**新账户系统数据库迁方案**

**成都天府通金融服务股份有限公司**

**研发部**

**文档信息**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本编号: | V1.0.0 | 文档版本日期: | 2020年6月04日 |
| 起草人: | 龚长华 | 起草日期: | 2020年6月04日 |

**版本记录**

| 版本编号 | 版本日期 | 创建/修改 | 说明 |
| --- | --- | --- | --- |
| V1.0.0 | 2020-06-04 | 龚长华 | 初稿 |
| V1.1.0 | 2020-06-05 | 龚长华 | 增加风险描述 |
|  |  |  |  |
|  |  |  |  |

目录

[1. 迁移背景 4](#_Toc5036)

[2. 环境介绍 4](#_Toc1394)

[2.1. 当前数据库集群架构 4](#_Toc6757)

[2.2. 新建数据库集群架构 5](#_Toc4782)

[2.3. 迁移环境说明 5](#_Toc25245)

[3. 总体设计 6](#_Toc28127)

[4. 迁移准备 7](#_Toc10904)

[4.1. 安装XTRABACKUP 7](#_Toc10719)

[4.2. 安装PERCONA-TOOLKIT 7](#_Toc22756)

[5. 迁移过程 7](#_Toc7603)

[5.1. 新建MYSQL实例 7](#_Toc4190)

[5.2. 搭建B-->B1复制链路① 7](#_Toc4363)

[5.3. 搭建B-->B2复制链路② 10](#_Toc28972)

[5.4. 搭建B1-->B2复制链路③ 10](#_Toc9663)

[5.5. 搭建B2-->B1复制链路④ 11](#_Toc22224)

[5.6. 新集群复制拓扑验证 17](#_Toc11827)

[5.7. 其他事项 17](#_Toc9082)

[6. 迁移验证 18](#_Toc24513)

[7. 风险评估 18](#_Toc27461)

# 迁移背景

天府通账户系统（简称新账户系统）经过重构后，需经过试运行阶段，试运行通过后才能完全替代原账户系统。在试运行阶段，新账户系统应用、新账户系统数据库均是独立并行、互不干涉，故需将原账户系统数据库数据迁移至新账户系统数据库数据，以满足新账户系统运行。

原账户系统数据库采用银联闭源的UPSQL和UPPROXY组合架构，新账户系统数据库采用开源的MySQL5.7和KEEPALIVED的架构，旨在解决闭源技术带来的不可控风险。

# 环境介绍

## 当前数据库集群架构

当前，银联承建的账户系统有三套数据库，分别是TFTACTDB, TFTMGMDB, TFTHISDB，均是采用两台UPSQL和两台UPPROXY。两台UPSQL基于传统的GTID模式互为主从的双向复制，两台UPPROXY作为UPSQL的代理，以主备模式运行，保证UPSQL集群的高可用，大致架构图如图1.

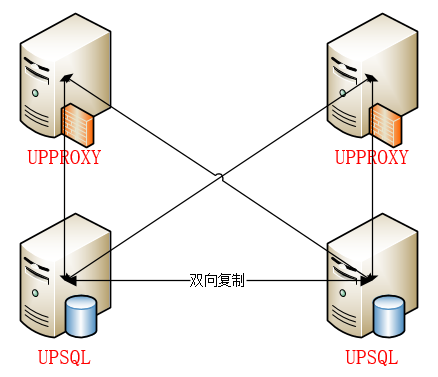


图 1

## 新建数据库集群架构

新建数据库集群，分为三套，分别是TFTACTDB, TFTMGMDB, TFTHISDB，采用两台MYSQL社区版5.7作为数据库集群节点。两台MYSQL基于传统的GTID模式互为主从的双向复制，分别在两台MYSQL节点上部署KEEPALIVED服务，以主备模式运行，保证MYSQL集群的高可用，大致架构图如图2.

