MySQL binlog浅析

本篇针对binlog、DML、事务以及恢复大表的数据误操作等解析。

my2sql

具有解析大事务或者长事务，生成 DML 统计信息的功能。

部署：

```
go1.20.linux-amd64.tar.gz -- https://studygolang.com/dl 下载相关版本(#go [-v需go1.40以上])
tar xf go1.20.linux-amd64.tar.gz
-- go env -w GOPROXY=https://goproxy.cn -- 更换proxy代理
go build -- 构建
ls -n -- 软连或环境变量配置
#my2sql
git clone https://github.com/liuhr/my2sql.git
cd my2sql/
go build .
```

my2sql参数：

image-my2sql.png

```
#查看mysql的相关参数设置
mysql> select @@server_id,@@binlog_format,@@binlog_row_image,@@max_binlog_size,@@log_bin_basena
+-------------+-----------------+--------------------+-------------------+--------------------
| @@server_id | @@binlog_format | @@binlog_row_image | @@max_binlog_size | @@log_bin_basename
+-------------+-----------------+--------------------+-------------------+--------------------
|      593308 | ROW             | FULL               |        1073741824 | /data/mysql_8034/bir
+-------------+-----------------+--------------------+-------------------+--------------------
1 row in set, 1 warning (0.08 sec)
mysql> \! ls -l /data/mysql_8034/binlog
总用量 6535916
-rw-r-----. 1 mysql mysql 1073832385 8月  31 09:33 mysql-bin.000031
-rw-r-----. 1 mysql mysql 1073787094 8月  31 09:41 mysql-bin.000032
-rw-r-----. 1 mysql mysql 1074143566 8月  31 09:48 mysql-bin.000033
-rw-r-----. 1 mysql mysql  848228870 8月  31 10:01 mysql-bin.000034
-rw-r-----. 1 mysql mysql  340981782 8月  31 10:04 mysql-bin.000035
-rw-r-----. 1 mysql mysql  219145438 8月  31 10:07 mysql-bin.000036
-rw-r-----. 1 mysql mysql 1073745167 8月  31 10:17 mysql-bin.000037
-rw-r-----. 1 mysql mysql  988884446 9月   6 10:38 mysql-bin.000038
-rw-r-----. 1 mysql mysql        197 9月   6 10:41 mysql-bin.000039
-rw-r-----. 1 mysql mysql        369 9月   6 10:41 mysql-bin.index
```

测试脚本：

```
[root@localhost my2sql]# cat sql.sh
#! /bin/bash
date;
/opt/my2sql/my2sql -user lixl -password 'PostgreSQL@14nb' -host 192.168.97.51   -port 3306 -moc
```

执行结果：

```
[root@localhost my2sql]# tree outfile/
outfile/
├── 37
│   ├── biglong_trx.txt
│   ├── binlog_status.txt
│   ├── forward.37.sql ## 标准sql
│   └── output
└── 38
    ├── biglong_trx.txt
    ├── binlog_status.txt
    ├── forward.38.sql ## 标准sql
    └── output

2 directories, 8 files
```

output文件：

```
[2023/09/06 11:09:59] [info] file.go:32 start to parse binlog from local files
[2023/09/06 11:09:59] [info] file.go:35 start to parse /data/mysql_8034/binlog/mysql-bin.00003
[2023/09/06 11:09:59] [info] file.go:44 start to parse /data/mysql_8034/binlog/mysql-bin.00003
[2023/09/06 11:09:59] [info] events.go:61 start thread 1 to generate redo/rollback sql
[2023/09/06 11:09:59] [info] stats_process.go:166 start thread to analyze statistics from binl
[2023/09/06 11:09:59] [info] events.go:221 start thread to write redo/rollback sql into file
[2023/09/06 11:09:59] [info] events.go:61 start thread 2 to generate redo/rollback sql
[2023/09/06 11:09:59] [info] events.go:61 start thread 3 to generate redo/rollback sql
[2023/09/06 11:09:59] [info] events.go:61 start thread 4 to generate redo/rollback sql
[2023/09/06 11:10:00] [info] events.go:255 finish processing mysql-bin.000038 10486451
..
```

biglong_trx.txt文件：

```
binlog            starttime           stoptime            startpos    stoppos     rows     durat
mysql-bin.000038  2023-08-31_10:17:24 2023-08-31_10:17:25 181844      184602      5        1
mysql-bin.000038  2023-08-31_10:17:24 2023-08-31_10:17:25 184681      187439      5        1
mysql-bin.000038  2023-08-31_10:17:24 2023-08-31_10:17:25 190355      193113      5        1
mysql-bin.000038  2023-08-31_10:17:24 2023-08-31_10:17:25 193192      195950      5        1
..
```

