

SDCC 2017 | 上海

数据库核心技术与应用实战峰会

CSDN

携程MySQL源码改造介绍

关于我

- 编写代码17年
- 携程MySQL源码研究近3年
- 数据库研究与开发12年多

版本历程

- 5.6.12
 - 2012年引入，第一个MySQL版本
- 5.6.21
 - 2014-2017，绝大部分的源码功能添加与探索集中在该版本上
- 5.7.17
 - 正在上线中的版本

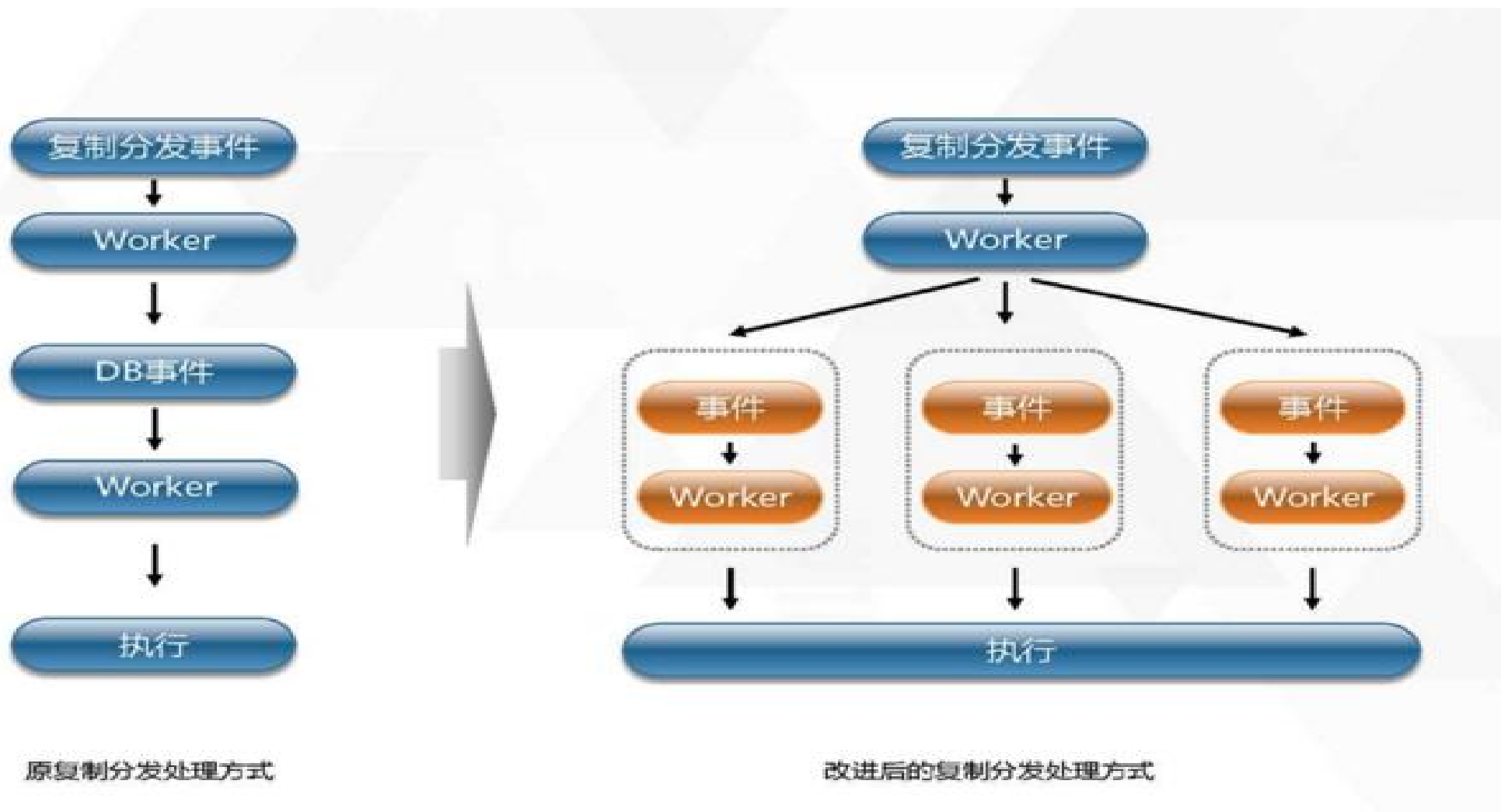
5.6.12的工作

- 异步复制
 - 针对携程监控场景的自定义改造
- flashback功能的移植
 - 移植自阿里数据库专家彭立勋所开发的代码

异步复制

- 背景
 - Zabbix监控下，复制分发延迟情况比较严重
- 特点
 - 部分表的update/insert的操作集中，且SQL相关性不强
- 改造目标
 - 提高slave的写入效率
- 改造手段
 - 在slave针对特定表的DML SQL进行事务拆分，由单线程转为多线程SQL运行

