游或下游不存在的表。
skip-non-existing-table = false

######################### Datasource config #########################
[data-sources]
[data-sources.mysql1] # mysql1 是该数据库实例唯一标识的自定义 id，用于下面 task.source-instances/task.target-instance 中
    host = "172.20.12.52"
    port = 3306
    user = "root"
    password = "1[redacted]," # 设置连接上游数据库的密码，可为明文或 Base64 编码。

[data-sources.tidb0]
    host = "172.20.12.53"
    port = 4000
    user = "root"
    password = "r[redacted]" # 设置连接下游数据库的密码，可为明文或 Base64 编码。

######################### Task config #########################
# 配置需要对比的*目标数据库*中的表
[task]
    # output-dir 会保存如下信息
    # 1 sql：检查出错误后生成的修复 SQL 文件，并且一个 chunk 对应一个文件
    # 2 log：sync-diff.log 保存日志信息
    # 3 summary：summary.txt 保存总结
    # 4 checkpoint：a dir 保存断点续传信息
    output-dir = "./output"

    # 上游数据库，内容是 data-sources 声明的唯一标识 id
    source-instances = ["mysql1"]

    # 下游数据库，内容是 data-sources 声明的唯一标识 id
    target-instance = "tidb0"

    # 需要比对的下游数据库的表，每个表需要包含数据库名和表名，两者由 `.` 隔开
    # 使用 ? 来匹配任意一个字符；使用 * 来匹配任意；详细匹配规则参考 golang regexp pkg: https://github.com/google/re2/wiki/Syntax
    target-check-tables = ["test.test1"]
```

执行对比命令，输出内容显示有一张表进行了对比而且对比失败。详细输出内容保存在output/fix-on-tidb0/目录下，对比日志保存在/output/sync_diff.log。

```
[tidb@tidb53 tidb-enterprise-toolkit-v7.1.2-linux-amd64]$ ./sync_diff_inspector --config=./config.toml
A total of 1 tables need to be compared

Comparing the table structure of ``test`.`test1`` ... equivalent
Comparing the table data of ``test`.`test1`` ... failure
_____________________________________________________________________
Progress [============================================================>] 100% 0/0
The data of `test`.`test1` is not equal

The rest of tables are all equal.

A total of 1 tables have been compared, 0 tables finished, 1 tables failed, 0 tables skipped.
The patch file has been generated in
        'output/fix-on-tidb0/'
You can view the comparision details through './output/sync_diff.log'
```

查看output输出目录，checkpoint保存断点续传信息(本示例中为空)，fix-on-target目录保存用于修复不一致的SQL文件，summary.txt保存校验统计结果。查看fix-on-target下的文件可以看出，针对不一致的行生成了一条replace into语句，表示在下游库执行这条SQL可以实现结果与上游库一致。summary.txt文件中输出了上下游数据库的连接信息以及对比统计结果，输出表明所对比的表结构一致，数据有1行不一致(DATA DIFF ROWS +1/-1)。

```
[tidb@tidb53 output]$ cat fix-on-tidb0/test\:test1\:0\:0-128\:0.sql
-- table: test.test1
-- range in sequence: (a) <= (50681)
/*
  DIFF COLUMNS ┆ `c`
╍╍╍╍╍╍╍╍╍╍╍╍╍╍┆╍╍╍╍╍╍
  source data  ┆ 'e'
╍╍╍╍╍╍╍╍╍╍╍╍╍╍┆╍╍╍╍╍╍
  target data  ┆ 'f'
╍╍╍╍╍╍╍╍╍╍╍╍╍╍┆╍╍╍╍╍╍
*/
REPLACE INTO `test`.`test1`(`a`,`b`,`c`) VALUES (5,'f','e');
[tidb@tidb53 output]$ cat summary.txt
Summary



Source Database



host = "172.20.12.52"
port = 3306
user = "root"

Target Databases



host = "172.20.12.53"
port = 4000
user = "root"

Comparison Result



The table structure and data in following tables are equivalent

The following tables contains inconsistent data

+----------------+---------+--------------------+----------------+---------+-----------+
|     TABLE      | RESULT  | STRUCTURE EQUALITY | DATA DIFF ROWS | UPCOUNT | DOWNCOUNT |
+----------------+---------+--------------------+----------------+---------+-----------+
| `test`.`test1` | succeed | true               | +1/-1          |       5 |         5 |
+----------------+---------+--------------------+----------------+---------+-----------+