binlog_status.txt文件：

```
binlog            starttime           stoptime            startpos   stoppos    inserts  update
mysql-bin.000038  2023-08-31_10:17:24 2023-08-31_10:17:53 4460       57340048   6639     20167
mysql-bin.000038  2023-08-31_10:17:24 2023-08-31_10:17:53 7297       57337211   6829     20402
mysql-bin.000038  2023-08-31_10:17:24 2023-08-31_10:17:54 421        57340708   6743     20067
mysql-bin.000038  2023-08-31_10:17:54 2023-08-31_10:18:23 57340767   117368791  7181     20840
mysql-bin.000038  2023-08-31_10:17:53 2023-08-31_10:18:23 57341865   117369693  7028     21109
mysql-bin.000038  2023-08-31_10:17:54 2023-08-31_10:18:23 57345904   117358345  6951     21528
mysql-bin.000038  2023-08-31_10:18:23 2023-08-31_10:18:53 117369948  176094031  6947     20843
mysql-bin.000038  2023-08-31_10:18:23 2023-08-31_10:18:53 117371150  176089919  6911     20376
mysql-bin.000038  2023-08-31_10:18:23 2023-08-31_10:18:54 117372785  176095593  6842     20881
mysql-bin.000038  2023-08-31_10:18:53 2023-08-31_10:19:23 176095848  236092469  7149     21309
mysql-bin.000038  2023-08-31_10:18:54 2023-08-31_10:19:23 176097050  236095306  7028     21339
mysql-bin.000038  2023-08-31_10:18:54 2023-08-31_10:19:23 176102724  236094404  6972     20799
..
```

forward.38.sql：

```
UPDATE `e`.`sbtest3` SET `k`=499896 WHERE `id`=500061;
UPDATE `e`.`sbtest3` SET `k`=491597 WHERE `id`=503705;
UPDATE `e`.`sbtest3` SET `c`='13148148113-80463883887-04983574644-65011861070-14645772762-81402
DELETE FROM `e`.`sbtest3` WHERE `id`=501078;
INSERT INTO `e`.`sbtest3` (`id`,`k`,`c`,`pad`) VALUES (501078,502027,'86190025454-12321210530-7
UPDATE `e`.`sbtest3` SET `k`=503248 WHERE `id`=499409;
UPDATE `e`.`sbtest3` SET `k`=362476 WHERE `id`=502126;
..
```

总结my2sql：

my2sql 限制，如：

1. my2sql 是伪装成从库去在线获取主库 binlog，然后进行解析，因此执行操作的数据库用户需要具有 SELECT，REPLICATION SALVE，REPLICATION CLIENT 的权限。
2. 使用回滚/闪回功能时，binlog 格式必须为 row ，且 binlog_row_image=full ， DML 统计以及大事务分析不受影响
3. 只能回滚 DML ， 不能回滚 DDL
4. my2sql 据某论坛 bigint() unsigned 在转储会有bug 导致标准sql 主键会生成-1值、需注意或github了解是否修复；
5. my2sql 并发功能效果不佳(仅限于本次测试)但比binlog2sql略好、默认两个线程。

binlog2sql

部署：

```
git clone https://github.com/danfengcao/binlog2sql.git

[root@db01 binlog2sql]# pwd
/root/binlog2sql
[root@db01 binlog2sql]# ls
binlog2sql  example  LICENSE  README.md  requirements.txt  tests

yum install python3 #安装python3环境：

[root@db01 binlog2sql]# cat requirements.txt #修改requirements.txt
PyMySQL==0.7.11
wheel==0.29.0
mysql-replication==0.13

把PyMySQL==0.7.11修改为：PyMySQL==0.9.3 #安装依赖：
pip3 install -r requirements.txt
pip3 show pymysql

可选：
连接mysql8.0后，升级pymysql至最新版本，上一步修改了就不用执行了
升级最新版本：
-- pip3 install --upgrade PyMySQL
```

