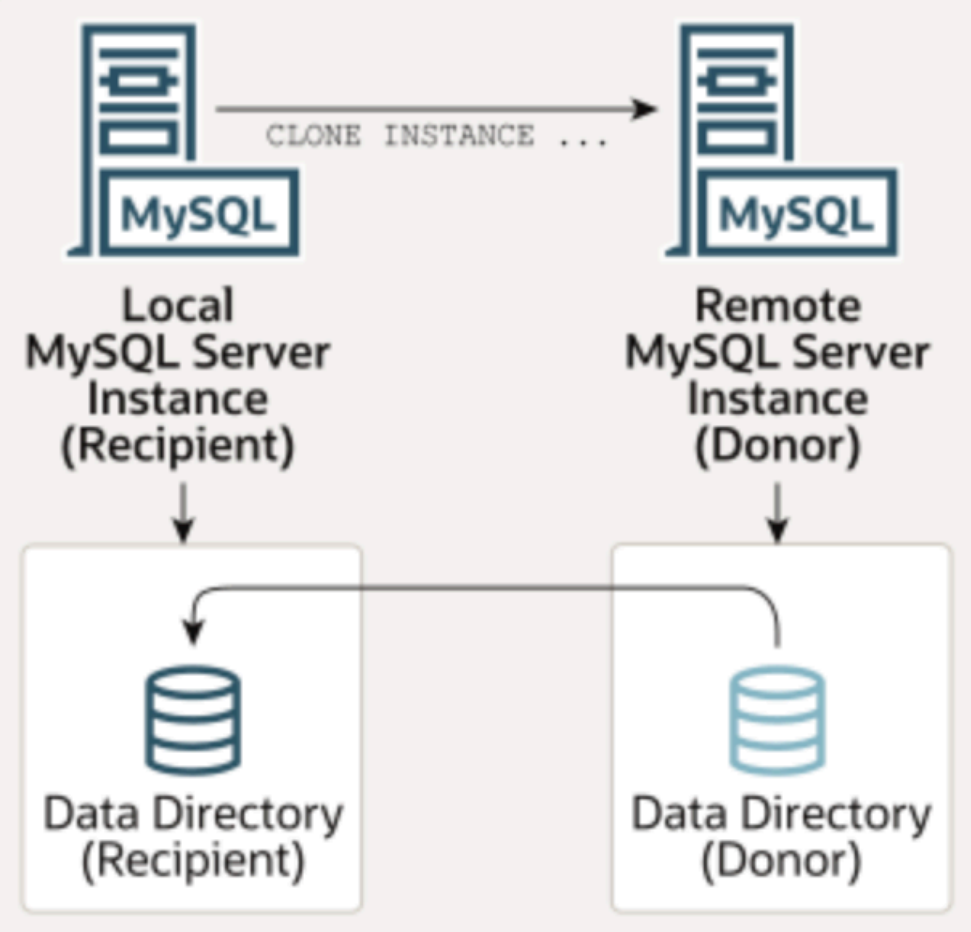
利用 MySQL 8.0 clone 插件远程克隆快速重建主从复制环境

原创 Ye 数据库搬山工 2025年03月10日 18:16 广东

01 概述

近期，某个项目组发现他们项目上的 MySQL 主从复制集群出现同步故障。由于主库的 binlog 日志仅保留 7 天，故障持续时间过长导致日志被清理。经过分析，我们只能通过主库备份恢复从库，并重新建立同步关系。考虑到数据库超过 50GB，且版本为 MySQL 8.0.40，最终决定采用 MySQL 8.0 的克隆插件，通过远程克隆快速重建主从复制，恢复主从数据实时同步。

MySQL 8.0.17+ 引入的远程克隆功能，可以通过网络直接从主实例克隆数据到目标实例，从而快速搭建主从复制环境。远程克隆涉及两个实例：被克隆的实例称为 **Donor**，接受克隆数据的实例称为 **Recipient**。克隆命令需在 Recipient 上发起。远程克隆的原理图如下所示：



02 环境介绍

本文描述的环境为模拟环境，仅用于测试方案可行性。

- 操作系统： Red Hat Enterprise Linux release 8.10 (Ootpa)
- 数据库版本：8.0.40 MySQL Community Server – GPL