Time Cost: 196.821975ms
Average Speed: 0.000291MB/s
```

2. TiDB->TiDB动态数据校验

TiDB->TiDB的数据一致性校验相比MySQL->TiDB的校验最主要区别在于它支持在线数据一致性校验，即上游TiDB不断有数据写入情况下的校验。TiDB到TiDB的数据同步最常用的方式是使用TiCDC实时同步工具，我们在文章
初识TiDB的增量数据同步工具TiCDC 介绍了TiCDC的基本用法。在基于TiCDC的主从同步场景中，由于数据是实时变化的，如果要进行数据一致性的比对，需要在TiCDC开启一个特殊的**syncpoint功能**。应用syncpoint功能后，可以对上下游TiDB集群进行一致性快照读和数据一致性校验。

假设我们有两套TiDB集群，且上游集群也安装了TiCDC。首先我们编辑以下changefeed.toml配置文件，它代表打开TiCDC任务中的syncpoint功能。

```
[tidb@tidb53 ~]$ cat changefeed.toml
# 开启 SyncPoint
enable-sync-point = true

# 每隔 5 分钟对齐一次上下游的 snapshot
sync-point-interval = "5m"

# 每隔 1 小时清理一次下游 tidb_cdc.syncpoint_v1 表中的 ts-map 数据
sync-point-retention = "1h"
[tidb@tidb53 ~]$
```

其次，基于上述配置文件创建一个changefeed任务，并使用相关命令查看changefeed，从输出可以看出syncpoint功能已经生效。

```
[tidb@tidb53 ~]$ tiup cdc cli changefeed create --server=http://172.20.12.52:8300 \
> --sink-uri="mysql://root:root123@172.20.12.53:4500/" \
> --changefeed-id="simple-replication-task" \
> --config=./changefeed.toml
tiup is checking updates for component cdc ...
Starting component `cdc`: /home/tidb/.tiup/components/cdc/v7.6.0/cdc cli changefeed create --server=http://172.20.12.52:8300 --sink-uri=mysql://root:root123@1
72.20.12.53:4500/ --changefeed-id=simple-replication-task --config=./changefeed.toml
Create changefeed successfully!
ID: simple-replication-task
Info: {"upstream_id":7342817568238281821,"namespace":"default","id":"simple-replication-task","sink_uri":"mysql://root:xxxxx@172.20.12.53:4500/","create_time"
:"2024-03-18T21:15:23.02753158+08:00","start_ts":448467493963431939,"config":{"memory_quota":1073741824,"case_sensitive":false,"force_replicate":false,"ignore
_ineligible_table":false,"check_gc_safe_point":true,"enable_sync_point":true,"bdr_mode":false,"sync_point_interval":300000000000,"sync_point_retention":360000
0000000,"filter":{"rules":["*.*"]},"mounter":{"worker_num":16},"sink":{"csv":{"delimiter":",","quote":"\"","null":"\\N","include_commit_ts":false,"binary_enco
ding_method":"base64","output_old_value":false},"encoder_concurrency":32,"terminator":"\r\n","date_separator":"day","enable_partition_separator":true,"enable_
kafka_sink_v2":false,"only_output_updated_columns":false,"delete_only_output_handle_key_columns":false,"content_compatible":null,"advance_timeout":150},"consi
stent":{"level":"none","max_log_size":64,"flush_interval":2000,"meta_flush_interval":200,"encoding_worker_num":16,"flush_worker_num":8,"use_file_backend":fals
e,"memory_usage":{"memory_quota_percentage":50,"event_cache_percentage":0}},"scheduler":{"enable_table_across_nodes":false,"region_threshold":100000,"write_ke
y_threshold":0},"integrity":{"integrity_check_level":"none","corruption_handle_level":"warn"},"changefeed_error_stuck_duration":1800000000000,"sql_mode":"ONLY
_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION","synced_status":{"synce
d_check_interval":300,"checkpoint_interval":15}},"state":"normal","creator_version":"v7.5.1","resolved_ts":448467493963431939,"checkpoint_ts":4484674939634319
39,"checkpoint_time":"2024-03-18 21:15:22.944"}
[tidb@tidb53 ~]$ tiup cdc:v7.5.1 cli changefeed list --server=http://172.20.12.52:8300
Starting component `cdc`: /home/tidb/.tiup/components/cdc/v7.5.1/cdc cli changefeed list --server=http://172.20.12.52:8300
[
  {
    "id": "simple-replication-task",
    "namespace": "default",
    "summary": {
      "state": "normal",
      "tso": 448467495326580739,
      "checkpoint": "2024-03-18 21:15:28.144",
      "error": null
    }
  }
]
```