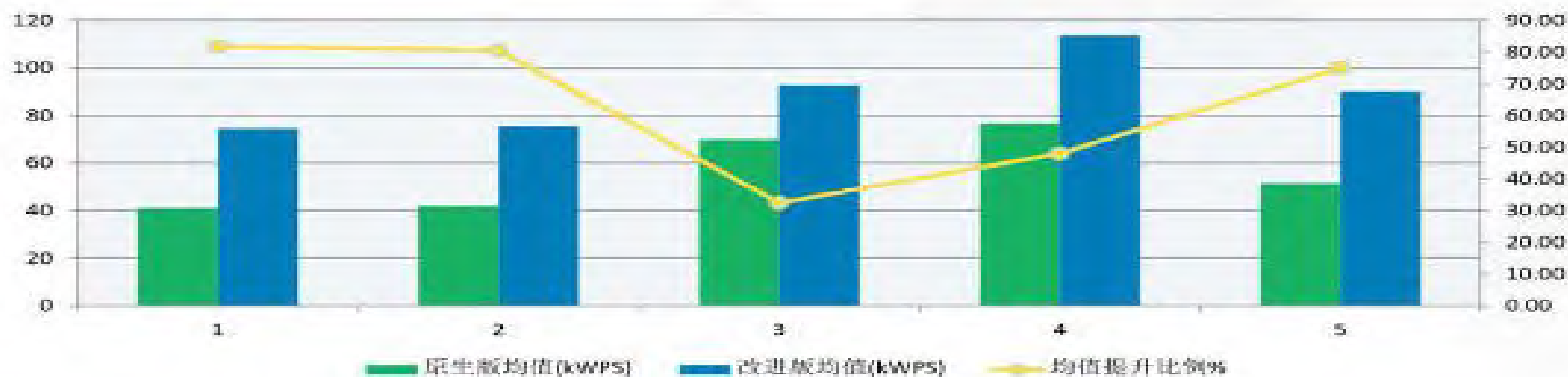
改造原理



异步复制对比测试

结果

测试场景	[1] 16Gbuf+2log	[2] 16Gbuf+4log	[3] 32Gbuf+2log	[4] 32Gbuf+4log	[5] 32Gbuf+4log+query
原生版均值(kWPS)	40.92	41.84	70	76.7	51.47
改进版均值(kWPS)	74.38	75.56	92.71	113.55	90.09
均值提升比例%	81.77%	80.59%	32.44%	48.04%	75.03%



初试啼声

5.6.21

- 移植工作
 - flashback功能
- 新增修改
 - 审计插件
 - Slow log功能增强
 - Show processlist功能增强
- 探索性功能
 - 时间序列存储引擎

审计插件

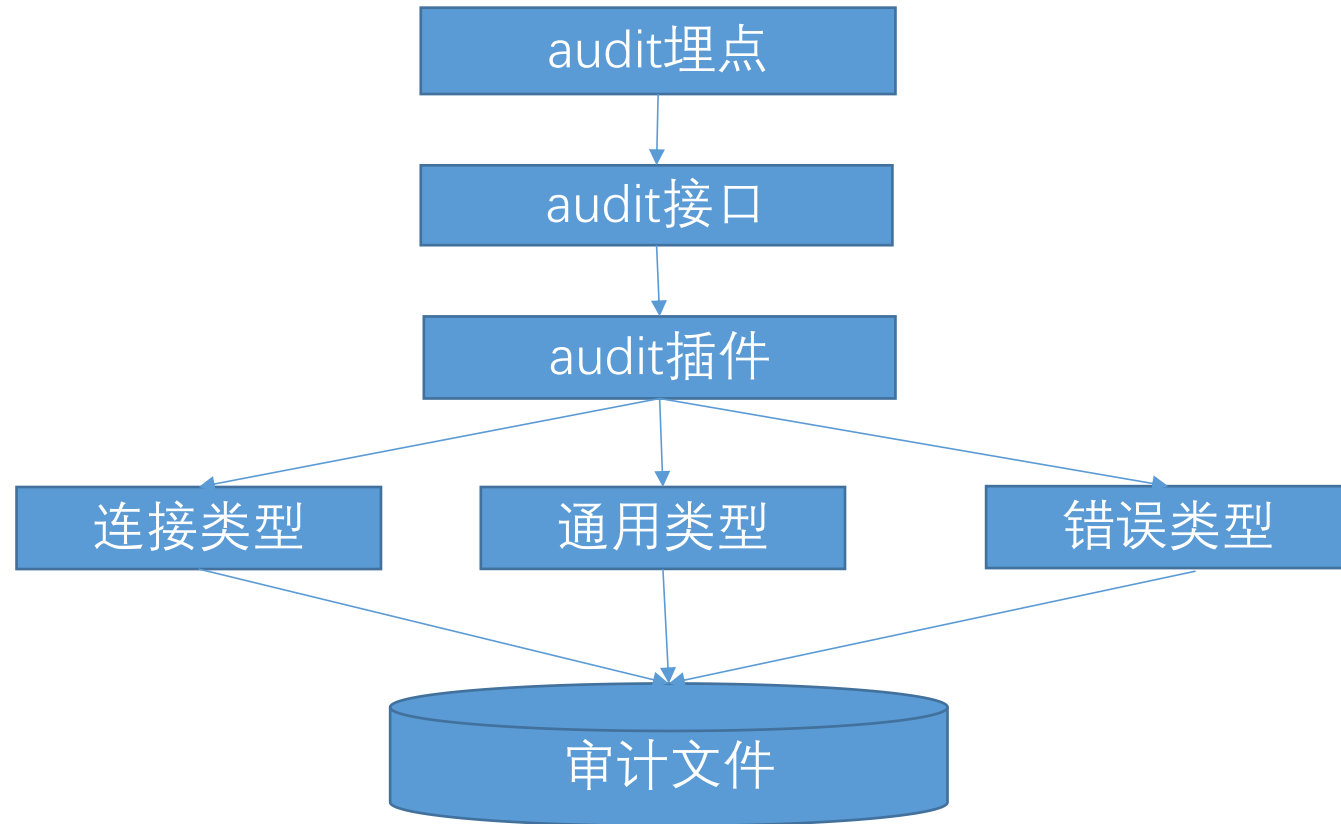
- 目的
 - 线上操作监控
- 功能特性
 - 用户级审计设定
 - 语句级/关键词审计设定
 - 动态配置
 - Json格式输出
 - 缓存提速

```
[general]
audit_file=ctrip_audit.log
audit_error_file=ctrip_audit_error.log
enable_buffer=1

[audit rule]
name=rule1
user=root
#host=192.168
event=connection:connect;connection:disconnect;general:status;general:error
#command=query
#sql_command=set_option
#sql_keyword=names
```

```
{
  "timestamp": "2017-02-06 15:31:54",
  "type": "general",
  "user": "root[root] @ localhost []",
  "host": "localhost",
  "ip": "",
  "command_class": "set_option",
  "sqltext": "set global ctrip_audit_flush_log=1",
  "code": 0
}, {
  "timestamp": "2017-02-06 17:00:28",
  "type": "general",
  "user": "root[root] @ localhost []",
  "host": "localhost",
  "ip": ""
```