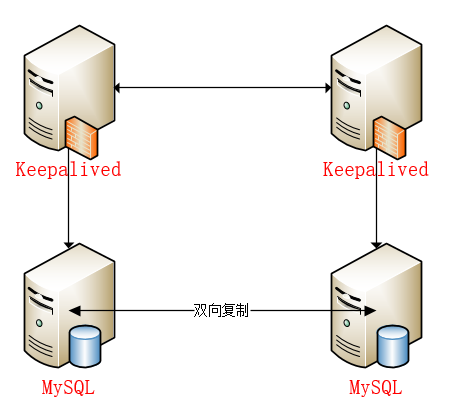


图 2

## 迁移环境说明

**原环境**

|  |  |  |
| --- | --- | --- |
| 服务器IP | 操作系统 | 主要软件 |
| 88.88.16.123(A服务) | CentOS Linux release 7.3.1611 (Core) | 5.7.17-upsql-2.0.3-log UPSQL Server |
| 88.88.16.112(B) | CentOS Linux release 7.4.1708 (Core) | 5.7.17-upsql-2.0.3-log UPSQL Server |

表 1

说明：其中A节点正在对外提供读写服务，B节点作为备用节点备用。

**新环境**

|  |  |  |
| --- | --- | --- |
| 服务器IP | 操作系统 | 主要软件 |
| 88.88.16.125(B1) | CentOS Linux release 7.3.1611 (Core) | 5.7.20-log MySQL Community Server (GPL); Keepalived v2.0.18 |
| 88.88.16.126(B2) | CentOS Linux release 7.3.1611 (Core) | 5.7.20-log MySQL Community Server (GPL); Keepalived v2.0.18 |

表 2

# 总体设计

为保证不影响线上应用正常运行，我们从B节点（备用）复制数据，迁移总体设计，参考[图3](#图3)。

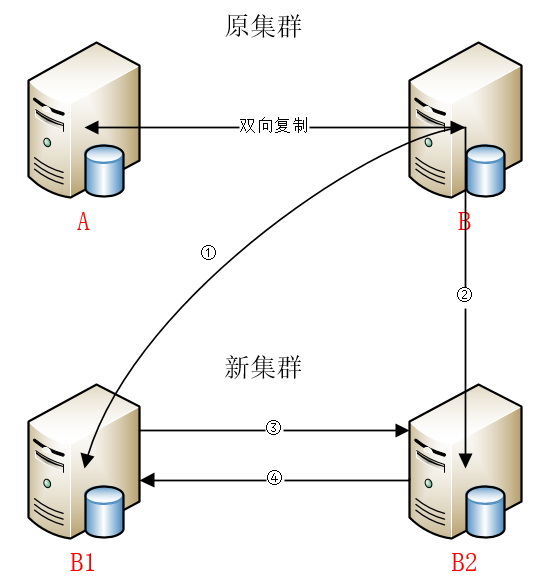


图 3

说明：

1. 箭头表示数据复制方向。

大致迁移步骤如下：

1. 新建MYSQL实例B1，充当B的从节点，进行复制数据；
2. 新建MYSQL实例B2，充当B的从节点，进行复制数据；
3. 修改B2实例的复制Master（B），将Master指向B1实例；
4. 在无业务且复制无延迟的前提下，修改B1实例的复制Master（B），将Master指向B2实例；
5. 业务访问数据库指向B1，完成业务切换。

# 迁移准备

## 安装XTRABACKUP

在环境所有服务器中安装XTRABACKUP工具。

yum install -y https://repo.percona.com/yum/percona-release-latest.noarch.rpm

percona-release setup ps80

yum install -y percona-xtrabackup-24

xtrabackup --version

## 安装PERCONA-TOOLKIT

在环境所有服务器中安装percona-toolkit工具。

yum install percona-toolkit-3.1.0-2.el7.x86\_64.rpm

# 迁移过程

## 新建MYSQL实例

略。

## 搭建B-->B1复制链路①

**在原集群B节点使用actdata（dba用户）进行热备**

[actdata@mkmsapp123 /home/actdata]$ innobackupex --defaults-file=/etc/myconf/actdata.cnf --host=88.88.16.123 --user=root --password=root --port=60001 --slave-info /home/actdata/backup