主机名	IP 地址	数据库角色
mm-host01	192.168.3.31	主库

mm-host02	192.168.3.32	从库

MySQL主从复制搭建请参考此篇：MySQL双主+Keepalived高可用：搭建指南

03 Clone Plugin的安装

Clone Plugin 可通过以下两种方式安装：

- 配置文件中指定：

```
[mysqld]
plugin-load-add=mysql_clone.so
```

- 动态加载：

```
mysql> install plugin clone soname 'mysql_clone.so';
```

执行 show plugins 查看插件是否安装成功。

```
mysql> show plugins;
+--------------------------------+----------+--------------------+----
------------+---------+
| Name                           | Status   | Type               |
 Library        | License |
+--------------------------------+----------+--------------------+----
------------+---------+
...
| clone                          | ACTIVE   | CLONE              |
 mysql_clone.so | GPL     |
...
```

Status 为 ACTIVE 代表插件加载成功。

04 克隆数据库

- 创建克隆用户并授权

在主从复制正常情况下仅在主库执行即可，主从复制异常，已停止同步情况下，主从库都要创建。

```
mysql -uroot -p
password:strong_password

createuser'clone_user'@'192.168.3.%' identified by'strong_password';
grant BACKUP_ADMIN on*.*to'clone_user'@'192.168.3.%';
grant CLONE_ADMIN on*.*to'clone_user'@'192.168.3.%';
grant SYSTEM_VARIABLES_ADMIN on*.*to'clone_user'@'192.168.3.%';
```

- 执行克隆任务：

在从库上执行克隆命令，需要使用克隆用户登录数据库，如下：

```
mysql -uclone_user -p -h192.168.3.32
```

- 设置克隆源，将clone_valid_donor_list设置为主库

```
SET GLOBAL clone_valid_donor_list = '192.168.3.31:3306';
```

- 开始克隆

```
CLONE
INSTANCE FROM 'clone_user'@'192.168.3.31':3306 IDENTIFIED BY 'strong_pa
ssword';
```

注意：在执行克隆时，会先将从库本地的数据目录清空，请确保本地数据目录没有重要数据，可以清空，否则将导致本地数据丢失。克隆完成后，从库MySQL实例自动重启。

- 从库重启完成后，查看Binlog文件、位置和已经执行的GTID。

```
SELECT                                                    BINLOG_FILE,
BINLOG_POSITION FROM performance_schema.clone_status;
SELECT @@GLOBAL.GTID_EXECUTED;
```

- 重建复制

```
mysql -uroot -p
slave> stop slave;
slave> reset slave all;

CHANGE MASTER TO
MASTER_HOST='192.168.3.31',
MASTER_USER='repl',
MASTER_PASSWORD='strong_password',
MASTER_PORT=3306,
MASTER_AUTO_POSITION=1;
```

- 启动复制，查看复制状态

```
start slave;
show slave status\G
```

05 克隆过程管理

- 克隆过程中的状态查询

克隆操作状态：Not Started（尚未开始）、In Progress（进行中）、Completed（已完成）、Failed（失败）。若为Failed状态，ERROR_NO和ERROR_MESSAGE会给出具体错误编码和信息。

```
SQL> SELECT STATE, ERROR_NO, ERROR_MESSAGE FROM performance_schema.clone_s
+-----------+----------+---------------+
| STATE     | ERROR_NO | ERROR_MESSAGE |
+-----------+----------+---------------+
| Completed |        0 |               |
+-----------+----------+---------------+
1 row in set (0.01 sec)
```

- 克隆（clone）操作的进度查询

```
SQL> SELECT
    stage,
    state,
    CAST(begin_time AS DATETIME) AS "START TIME",
    CAST(end_time AS DATETIME) AS "FINISH TIME",
    LPAD(
        sys.format_time(
            POWER(10, 12) * (UNIX_TIMESTAMP(end_time) - UNIX_TIMESTAMP(beg
        ),
        10,
        ' '
    ) AS DURATION,
    LPAD(
        CONCAT(
            FORMAT(ROUND(estimate /1024/1024, 0), 0),
            "MB"
        ),
        16,
        ' '
    ) AS "Estimate",
    CASE
        WHEN begin_time ISNULLTHEN LPAD('%0', 7, ' ')
        WHEN estimate >0THEN LPAD(CONCAT(ROUND(data *100/ estimate, 0), "%
        WHEN end_time ISNULLTHEN LPAD('0%', 7, ' ')
        ELSE LPAD('100%', 7, ' ')
    ENDAS "Done(%)"
FROM
    performance_schema.clone_progress;

+-----------+-----------+--------------------+--------------------+-----
```

```
| stage     | state     |STARTTIME           | FINISH TIME          | DURATI
+-----------+-----------+--------------------+----------------------+-----
|DROP DATA  | Completed |2025-03-1101:11:55|2025-03-1101:11:55|   97.63 ms
| FILE COPY| Completed |2025-03-1101:11:55|2025-03-1101:12:04|        9 s
| PAGE COPY| Completed |2025-03-1101:12:04|2025-03-1101:12:04|130.69 ms |
| REDO COPY| Completed |2025-03-1101:12:04|2025-03-1101:12:04|130.81 ms |
| FILE SYNC | Completed |2025-03-1101:12:04|2025-03-1101:12:05|613.86 ms
| RESTART   | Completed |2025-03-1101:12:05|2025-03-1101:12:08|     2.93 s
| RECOVERY  | Completed |2025-03-1101:12:08|2025-03-1101:12:09|980.35 ms
+-----------+-----------+--------------------+----------------------+-----
7rowsinset (0.00 sec)
```