原理图



审计代码

- 插件声明

```
mysql_declare_plugin(ctrip_audit)
{
    MYSQL_AUDIT_PLUGIN,           /* type */
    &ctrip_audit_descriptor,      /* descriptor */
    "CTRIP_AUDIT",                /* name, install plugin's plugin_name */
    "Ctrip Corp Jiangyx",        /* author */
}
```

- 接口定义

```
static struct st_mysql_audit ctrip_audit_descriptor=
{
    MYSQL_AUDIT_INTERFACE_VERSION, /* interface version */
    NULL, /* contains the function */
    ctrip_audit_notify, /* notify function */
    { (unsigned long) MYSQL_AUDIT_GENERAL_ALL,
      (unsigned long) MYSQL_AUDIT_CONNECTION_ALL,
      (unsigned long) MYSQL_AUDIT_PARSE_ALL,
    }
```

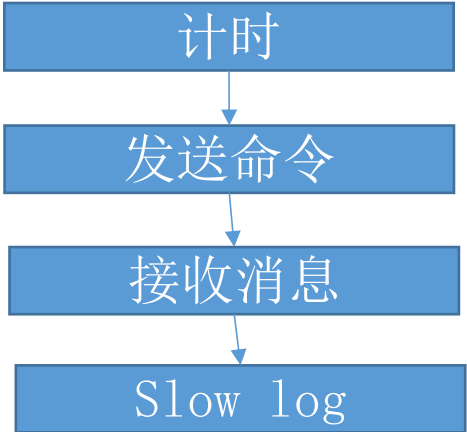
- 处理函数

```
static int ctrip_audit_notify(MYSQL_THD thd,
                             mysql_event_class_t event_class,
                             const void *event)
{
    number_of_calls++;

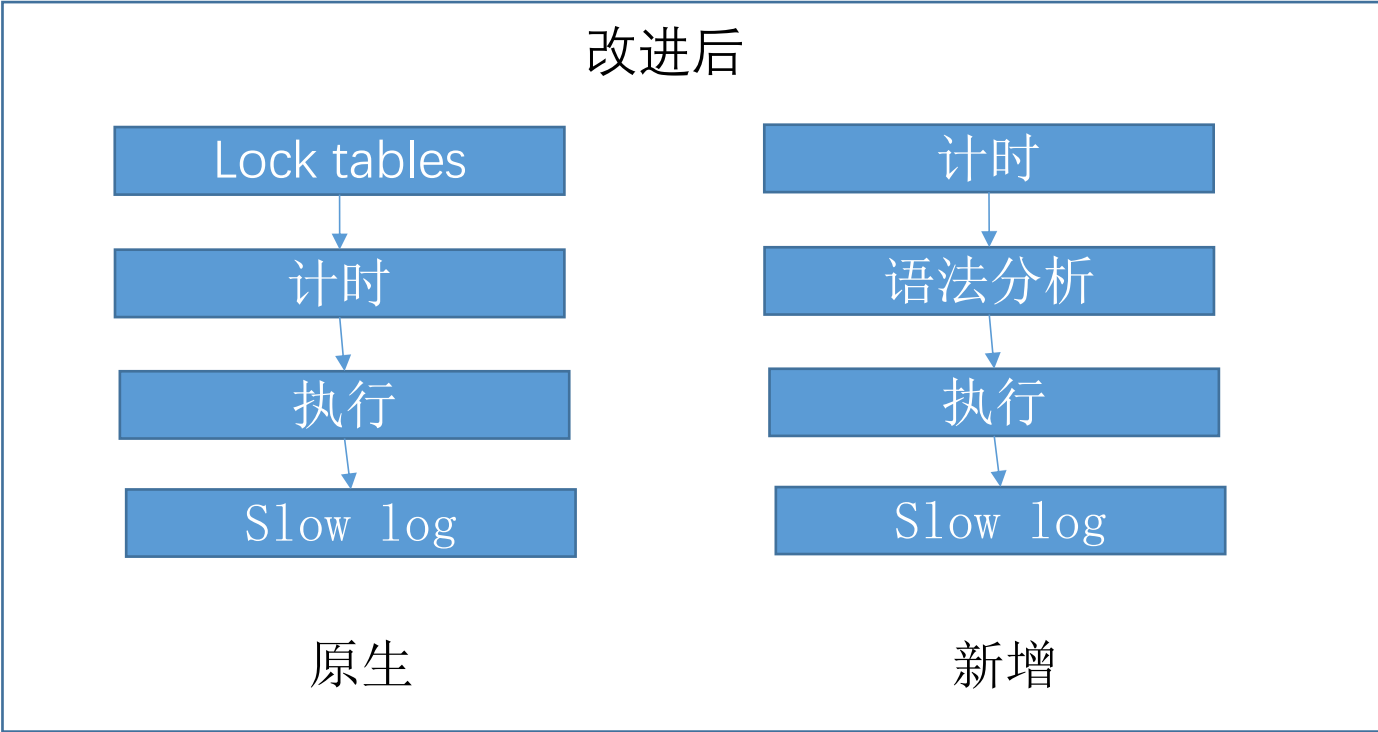
    ctrip_audit_process_event(thd, event_class, event);
}
```