**热备后，应用日志**

[actdata@mkmsapp123 /home/actdata]$ innobackupex --apply-log /home/actdata/backup/2020-05-29\_16-56-39

**将应用日志后的文件复制到目标库机器B1**

[actdata@mkmsapp123 /home/actdata/backup/2020-05-29\_16-56-39]$ scp -rp 2020-05-29\_16-56-39 root@88.88.16.125:/root/dba/backup

**在目标机器B1停止MySQL（root用户），并删除数据目录，以备还原**

[root@kettle backup]# rm -rf /data/mysqldata/3306/data/\*

**结合Xtrabackup备份的backup-my.cnf文件和UPSQL配置文件，在B1上修改/etc/my.cnf**

**配置文件中参数，根据机器硬件做不同的调整**

[mysqld]

innodb\_checksum\_algorithm=crc32

#innodb\_log\_checksum\_algorithm=strict\_crc32

innodb\_data\_file\_path=ibdata1:12M:autoextend

#innodb\_fast\_checksum=false

innodb\_page\_size=16384

#innodb\_log\_block\_size=512

innodb\_undo\_directory=/data/mysqldata/3306

#redo\_log\_version=1

#master\_key\_id=0

server\_id = 887

slow\_query\_log\_file = /data/mysqldata/3306/log/slow\_query.log

datadir=/data/mysqldata/3306/data

socket=/data/mysqldata/3306/mysql.sock

user=mysql

# Disabling symbolic-links is recommended to prevent assorted security risks

symbolic-links=0

log\_bin=/data/mysqldata/3306/binlog/mysql-bin.log

log\_bin\_index=/data/mysqldata/3306/binlog/mysql-bin.index

binlog\_format = row

#binlog\_row\_image = minimal

relay\_log=/data/mysqldata/3306/log/mysql-relay-bin.log

port=3306

default\_time\_zone = "+08:00"

character\_set\_server = utf8mb4

transaction\_isolation = READ-COMMITTED

explicit\_defaults\_for\_timestamp = 1

max\_allowed\_packet = 16M

lower\_case\_table\_names = 1

open\_files\_limit = 10240

secure\_file\_priv = ""

#plugin\_load = "upsql\_auth=upsql\_auth.so;rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so"

plugin\_load = "rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so"