```
[tidb@tidb53 ~]$ tiup cdc:v7.5.1 cli changefeed query --server=http://172.20.12.52:8300 --changefeed-id=simple-replication-task
Starting component `cdc`: /home/tidb/.tiup/components/cdc/v7.5.1/cdc cli changefeed query --server=http://172.20.12.52:8300 --changefeed-id=simple-replication
-task
{
  "upstream_id": 7342817568238281821,
  "namespace": "default",
  "id": "simple-replication-task",
  "sink_uri": "mysql://root:xxxxx@172.20.12.53:4500/",
  "config": {
    "memory_quota": 1073741824,
    "case_sensitive": false,
    "force_replicate": false,
    "ignore_ineligible_table": false,
    "check_gc_safe_point": true,
    "enable_sync_point": true,
    "bdr_mode": false,
    "sync_point_interval": 300000000000,
    "sync_point_retention": 360000000000,
    "filter": {
      "rules": [
        "*.*"
      ]
    },
    "mounter": {
      "worker_num": 16
    },
    "sink": {
      "delete_only_output_handle_key_columns": null,
      "advance_timeout": 150
    },
```

实际上，在TiCDC启用syncpoint功能后，下游的TIDB集群会自动创建数据库**tidb_cdc**并在tidb_cdc库下创建**syncpoint_v1**表，表中的数据记录着上游快照TSO(primary_ts)及下游快照TSO(secondary_ts)。

```
mysql> use tidb_cdc;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+--------------------+
| Tables_in_tidb_cdc |
+--------------------+
| syncpoint_v1       |
+--------------------+
1 row in set (0.00 sec)

mysql> select * from tidb_cdc.syncpoint_v1;
+------------------+------------------------+--------------------+--------------------+---------------------+
| ticdc_cluster_id | changefeed             | primary_ts         | secondary_ts       | created_at          |
+------------------+------------------------+--------------------+--------------------+---------------------+
| default          | simple-replication-task | 448467493963431939 | 448467494054920205 | 2024-03-18 21:15:23 |
+------------------+------------------------+--------------------+--------------------+---------------------+
1 row in set (0.00 sec)
```

基于上下游TSO，我们可以让sync-diff-inspector分别对上下游进行一致性快照读，如果数据校验正常，代表基于此快照的上下游数据是完全一致的。当然，需要在数据库配置中额外增加一项snapshot配置，如下图所示**snapshot = "auto"**, 代表使用 TiCDC 在上下游的同步时间点。

```
# Diff Configuration.

######################### Global config #########################

# 检查数据的线程数量，上下游数据库的连接数会略大于该值
check-thread-count = 4

# 如果开启，若表存在不一致，则输出用于修复的 SQL 语句。
export-fix-sql = true

# 只对比表结构而不对比数据
check-struct-only = false

# 如果开启，会跳过校验上游或下游不存在的表。
skip-non-existing-table = false

######################### Datasource config #########################
[data-sources]
[data-sources.tidb1] # mysql1 是该数据库实例唯一标识的自定义 id，用于下面 task.source-instances/task.target-instance 中
    host = "172.20.12.53"
    port = 4000
    user = "root"
    password = "root123" # 设置连接上游数据库的密码，可为明文或 Base64 编码。
    snapshot = "auto"

[data-sources.tidb2]
    host = "172.20.12.53"
    port = 4500
    user = "root"
    password = "root123" # 设置连接下游数据库的密码，可为明文或 Base64 编码。
    snapshot = "auto"