Slow log功能增强

- DBA与用户的矛盾
 - slow log记录执行时间，不包括上锁时间
 - 用户关注命令发出到结果返回的时间
- 改进
 - 增加对上锁时间的记录



客户端



原生

新增

Show processlist功能增强

- 与pstree等工具配合使用，更准确判断各个连接的状态

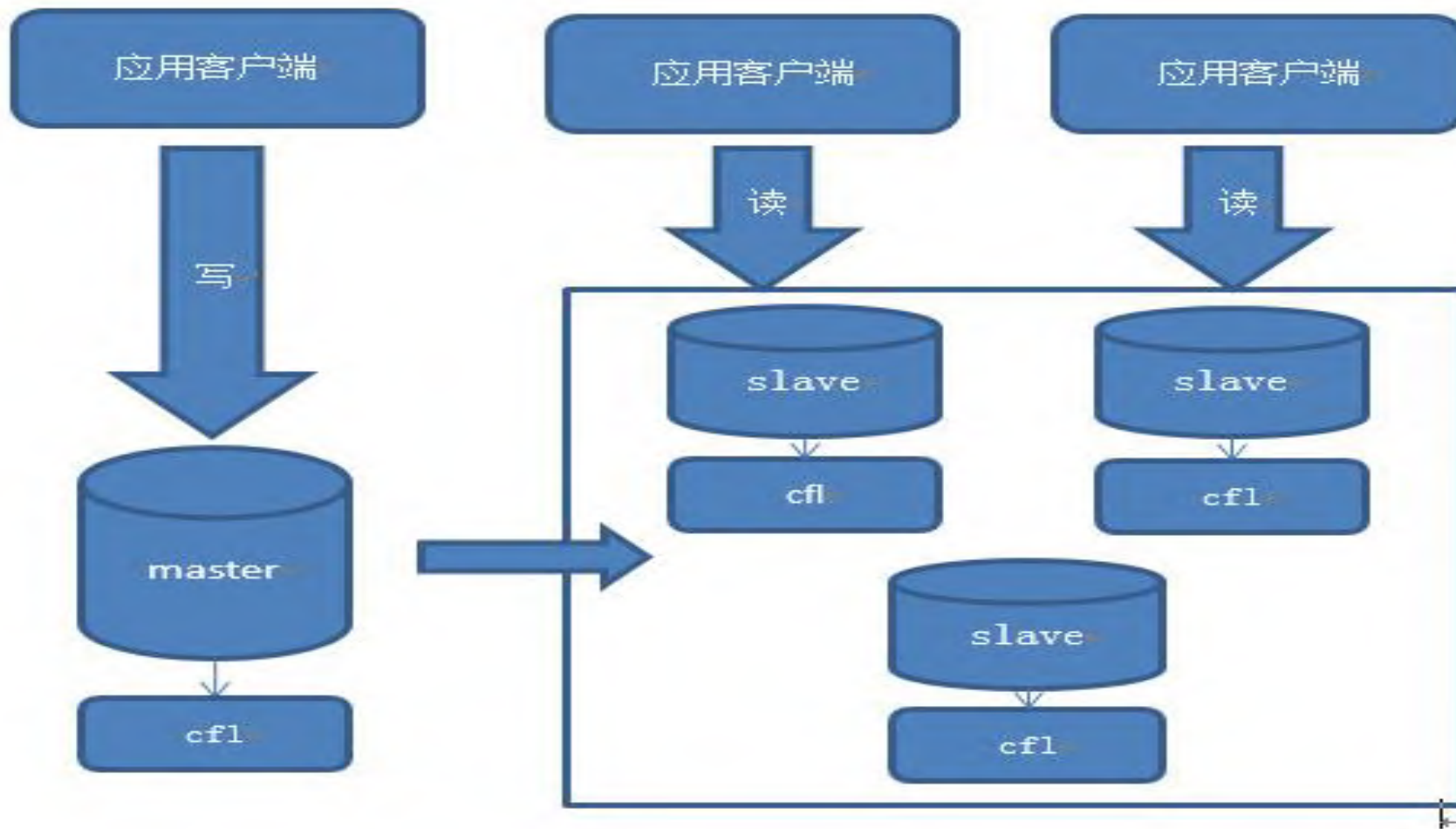
```
Thread 2 (Thread 0x7fe9f8179700 (LWP 24565)):  
#0 0x000000000130e711 in poll () from /lib64/libc.so.6  
#1 0x0000000001d9107a in via_write ()  
#2 0x0000000001d910bd in via_socket_write ()  
#3 0x0000000001d910f9 in via_read ()  
#4 0x0000000001e79101 in net_read_new_loop(st_net*, unsigned loop) ()  
#5 0x0000000001e791de in net_read_packet_header(st_net*) ()  
#6 0x0000000001e791e7 in net_read_packet(st_net*, unsigned loop) ()  
#7 0x0000000001e791f2 in mysql_read ()  
#8 0x0000000001e99019 in Protocol_class::read_packet() ()  
#9 0x0000000001e99014 in Protocol_class::get_command(DDL_STMT*, unsigned  
command) ()  
#10 0x0000000001e64c1c in do_command(THD*) ()  
#11 0x0000000001e615c4 in handle_connection ()  
#12 0x0000000001d74196 in pfs_spawn_thread ()  
#13 0x00000000013407ad in start_thread () from /lib64/libpthread.so.0  
#14 0x000000000130e890 in clone () from /lib64/libc.so.6
```

```
mysql> show processlist\G  
***** 1. row *****  
  Id: 3  
  User: root  
  Host: localhost  
  db: NULL  
Command: Query  
  Time: 0  
  State: starting  
  Info: show processlist  
  Thd_id: 140642866403072  
  Lwpid: 24565  
1 row in set (0.00 sec)
```