# connection settings #

interactive\_timeout = 31536000

wait\_timeout = 31536000

lock\_wait\_timeout = 3600

skip\_name\_resolve = 1

max\_connections = 5000 # ulimit -n >= 5810

thread\_cache\_size = 1536

thread\_stack = 512K

max\_connect\_errors = 50000

connect\_timeout = 60

# table cache performance settings #

table\_open\_cache = 4096

table\_definition\_cache = 4096

table\_open\_cache\_instances = 64

# session memory settings #

read\_buffer\_size = 8M # 影响ORDER BY性能，缓存嵌套查询结果

read\_rnd\_buffer\_size = 16M # 提升ORDER BY性能

sort\_buffer\_size = 16M # 提升ORDER BY和GROUP BY性能

join\_buffer\_size = 16M # 联表查询且无索引可用时使用

tmp\_table\_size = 32M # 复杂查询使用内存临时表大小，超过限制则会创建临时文件

key\_buffer\_size = 32M

query\_cache\_type = 0

query\_cache\_size = 0

# log settings #

slow\_query\_log = 1

log\_queries\_not\_using\_indexes = 1

log\_slow\_admin\_statements = 1

log\_slow\_slave\_statements = 1

log\_throttle\_queries\_not\_using\_indexes = 10

expire\_logs\_days = 10

log\_bin\_trust\_function\_creators = 1

binlog\_cache\_size = 4M

max\_binlog\_cache\_size = 4G

binlog\_gtid\_simple\_recovery = 1

log\_timestamps = system

back\_log = 1500

long\_query\_time=2

# innodb settings #

#innodb\_data\_file\_path = ibdata1:1G:autoextend

innodb\_buffer\_pool\_size = 6G

innodb\_buffer\_pool\_instances = 8

innodb\_log\_file\_size = 512M

innodb\_log\_files\_in\_group = 7

innodb\_lru\_scan\_depth = 4096

innodb\_lock\_wait\_timeout = 60

innodb\_io\_capacity = 2000

innodb\_io\_capacity\_max = 4000

innodb\_flush\_method = O\_DIRECT

innodb\_undo\_tablespaces = 3

innodb\_thread\_concurrency = 8 # cores \* 2

innodb\_write\_io\_threads = 2 # cores / 2

innodb\_read\_io\_threads = 2 # cores / 2

innodb\_print\_all\_deadlocks = 1

innodb\_sort\_buffer\_size = 64M

innodb\_stats\_persistent\_sample\_pages = 64

innodb\_online\_alter\_log\_max\_size = 4G

innodb\_log\_buffer\_size = 128M

innodb\_rollback\_on\_timeout = ON

innodb\_purge\_threads = 8

innodb\_page\_cleaners = 8

innodb\_undo\_log\_truncate = 1

innodb\_sync\_spin\_loops = 30

innodb\_spin\_wait\_delay = 6

# replication settings #

gtid\_mode = ON

enforce\_gtid\_consistency = 1

master\_verify\_checksum = ON

# binlog\_group\_commit\_sync\_delay = 10

# binlog\_group\_commit\_sync\_no\_delay\_count = 10

# master settings #

auto\_increment\_increment=2

auto\_increment\_offset=2

# slave settings #

master\_info\_repository = TABLE

relay\_log\_info\_repository = TABLE

slave\_parallel\_type = LOGICAL\_CLOCK

slave\_parallel\_workers = 20

slave\_rows\_search\_algorithms = "INDEX\_SCAN,HASH\_SCAN"

log\_slave\_updates = 1

relay\_log\_recovery = 1

# semi-sync replication settings #

loose\_rpl\_semi\_sync\_master\_enabled = 1

loose\_rpl\_semi\_sync\_slave\_enabled = 1

loose\_rpl\_semi\_sync\_master\_timeout = 3000

[mysqld\_safe]

log\_error=/data/mysqldata/3306/log/mysql\_error.log

pid\_file=/data/mysqldata/3306/mysql.pid

[client]

port=3306

socket=/data/mysqldata/3306/mysql.sock

**在目标机器B1进行恢复**

[root@kettle backup]# innobackupex --defaults-file=/etc/my.cnf --copy-back /root/dba/backup/2020-05-29\_16-56-39/

[root@kettle 3306]# chown mysql:mysql -R /data/mysqldata/3306

[root@kettle 2020-06-04\_10-12-32]# systemctl start mysql

**在目标机器B1，恢复B-->B1的复制链路**

mysql> stop slave;

Query OK, 0 rows affected (0.00 sec)

mysql> reset slave;

Query OK, 0 rows affected (0.01 sec)

mysql> reset master;

Query OK, 0 rows affected (0.01 sec)

gtid\_purged值来自xtrabackup\_slave\_info

mysql> SET GLOBAL gtid\_purged='842db0f4-a092-11ea-b10c-0050569e2c8c:1-204, d17f4f41-a08f-11ea-bfab-00505692b7fa:13-6631';

Query OK, 0 rows affected (0.01 sec)

mysql> CHANGE MASTER TO MASTER\_AUTO\_POSITION=1;

Query OK, 0 rows affected (0.07 sec)

mysql> start slave;

Query OK, 0 rows affected (0.04 sec)

**检查复制链路是否正常，注意Slave\_IO\_Running, Slave\_SQL\_Running（B1上执行）**

mysql> show slave status \G

**设置B1只读（B1上执行）**

mysql> set global super\_read\_only = 1;

Query OK, 0 rows affected (0.00 sec)

## 搭建B-->B2复制链路②

同[搭建B-->B1复制链路](#_搭建B-->B1复制链路)，注意修改my.cnf文件中server\_id, auto\_increment\_off.

auto\_increment\_increment=2

auto\_increment\_offset=1

server\_id = 888

注意：为防止意外，B2需打开只读模式。

## 搭建B1-->B2复制链路③

经过上述步骤后，B2的当前Master是B，现将B2的Master修改B1。以下操作在B2执行。

mysql> stop slave;

Query OK, 0 rows affected (0.01 sec)

mysql> reset slave all;

Query OK, 0 rows affected (0.02 sec)

mysql> change master to master\_host='88.88.16.125', master\_user='repluser', master\_password='repluser', master\_port=3306, master\_auto\_position=1;

Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> start slave;

Query OK, 0 rows affected (0.04 sec)

此时，B2的Master不再是B，而是B1.