binlog2sql参数：

image-binlog2sql.png

- -d, --databases 只输出目标db的sql。可选。默认为空。
- -t, --tables 只输出目标tables的sql。可选。默认为空。

解析测试(delete)：

```
[root@postgre binlog2sql]# python3 binlog2sql.py -ulixl -plixl -d nglicps2 -t user_t --start-fi
DELETE FROM `nglicps2`.`user_t` WHERE `id`=8 AND `user_name`='aaa' AND `password` IS NULL AND `
DELETE FROM `nglicps2`.`user_t` WHERE `id`=9 AND `user_name`='bbb' AND `password` IS NULL AND `
DELETE FROM `nglicps2`.`user_t` WHERE `id`=10 AND `user_name`='aaa' AND `password` IS NULL AND
DELETE FROM `nglicps2`.`user_t` WHERE `id`=11 AND `user_name`='bbb' AND `password` IS NULL AND
DELETE FROM `nglicps2`.`user_t` WHERE `id`=12 AND `user_name`='aaa' AND `password` IS NULL AND
DELETE FROM `nglicps2`.`user_t` WHERE `id`=13 AND `user_name`='bbb' AND `password` IS NULL AND
```

回滚测试(insert)：

1. 直接执行生成的语句
2. 导出到文件，进入mysql中进行恢复
3. 可增加（--sql-type=delete --start-position=2698 --stop-position=3514 -B > /root/city_delete.sql）

```
[root@postgre binlog2sql]# python3 binlog2sql.py -ulixl -plixl -d nglicps2 -t user_t --start-fi
INSERT INTO `nglicps2`.`user_t`(`id`, `user_name`, `password`, `age`) VALUES (13, 'bbb', NULL,
INSERT INTO `nglicps2`.`user_t`(`id`, `user_name`, `password`, `age`) VALUES (12, 'aaa', NULL,
INSERT INTO `nglicps2`.`user_t`(`id`, `user_name`, `password`, `age`) VALUES (11, 'bbb', NULL,
INSERT INTO `nglicps2`.`user_t`(`id`, `user_name`, `password`, `age`) VALUES (10, 'aaa', NULL,
INSERT INTO `nglicps2`.`user_t`(`id`, `user_name`, `password`, `age`) VALUES (9, 'bbb', NULL, 3
INSERT INTO `nglicps2`.`user_t`(`id`, `user_name`, `password`, `age`) VALUES (8, 'aaa', NULL, 2
```

远程使用：

例如：我的mysql服务器是192.168.0.51，我使用远程主机连接；远程访问，加上-h -P参数

```
[root@db02 binlog2sql]# python3 binlog2sql.py -h 192.168.0.51 -P3306 -uroot -p123456 -d world -
[root@db02 binlog2sql]# python3 binlog2sql.py -h 192.168.0.51 -P3306 -uroot -p123456 -d world -
```

binlog2sql总结：

binlog2sql 限制，如

1. MySQL Server 须设置 server_id， log_bin， max_binlog_size=1G， binlog_format=row， binlog_row_image=full 这些参数，
2. 与 my2sql 一样，也是伪装成从库拉取 binlog， 需要连接数据库的用户有 SELECT， REPLICATION SLAVE， REPLICATION CLIENT 权限

my2sql对比binlog2sql：

image-my2_vs_binlog2.png

binlog预处理：

筛选具体SQL的binlog信息：

```
[root@postgre binlog]# /data/mysql_basedir_3306/bin/mysqlbinlog /data/mysql_3306/binlog/mysql-b
#230602 15:42:54 server id 2130706431  end_log_pos 251684716 CRC32 0x1e62252a   Rows_query
# DELETE FROM nglicps2.cps_transactions_his
# WHERE id = 37
--
#230602 15:43:16 server id 2130706431  end_log_pos 251685652 CRC32 0x2e435600   Rows_query
# DELETE FROM nglicps2.cps_transactions_his
# WHERE id = 38
..
```

- skip-gtids=true：忽略 GTID 显示。

组提交binlog详情依据last_committed分组:

```
[root@postgre opt]# /data/mysql_basedir_3306/bin/mysqlbinlog /data/mysql_3306/binlog/mysql-bin.
# at 194
#230629  9:14:18 server id 2130706431  end_log_pos 259 CRC32 0x2bde37c8         GTID    last_co
--
# at 392
#230629  9:14:18 server id 2130706431  end_log_pos 457 CRC32 0x56a98d76         GTID    last_co
--
# at 737
#230629  9:15:53 server id 2130706431  end_log_pos 802 CRC32 0x410d98bb         GTID    last_co
--
# at 20897
..
```

过滤信息将两行数据相减得出每个事物大小(`at 392` - `at 194` = 事务大小):

```
[root@postgre opt]# /data/mysql_basedir_3306/bin/mysqlbinlog /data/mysql_3306/binlog/mysql-bin.
20160
445
442
439
435
433
432
429
425
..
```