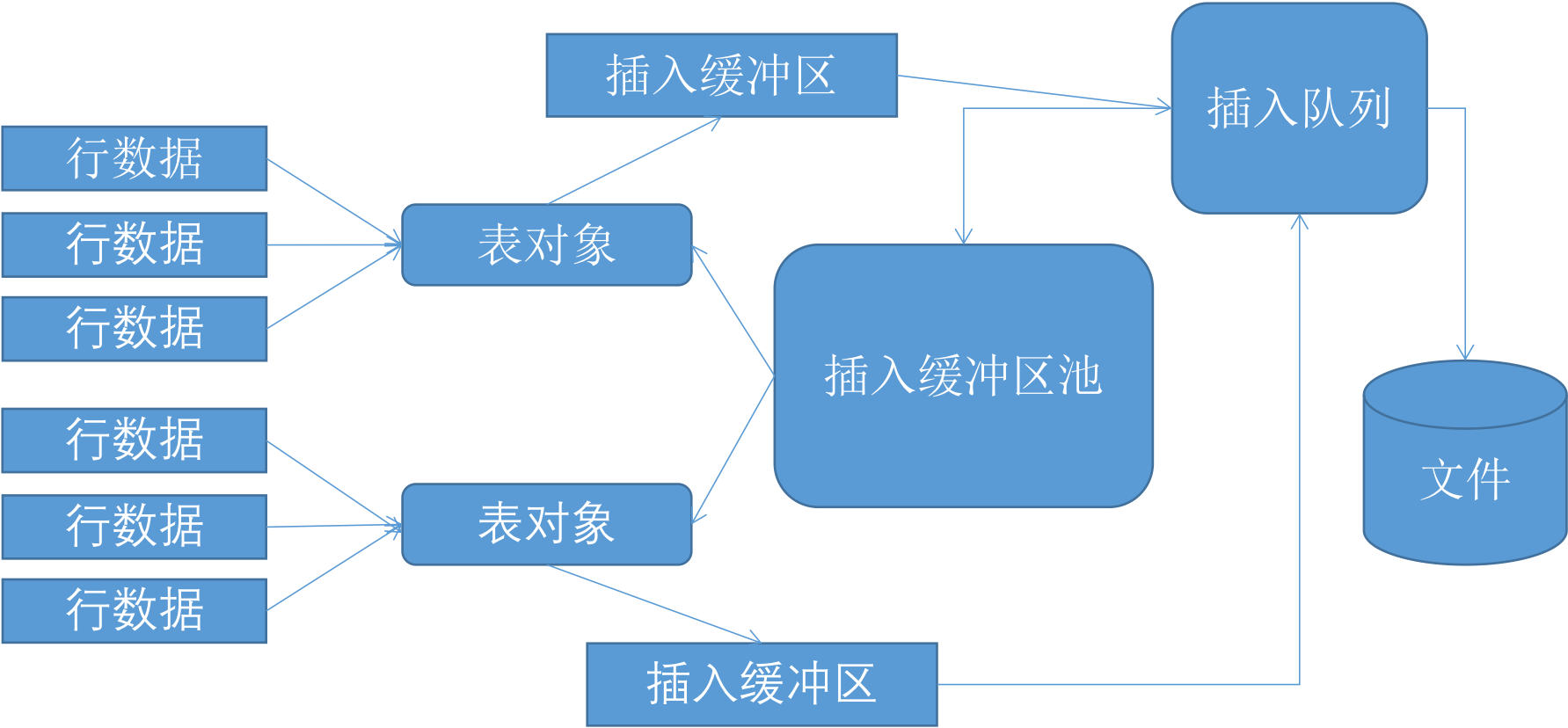
时间序列存储引擎

- 出发点
 - 利用MySQL已有的网络框架
 - 利用MySQL已有的高可用架构
 - SQL语句降低开发者的学习曲线
 - SQL方式标准化时间序列存储引擎的操作方式

时间序列存储引擎-运维视图

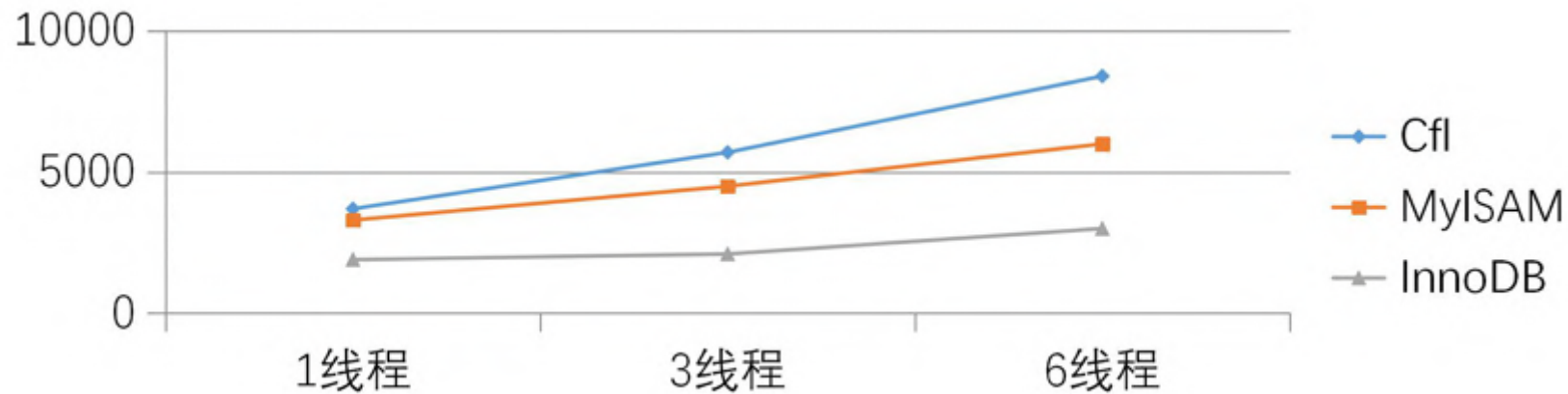


时间序列存储引擎-运行视图



时间序列存储引擎-插入性能对比

引擎\线程	1线程 (ips)	3线程 (ips)	6线程 (ips)
Cfl	3700	5700	8400
MyISAM	3300	4500	6000
InnoDB	1900	2100	3000
2核ssd虚拟机			