## 搭建B2-->B1复制链路④

经过上述复制拓扑的构建，当前的复制拓扑，如图：

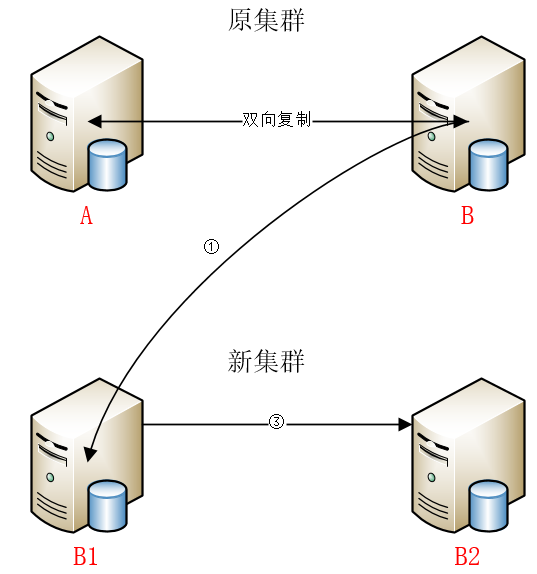


图 4

目前，我们需构建B2-->B1的复制链路，需断开①这条复制链路，将B1的Master指向B2.

在我们开展上述工作前，需保证原集群无业务，且B全部日志已经复制到B1，保证“原集群”和“新集群”数据一致。

检查原集群中对外提供服务的节点A的应用连接情况：show full processlist;

mysql> show full processlist;

+-----+-------------+--------------------+---------+------------------+-------+---------------------------------------------------------------+-----------------------+

| Id | User | Host | db | Command | Time | State | Info |

+-----+-------------+--------------------+---------+------------------+-------+---------------------------------------------------------------+-----------------------+

| 47 | root | 88.88.10.77:9406 | NULL | Sleep | 272 | | NULL |

| 48 | root | 88.88.10.77:9422 | NULL | Sleep | 272 | | NULL |

| 78 | system user | | NULL | Connect | 25367 | Waiting for master to send event | NULL |

| 79 | system user | | NULL | Connect | 564 | Slave has read all relay log; waiting for more updates | NULL |

| 80 | system user | | NULL | Connect | 7922 | Waiting for an event from Coordinator | NULL |

| 81 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 82 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 83 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 84 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 85 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 86 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 87 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 88 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 89 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 90 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 91 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 92 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 93 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 94 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 95 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 96 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 97 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 98 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 99 | system user | | NULL | Connect | 25367 | Waiting for an event from Coordinator | NULL |

| 100 | repluser | 88.88.16.112:38430 | NULL | Binlog Dump GTID | 25364 | Master has sent all binlog to slave; waiting for more updates | NULL |

| 108 | cup\_dba | localhost | testrep | Query | 0 | starting | show full processlist |

+-----+-------------+--------------------+---------+------------------+-------+---------------------------------------------------------------+-----------------------+

26 rows in set (0.00 sec)

如果存在应用连接，需先停止应用。

为防止在构建复制链路时，出现其他应用中途连接“原集群”修改数据，对A库加上全局只读锁。

mysql> flush tables with read lock;

Query OK, 0 rows affected (0.00 sec)

检查B1的复制情况，确保“原集群”日志全部在B1全部应用。这里，需注意B1的MASTER当前是B。所以，我们需要确保A-->B-->B1复制链路都没有延迟，并且日志应用完毕。

**A节点（对外提供）Master信息（A节点执行）**

mysql> show master status \G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: actdata.000002