字段说明：

字段	说明
STAGE	克隆操作的阶段，包括 DROP DATA、FILE COPY、PAGE COPY、REDO COPY、FILE SYNC、RESTART、RECOVERY 等。
STATE	当前阶段的状态，包括 Not Started、In Progress、Completed。
START TIME	当前阶段的开始时间，格式化为 `DATETIME` 。
FINISH TIME	当前阶段的结束时间，格式化为 `DATETIME` 。
DURATION	当前阶段的持续时间，格式化为可读的时间格式（如 `1h 30m 15s` ）。
Estimate	预估的数据量，格式化为 `MB` 单位。
Done(%)	当前阶段的完成百分比，根据 `estimate` 和 `data` 字段动态计算。

- 如何停止远程克隆

查询克隆操作的PID：Processlist ID。对应show processlist中的ID。

```
SQL> select * from performance_schema.clone_status\G
```

如果要终止当前的克隆操作，执行kill processlist_id命令即可。

```
SQL> Kill+id号;
```

06 克隆插件的实现细节

- 克隆插件的实现可以划分为五个主要阶段：

1. **INIT 阶段**

   初始化一个克隆对象，记录当前 CHECKPOINT 的 LSN，作为后续操作的参考。

2. **FILE COPY 阶段**

   将捐赠者的 InnoDB 数据文件进行物理拷贝，同时在拷贝前启动“Page Tracking”，记录克隆开始后的数据页修改。在拷贝完成后，记录一个“CLONE FILE END LSN”。

3. **PAGE COPY 阶段**

   对在 FILE COPY 阶段期间发生修改的页面进行补传，插件会对这些页面按表空间（tablespace）和页号排序，以减少随机读写；同时启动“Redo Archiving”，归档后续产生的 redo 日志。

4. **REDO COPY 阶段**

   复制归档文件中，从“CLONE FILE END LSN”到“CLONE LSN”之间的 redo 日志，保证目标实例数据的一致性。

5. **Done 阶段**

   调用 snapshot_end() 销毁克隆对象，完成整个克隆流程。

07 使用限制

类别	限制条件
版本要求	MySQL 版本需大于等于 8.0.17，不支持跨版本克隆，要求主、从实例版本一致。
操作限制	1. 克隆过程中不允许执行 DDL 操作。 2. 捐赠者（Donor）的 undo 表空间文件名称不能重复。 3. 克隆操作不会迁移以下内容：my.cnf 配置文件、binlog 二进制日志。
平台和系统要求	两台机器必须运行相同的操作系统，平台和架构需一致。
存储引擎支持	仅支持 InnoDB 引擎的数据克隆。对于 MyISAM 和 CSV 存储的表（包括 sys 模式中的表），将被克隆为空表。
实例配置	1. 两台 MySQL 实例的 innodb_page_size 和 innodb_data_file_path（ibdata 文件名）必须相同。 2. 目标实例（Recipient）需正确设置变量 clone_valid_donor_list。 3. 捐赠者和接受者都需要预先安装克隆插件。

08 总结

MySQL8 的克隆插件提供了一种内置且高效的数据复制方式，简化了从库或组复制节点的搭建过程。它通过 FILE COPY、PAGE COPY 与 REDO COPY 等阶段保证了数据的一致性和快速恢复，同时也有一些使用上的限制（如版本、存储引擎、DDL 限制等），建议根据实际场景选择使用。

END