时间序列存储引擎-源码

- MySQL文件：handler.h/handler.c
- Cfl文件：

```
cf1_ato.h      cf1_dt.h      cf1_insert_buffer.cc  cf1_table.h
cf1_buffer.cc  cf1_endian.h  cf1_insert_buffer.h  CMakeCache.txt
cf1_buffer.h   cf1_file.cc   cf1_page.cc          CMakeFiles
cf1_cursor.cc  cf1_file.h    cf1_page.h           CMakeLists.txt
cf1_cursor.h   cf1_h         cf1_row.cc           ha_cf1.cc
cf1_data.cc    cf1_index.cc  cf1_row.h            ha_cf1.h
cf1_data.h     cf1_index.h   cf1_table.cc         readme
```

- 程序片段

```
class handler :public Sql_alloc
{
public:
    typedef unsigned long Table_flags;
protected:
    TABLE_SHARE *table_share;           /* The table definition */
    TABLE *table;                       /* The current open table */
    Table_flags cached_table_flags;      /* Set by init() and open() */

    int ha_open(TABLE *table, const char *name, int mode, int test_if_locked);
    int ha_close(void);
    int ha_index_init(uint idx, bool sorted);
    int ha_index_end();
    int ha_rnd_init(bool scan);
    int ha_rnd_end();
    int ha_rnd_next(uchar *buf);
    int ha_rnd_pos(uchar *buf, uchar *pos);
};
```

```
class ha_cf1: public handler
{
    THR_LOCK_DATA lock;           /* MySQL lock
    Cf1_share *share;             /* Shared lock info
    Cf1_share *get_share();       /* Get the share
```

渐入佳境

5.7.17

- 移植功能
 - 5.6.21的线上功能
- 新功能
 - 记录binlog中DML的发起者 (invoker)
 - mysqlfbtool, flashback结果转换为可执行的SQL语句的工具
 - 保留连接的改造
 - performance shcema功能改进
 - RPM打包开发

binlog中的invoker

- 修改目的
 - 记录是谁进行了DML操作
 - 为后续的按用户操作回滚提供基础
- 效率问题
 - 以事务为单元进行记录
 - 写入量占比极低

mysqlfbtool

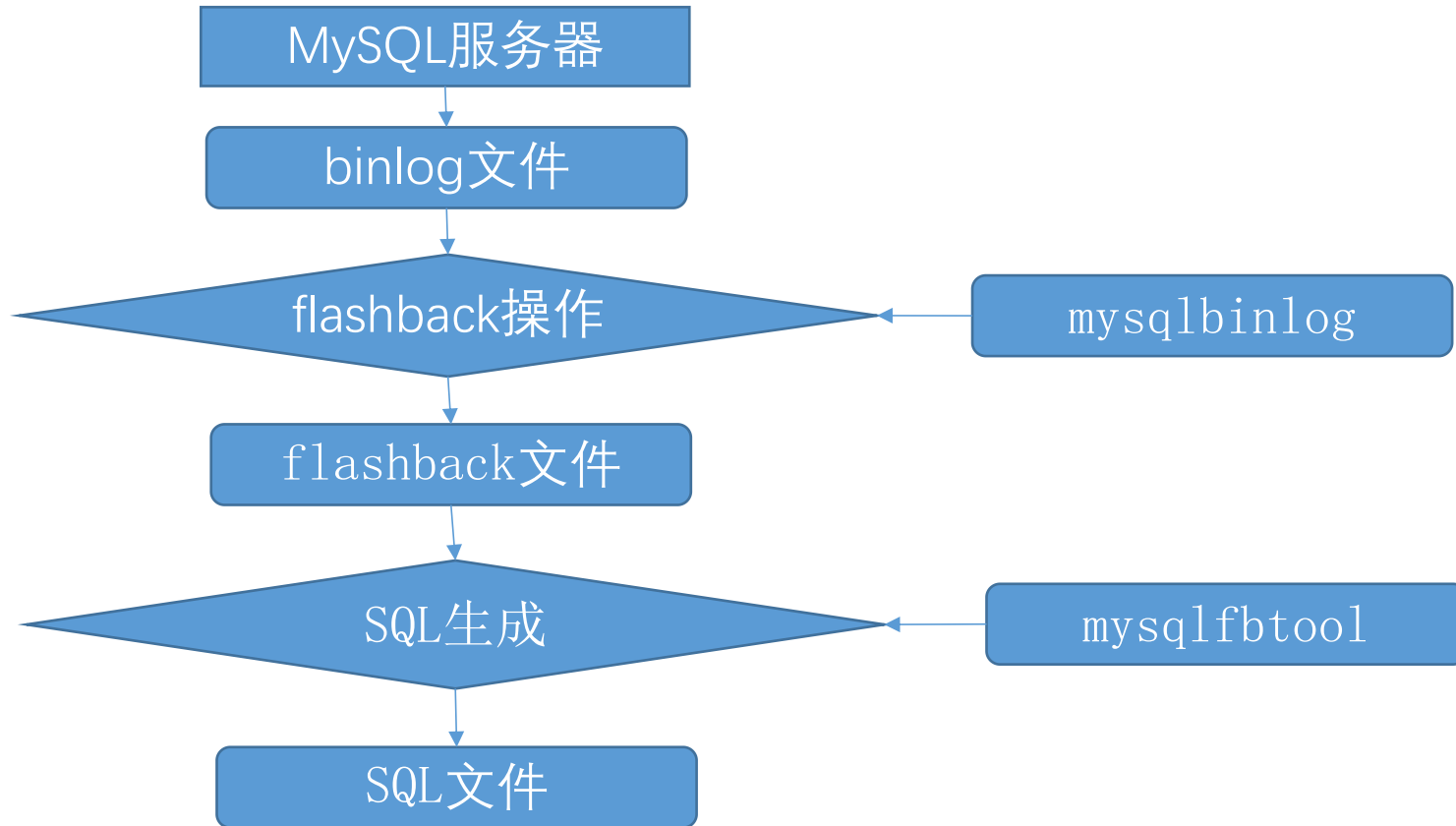
- 将mysqlbinlog工具解析出的语句还原为SQL

```
./bin/mysqlbinlog -v -B data/mysql-bin.000001 > mysql-bin.000001.fb
```

```
### flashback  
### DELETE FROM `test`.`t1`  
### WHERE  
###   @1=1  
###   @2='1'  
# -# 774
```

```
./bin/mysqlfbtool -uroot -p -S mysql.sock mysql-bin.000001.fb  
Enter password:  
delete from test.t1 where f1=1 and f2='1';
```


Invoker&Flashback&Mysqlfbtool



保留连接的改造

- MySQL达到最大连接数后，允许1个超级用户进行连接

`mysql` actually permits `max_connections+1` clients to connect. The extra connection is reserved for use by accounts that have the `SUPER` privilege. By granting the `SUPER` privilege to administrators and not to normal users (who should not need it), an administrator can connect to the server and use `SHOW PROCESSLIST` to diagnose problems even if the maximum number of unprivileged clients are connected. See Section 13.7.5.30, "SHOW PROCESSLIST Syntax".