Position: 8211003

Binlog\_Do\_DB:

Binlog\_Ignore\_DB:

Executed\_Gtid\_Set: 842db0f4-a092-11ea-b10c-0050569e2c8c:1-23605,

d17f4f41-a08f-11ea-bfab-00505692b7fa:1-3:13-6631

1 row in set (0.00 sec)

**B节点的复制信息（B节点执行）**

mysql> show slave status \G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Slave\_IO\_State: Waiting for master to send event

Master\_Host: 88.88.16.123

Master\_User: repluser

Master\_Port: 60001

Connect\_Retry: 60

Master\_Log\_File: actdata.000002

Read\_Master\_Log\_Pos: 8211003

Relay\_Log\_File: actdata.000002

Relay\_Log\_Pos: 6467077

Relay\_Master\_Log\_File: actdata.000002

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

Replicate\_Do\_DB:

Replicate\_Ignore\_DB:

Replicate\_Do\_Table:

Replicate\_Ignore\_Table:

Replicate\_Wild\_Do\_Table:

Replicate\_Wild\_Ignore\_Table:

Last\_Errno: 0

Last\_Error:

Skip\_Counter: 0

Exec\_Master\_Log\_Pos: 8211003

Relay\_Log\_Space: 6467276

Until\_Condition: None

Until\_Log\_File:

Until\_Log\_Pos: 0

Master\_SSL\_Allowed: No

Master\_SSL\_CA\_File:

Master\_SSL\_CA\_Path:

Master\_SSL\_Cert:

Master\_SSL\_Cipher:

Master\_SSL\_Key:

Seconds\_Behind\_Master: 0

Master\_SSL\_Verify\_Server\_Cert: No

Last\_IO\_Errno: 0

Last\_IO\_Error:

Last\_SQL\_Errno: 0

Last\_SQL\_Error:

Replicate\_Ignore\_Server\_Ids:

Master\_Server\_Id: 391

Master\_UUID: 842db0f4-a092-11ea-b10c-0050569e2c8c

Master\_Info\_File: mysql.slave\_master\_info

SQL\_Delay: 0

SQL\_Remaining\_Delay: NULL

Slave\_SQL\_Running\_State: Slave has read all relay log; waiting for more updates

Master\_Retry\_Count: 86400

Master\_Bind:

Last\_IO\_Error\_Timestamp:

Last\_SQL\_Error\_Timestamp:

Master\_SSL\_Crl:

Master\_SSL\_Crlpath:

Retrieved\_Gtid\_Set: 842db0f4-a092-11ea-b10c-0050569e2c8c:50-23605

Executed\_Gtid\_Set: 842db0f4-a092-11ea-b10c-0050569e2c8c:1-23605,

d17f4f41-a08f-11ea-bfab-00505692b7fa:1-3:13-6631

Auto\_Position: 1

Replicate\_Rewrite\_DB:

Channel\_Name:

Master\_TLS\_Version:

1 row in set (0.00 sec)

从上验证得知，A-->B复制链路正常，并且日志应用完毕。同理，查看下B-->B1的复制情况。

**B节点Master信息（B节点执行）**

mysql> show master status \G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: actdata.000001

Position: 6184845

Binlog\_Do\_DB:

Binlog\_Ignore\_DB:

Executed\_Gtid\_Set: 842db0f4-a092-11ea-b10c-0050569e2c8c:1-23605,

d17f4f41-a08f-11ea-bfab-00505692b7fa:1-3:13-6631

1 row in set (0.00 sec)

**B1节点的复制信息（B1节点执行）**

mysql> show slave status \G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Slave\_IO\_State: Waiting for master to send event

Master\_Host: 88.88.16.112

Master\_User: repluser

Master\_Port: 60001

Connect\_Retry: 60

Master\_Log\_File: actdata.000001

Read\_Master\_Log\_Pos: 6184845

Relay\_Log\_File: mysql-relay-bin.000002

Relay\_Log\_Pos: 6144800

Relay\_Master\_Log\_File: actdata.000001

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

Replicate\_Do\_DB:

Replicate\_Ignore\_DB:

Replicate\_Do\_Table:

Replicate\_Ignore\_Table:

Replicate\_Wild\_Do\_Table:

Replicate\_Wild\_Ignore\_Table:

Last\_Errno: 0

Last\_Error:

Skip\_Counter: 0

Exec\_Master\_Log\_Pos: 6184845

Relay\_Log\_Space: 6145007

Until\_Condition: None

Until\_Log\_File:

Until\_Log\_Pos: 0

Master\_SSL\_Allowed: No

Master\_SSL\_CA\_File:

Master\_SSL\_CA\_Path:

Master\_SSL\_Cert:

Master\_SSL\_Cipher:

Master\_SSL\_Key:

Seconds\_Behind\_Master: 0

Master\_SSL\_Verify\_Server\_Cert: No

Last\_IO\_Errno: 0

Last\_IO\_Error:

Last\_SQL\_Errno: 0

Last\_SQL\_Error:

Replicate\_Ignore\_Server\_Ids:

Master\_Server\_Id: 390

Master\_UUID: d17f4f41-a08f-11ea-bfab-00505692b7fa

Master\_Info\_File: mysql.slave\_master\_info

SQL\_Delay: 0

SQL\_Remaining\_Delay: NULL

Slave\_SQL\_Running\_State: Slave has read all relay log; waiting for more updates

Master\_Retry\_Count: 86400

Master\_Bind:

Last\_IO\_Error\_Timestamp:

Last\_SQL\_Error\_Timestamp:

Master\_SSL\_Crl:

Master\_SSL\_Crlpath:

Retrieved\_Gtid\_Set: 842db0f4-a092-11ea-b10c-0050569e2c8c:205-23605,

d17f4f41-a08f-11ea-bfab-00505692b7fa:1-3

Executed\_Gtid\_Set: 842db0f4-a092-11ea-b10c-0050569e2c8c:1-23605,

b9eb163f-a60a-11ea-b2f3-0050569276dc:1,

d17f4f41-a08f-11ea-bfab-00505692b7fa:1-3:13-6631

Auto\_Position: 1

Replicate\_Rewrite\_DB:

Channel\_Name:

Master\_TLS\_Version:

1 row in set (0.00 sec)

从上验证得知，B-->B1复制链路正常，并且日志应用完毕。

当我们检查完上述复制情况后，我们可确定“原集群”和“新集群”数据一致。现在，构建B2-->B1的复制链路，使得B1，B2实现互为主从的双主模式。我们需将B1的Master由B更改成B2，以下操作均在B1执行。

**B1节点停止原有复制关系**

mysql> stop slave;

Query OK, 0 rows affected (0.00 sec)

mysql> reset slave all;

Query OK, 0 rows affected (0.02 sec)

**将B1的Master指向B2**

mysql> change master to master\_host='88.88.16.126', master\_user='repluser', master\_password='repluser',master\_port=3306,master\_auto\_position=1;

Query OK, 0 rows affected, 2 warnings (0.01 sec)

**启动B2-->B1复制链路**

mysql> start slave;

Query OK, 0 rows affected (0.04 sec)

**查看复制链路状态**

mysql> show slave status \G

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Slave\_IO\_State: Waiting for master to send event

Master\_Host: 88.88.16.126

Master\_User: repluser

Master\_Port: 3306

Connect\_Retry: 60

Master\_Log\_File: mysql-bin.000001

Read\_Master\_Log\_Pos: 6144712

Relay\_Log\_File: mysql-relay-bin.000002

Relay\_Log\_Pos: 639

Relay\_Master\_Log\_File: mysql-bin.000001

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

Replicate\_Do\_DB:

Replicate\_Ignore\_DB:

Replicate\_Do\_Table:

Replicate\_Ignore\_Table:

Replicate\_Wild\_Do\_Table:

Replicate\_Wild\_Ignore\_Table:

Last\_Errno: 0

Last\_Error:

Skip\_Counter: 0

Exec\_Master\_Log\_Pos: 6144712

Relay\_Log\_Space: 846

Until\_Condition: None

Until\_Log\_File:

Until\_Log\_Pos: 0

Master\_SSL\_Allowed: No

Master\_SSL\_CA\_File:

Master\_SSL\_CA\_Path:

Master\_SSL\_Cert:

Master\_SSL\_Cipher:

Master\_SSL\_Key:

Seconds\_Behind\_Master: 0

Master\_SSL\_Verify\_Server\_Cert: No

Last\_IO\_Errno: 0

Last\_IO\_Error:

Last\_SQL\_Errno: 0

Last\_SQL\_Error:

Replicate\_Ignore\_Server\_Ids:

Master\_Server\_Id: 888

Master\_UUID: a08a0f78-a631-11ea-b98e-0050569276d0

Master\_Info\_File: mysql.slave\_master\_info

SQL\_Delay: 0

SQL\_Remaining\_Delay: NULL

Slave\_SQL\_Running\_State: Slave has read all relay log; waiting for more updates

Master\_Retry\_Count: 86400

Master\_Bind:

Last\_IO\_Error\_Timestamp:

Last\_SQL\_Error\_Timestamp:

Master\_SSL\_Crl:

Master\_SSL\_Crlpath:

Retrieved\_Gtid\_Set: a08a0f78-a631-11ea-b98e-0050569276d0:1

Executed\_Gtid\_Set: 842db0f4-a092-11ea-b10c-0050569e2c8c:1-23605,

a08a0f78-a631-11ea-b98e-0050569276d0:1,

b9eb163f-a60a-11ea-b2f3-0050569276dc:1,

d17f4f41-a08f-11ea-bfab-00505692b7fa:1-3:13-6631

Auto\_Position: 1

Replicate\_Rewrite\_DB:

Channel\_Name:

Master\_TLS\_Version:

1 row in set (0.00 sec)

## 新集群复制拓扑验证

现在，原集群和新集群之间无任何复制链路关联。新集群由B1，B2组成，以互为主从架构，主备模式对外提供服务。这里，我们对B1<-->B2复制链路进行验证。

**在B1，B2取消只读限制**

mysql> set global super\_read\_only = 0;

Query OK, 0 rows affected (0.00 sec)

**在B1执行**

mysql> use testrep;

Database changed

mysql> insert into testrep values(1, 'test for sync');

Query OK, 1 row affected (0.01 sec)

**在B2查看**

mysql> select \* from testrep.testrep;

+------+---------------+

| id | teststr |

+------+---------------+

| 1 | test for sync |

+------+---------------+

1 row in set (0.00 sec)

**在B2执行**

mysql> insert into testrep.testrep values(2, 'test for sync1');

Query OK, 1 row affected (0.01 sec)

**在B1查看**

mysql> select \* from testrep.testrep;

+------+----------------+

| id | teststr |

+------+----------------+

| 1 | test for sync |

| 2 | test for sync1 |

+------+----------------+

2 rows in set (0.00 sec)

经过上述验证，B1的更改同步应用到B2，B2的更改同步应用到B1，复制链路正常。

验证通过后，可将应用指向新集群。

## 其他事项

原集群A节点，前期发起全局只读锁，后续需继续使用，可使用unlock tables进行解锁。

mysql> unlock tables;

Query OK, 0 rows affected (0.00 sec)

# 迁移验证

使用percona-toolkit工具，对核心业务表进行抽检。

[root@other126 dba]# pt-table-checksum --nocheck-replication-filters --no-check-binlog-format --replicate=testrep.checksums --databases=testrep --tables=test h=127.0.0.1,u=root,p=root,P=3306

# A software update is available:

TS ERRORS DIFFS ROWS DIFF\_ROWS CHUNKS SKIPPED TIME TABLE

06-04T18:11:43 0 0 30204 0 4 0 0.896 testrep.test

# 风险评估

在进行原集群和新集群复制链路切断时，即[搭建B2-->B1复制链路④](#_搭建B2-->B1复制链路④)，原集群本身处于只读状态，如果有应用中途发起业务操作对原集群进行更改操作，应用会超时。如果应用没有重试机制，这部分数据将丢失。