直观查看组提交信息:

```
[root@postgre opt]# /data/mysql_basedir_3306/bin/mysqlbinlog /root/mysql-bin.000013 | grep -a '
last_committed=120416
last_committed=128345
last_committed=139306
last_committed=179801
last_committed=215436
last_committed=230472
last_committed=230477
last_committed=230490
last_committed=230502
last_committed=230527
..
[root@postgre opt]# /data/mysql_basedir_3306/bin/mysqlbinlog /root/mysql-bin.000013 | grep -a '
#220606 17:42:43 server id 1110053  end_log_pos 105265545 CRC32 0x9eae9aa9      GTID    last_co
#220606 17:42:43 server id 1110053  end_log_pos 105266322 CRC32 0x2ff09f2e      GTID    last_co
..
```

查看事务数：

```
[root@postgre opt]# /data/mysql_basedir_3306/bin/mysqlbinlog /root/mysql-bin.000013 | grep -a '
530770
[root@postgre opt]# /data/mysql_basedir_3306/bin/mysqlbinlog /root/mysql-bin.000013 | grep -a '
529221
```

备份+binlog实现时间点恢复数据：

```
[lixl@172-31-1-33 ~]$ /usr/bin/mysqlbinlog --start-datetime="2021-12-30 03:30:00" --stop-dateti
```

image-binlog0000001x.png

analysis_binlog

前两者做为闪回工具进行数据恢复，此工具统计事务信息比较实用、跟my2sql统计DML类似、但是此工具统计信息更加全面且支持并行解析以及生成binlog sql。

部署：

```
git clone https://gitee.com/mo-shan/analysis_binlog.git
cd analysis_binlog
sed -i 's#^mysqlbinlog=.*#mysqlbinlog=\"/usr/local/mysql/bin/mysqlbinlog\"#g' bin/analysis_binl
sed -i 's#^work_dir=.*#work_dir=\"/home/moshan/analysis_binlog\"#g' bin/analysis_binlog
chmod +x bin/analysis_binlog
echo "export PATH=/home/moshan/analysis_binlog/bin:${PATH}" >> ${HOME}/.bashrc
```

binlog文件测试(DML汇总)：

```
[root@localhost analysis_binlog]# ./bin/analysis_binlog -bfile=/data/mysql_8034/binlog/mysql-bi

[2023-08-25 15:21:52] [WARN] [172.17.0.1] Version : v_1.3
[2023-08-25 15:21:52] [INFO] [172.17.0.1] THREAD_1:Analysing --> /data/mysql_8034/binlog/mysql-
[2023-08-25 15:25:15] [INFO] [172.17.0.1] THREAD_1:Analysis completed --> /data/mysql_8034/binl
[root@localhost analysis_binlog]# cat res/mysql-bin.000027.res
Table                                             First Time         Last Time          Type
e.sbtest1                                         230822 14:54:15    230822 14:57:43    DML

Table                                             First Time         Last Time          Type
The total                                         230814 15:53:20    230822 14:57:43    DML

Table                                             First Time         Last Time          Type
Transaction                                       230814 15:53:20    230822 14:57:43    DML
```

参数浅析:

- -bfile: 指定binlog文件, 支持多个文件并行分析, 多个文件用逗号相隔, 需要并行分析时请结合-w参数使用
- -w : 指定并行数, 当需要分析多个binlog文件时该参数有效, 默认是1
- -t : 指定显示结果的格式/内容, 供选选项有"detail|simple". 当指定detail的时候结果较为详细, 会打印详细的分析过程, 消耗时间也不直观, simple只做了统计工作
- -s : 指定排序规则, 供选选项有"insert|update|delete". 默认会把统计结果做一个排序, 按照表的维度统计出insert update delete的次数, 并按照次数大小排序(默认insert)

1. 如果不确定 SQL 格式或是无法筛选到数据，比如因为 delete from 中间冷不丁多一个空格出来，可以使用 grep 多次过滤筛选，比如：grep -C 1 -i "Rows_query" | grep -C 1 -i "Audit_Orga_Specialtype" | grep -C 1 -i "delete" 筛选对应表上的 delete 操作。
2. 触发器执行的 SQL 不会记录在 Rows_query_event 中，只会记录对应的行数据。
3. --database 是无法过滤 rows_query_event 的，只可以过滤行数据。

资料:

- https://github.com/liuhr/my2sql
- https://github.com/danfengcao/binlog2sql
- https://gitee.com/mo-shan/analysis_binlog