######################### Task config #########################
# 配置需要对比的*目标数据库*中的表
[task]
    # output-dir 会保存如下信息
    # 1 sql：检查出错误后生成的修复 SQL 文件，并且一个 chunk 对应一个文件
    # 2 log：sync-diff.log 保存日志信息
    # 3 summary：summary.txt 保存总结
    # 4 checkpoint：a dir 保存断点续传信息
    output-dir = "./output"

    # 上游数据库，内容是 data-sources 声明的唯一标识 id
    source-instances = ["tidb1"]

    # 下游数据库，内容是 data-sources 声明的唯一标识 id
    target-instance = "tidb2"

    # 需要比对的下游数据库的表，每个表需要包含数据库名和表名，两者由 `.` 隔开
    # 使用 ? 来匹配任意一个字符，使用 * 来匹配任意，详细匹配规则参考 golang regexp pkg: https://github.com/google/re2/wiki/Syntax
    target-check-tables = ["test.test1"]
```

执行数据校验后，通过查看output/summary.txt并对比tidb_cdc.syncpoint_v1发现，每次数据对比时均按照当前TiCDC所记录的最大的上下游TSO来进行快照一致性对比，tidb_cdc.syncpoint_v1表中的数据也在不断变化累加。

```
[tidb@tidb53 tidb-enterprise-toolkit-v7.1.2-linux-amd64]$ ./sync_diff_inspector --config=./config_t2t.toml
A total of 1 tables need to be compared

Comparing the table structure of ``test`.`test1`` ... equivalent
Comparing the table data of ``test`.`test1`` ... equivalent

Progress [============================================================>] 100% 0/0
A total of 1 table have been compared and all are equal.
You can view the comparision details through './output/sync_diff.log'
[tidb@tidb53 tidb-enterprise-toolkit-v7.1.2-linux-amd64]$
[tidb@tidb53 tidb-enterprise-toolkit-v7.1.2-linux-amd64]$
[tidb@tidb53 tidb-enterprise-toolkit-v7.1.2-linux-amd64]$ cat output/summary.txt
Summary


Source Database

host = "172.20.12.53"
port = 4000
user = "root"
snapshot = "448467887179431936"

Target Databases

host = "172.20.12.53"
port = 4500
user = "root"
snapshot = "448467887493742594"

Comparison Result


The table structure and data in following tables are equivalent

+----------------+---------+-----------+
|     TABLE      | UPCOUNT | DOWNCOUNT |
+----------------+---------+-----------+
| `test`.`test1` |       6 |         6 |
+----------------+---------+-----------+


Time Cost: 67.001942ms
Average Speed: 0.001025MB/s
```

```
mysql> select * from tidb_cdc.syncpoint_v1 order by created_at desc;
+-----------------+----------------------+-------------------+-------------------+---------------------+
| ticdc_cluster_id | changefeed           | primary_ts        | secondary_ts      | created_at          |
+-----------------+----------------------+-------------------+-------------------+---------------------+
| default          | simple-replication-task | 448467887179431936 | 448467887493742594 | 2024-03-18 21:40:24 |
| default          | simple-replication-task | 448467808536231936 | 448467808745947137 | 2024-03-18 21:35:23 |
| default          | simple-replication-task | 448467729893031936 | 448467730128699393 | 2024-03-18 21:30:23 |
| default          | simple-replication-task | 448467651249831936 | 448467651564142594 | 2024-03-18 21:25:24 |
| default          | simple-replication-task | 448467572606631936 | 448467572894990337 | 2024-03-18 21:20:24 |
| default          | simple-replication-task | 448467493963431939 | 448467494054920205 | 2024-03-18 21:15:23 |
+-----------------+----------------------+-------------------+-------------------+---------------------+
6 rows in set (0.01 sec)
```

四． sync-diff-inspector使用小结

本文较完整的概述了TiDB之sync-diff-inspector数据对比工具，并通过实际案例演示如果使用这款工具进行MySQL->TiDB的静态数据比对以及TiDB->TiDB的动态数据比对，了解到要使用TiDB->TiDB的动态数据比对需要开启TiCDC中的syncpoint功能。由于本篇幅只是一个基础的知识了解，涉及到各种复杂的配置未作过多描述，读者可基于TiDB官方文档查看详细资料。