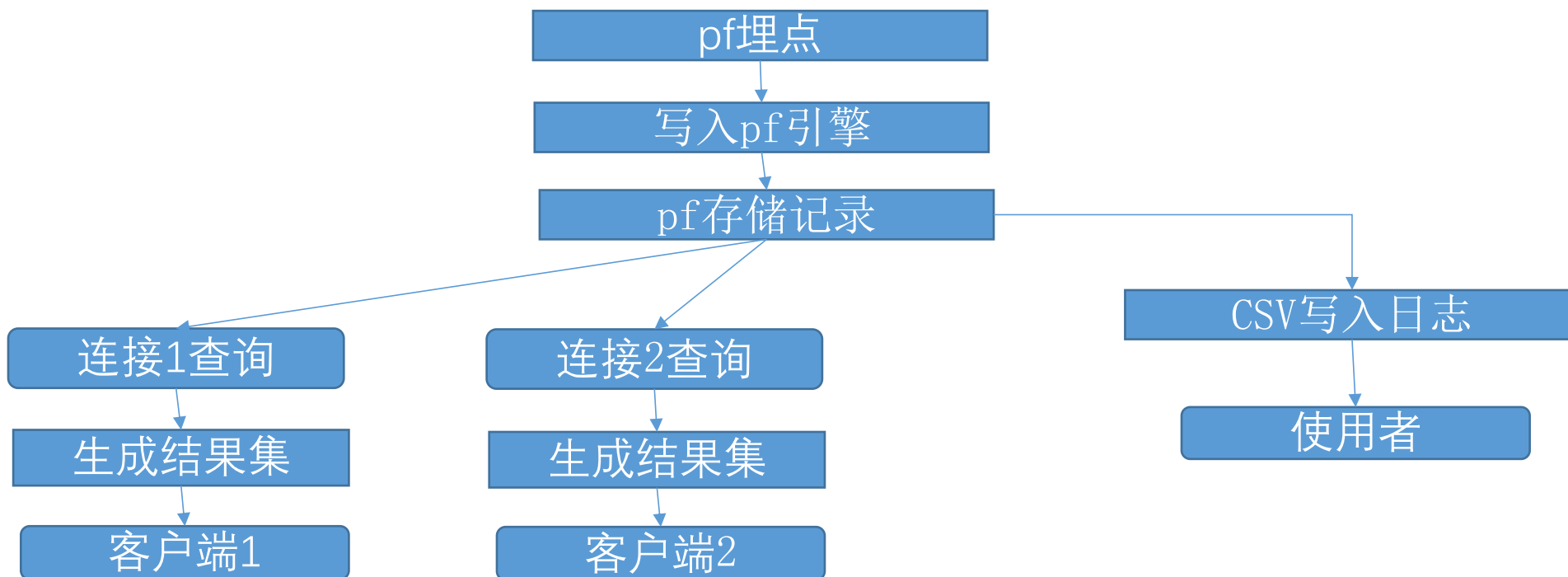
- 增加配置项，预留更多的连接

```
max-connections = 5  
reserved-connections = 2
```

performance shcema改进

- 目标
 - 期望如SQL Server一样生成语句的general log
- 效率问题
 - events_statements_history_long的数据针对每次查询都要重新生成
- 功能
 - 在events_statements系列表中增加host&user的输出
 - 运行中动态配置events_statements事件的记录写盘
 - 避免开启events_statements_history_long的读写效率问题
 - 便于备份和使用

performance shcema的运行图



RPM打包开发

- 问题
 - 5.7.17无法按照5.6的操作进行打包
- 解决
 - 仿照5.6.X的打包方式重新编写mysql.spec.sh
- 特色
 - 增加了携程定制的打包内容，如mysqlfbtool以及名字等

总结

- 版本
 - 5.6.12/5.6.21/5.7.17
- 功能点
 - 11个线上运行功能
 - 1个探索性开发

源码改造经验

- 太深的源码改造将影响版本升级。在享受新版本和自定制中存在一个平衡。由此，总结下来一些原则减少版本升级中的问题
 - 1、尽可能的利用MySQL原有的函数
 - 2、尽可能通过插件来完成功能
 - 3、如果不可避免要进行代码修改，则尽可能通过新增函数/源文件方式
 - 4、尽可能不要尝试去解析并修改数据内容
 - 5、尽可能不要触动MySQL的运行机制

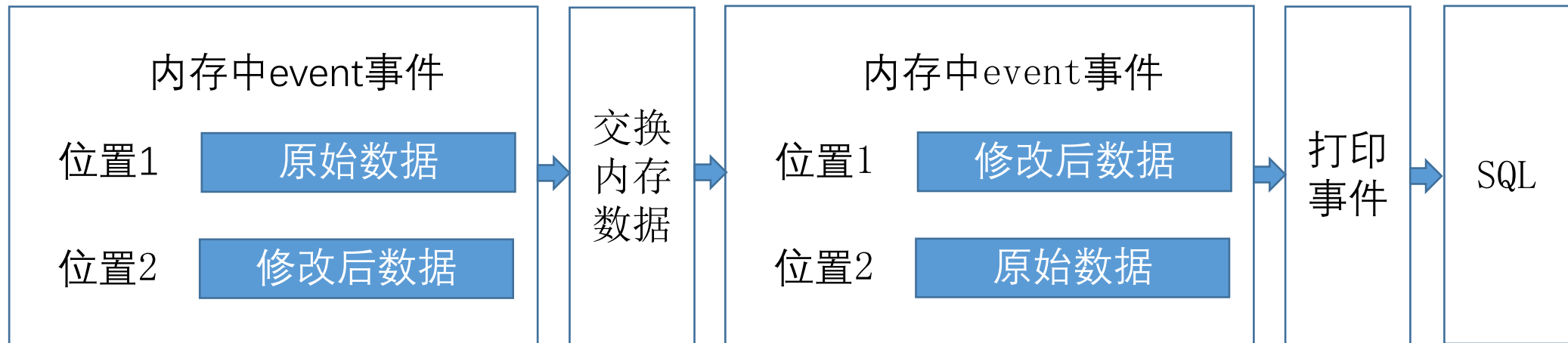
异步复制的移植失败

- 遇到问题
 - 5.6.12代码移植到5.6.21后，MySQL服务器出现宕机
- 无法定位问题
 - 由于错误出现位置的随机性，而且出现问题的间隔较长，无法准确重现问题，最终只能放弃这个功能
- 分析
 - 该修改改变了复制分发的运行机制，当代码出现变化后，在原理层面就无法
- 教训
 - 尽可能避免触动MySQL的运行机制

flashback的两次移植的差异

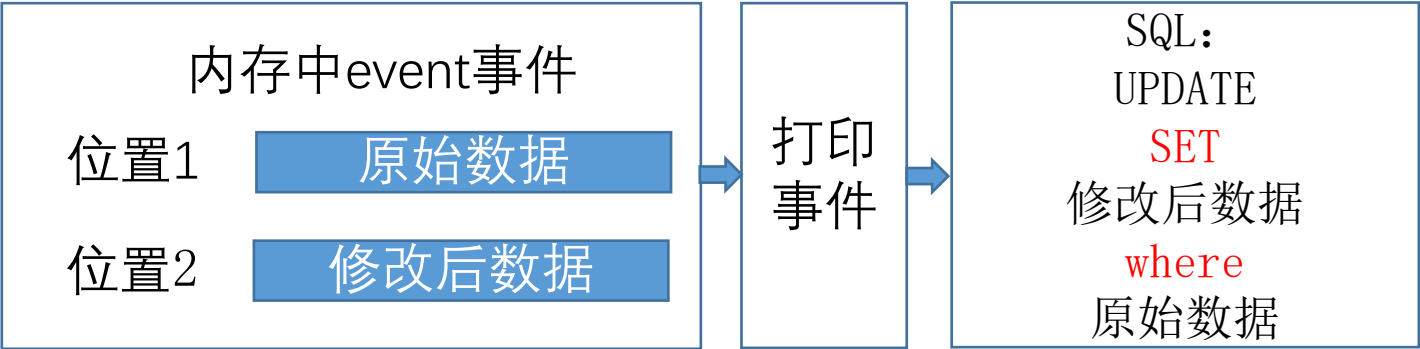
- UPDATE回滚的两个思路
 - 在内存中互换行数据
 - 在输出文本中交换set和where的位置
- 5.6.X上的移植
 - 代码相近，写入文件的binlog事件完全一致，patch工作相对简单
- 5.7.17上的移植
 - 新增数据类型增加，需要继续升级
- 得失
 - 前者尽可能还原数据为binlog模式，其结果可直接由mysql使用进行还原工作，但在升级过程中需要更多的修改MySQL代码
 - 后者减少对MySQL代码的修改，但只能处理-v参数所生成的文本结果，失去原有的便利性
- 总结
 - 尽可能少改动MySQL源码，把更多工作留给外部的工具

5.6.X

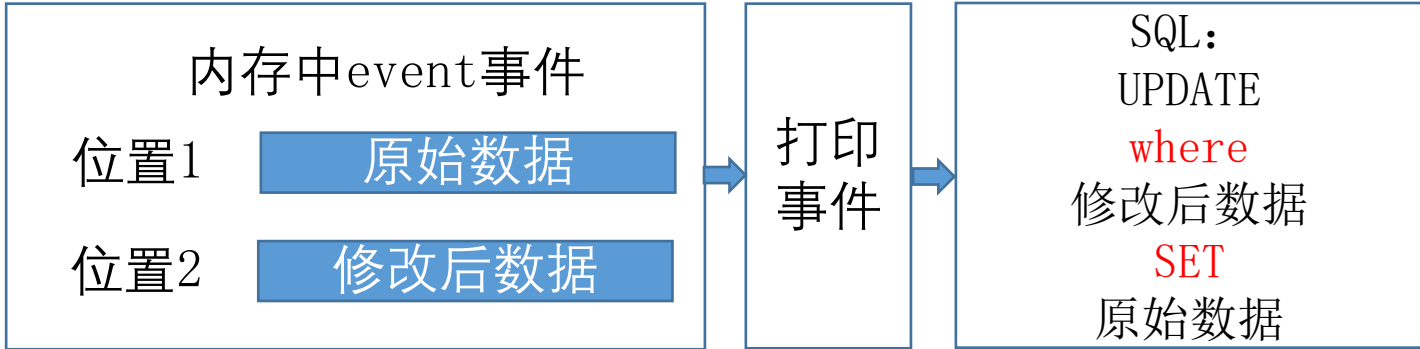


5.7.X

正常打印方式



回滚打印方式



路在远方

源码地址

- 5.6.12
 - <https://github.com/ctripopsdba/mysql-5.6.12>
- 5.6.21
 - <https://github.com/ctripopsdba/mysql-5.6.21>
- 5.7.17
 - <https://github.com/ctripopsdba/mysql-5.7.17>

