ZHDBA.COM
中华数据库行业协会

MySQL DBA
甩锅特技集锦

个人介绍

拥有丰富的一二线MySQL运维经验；

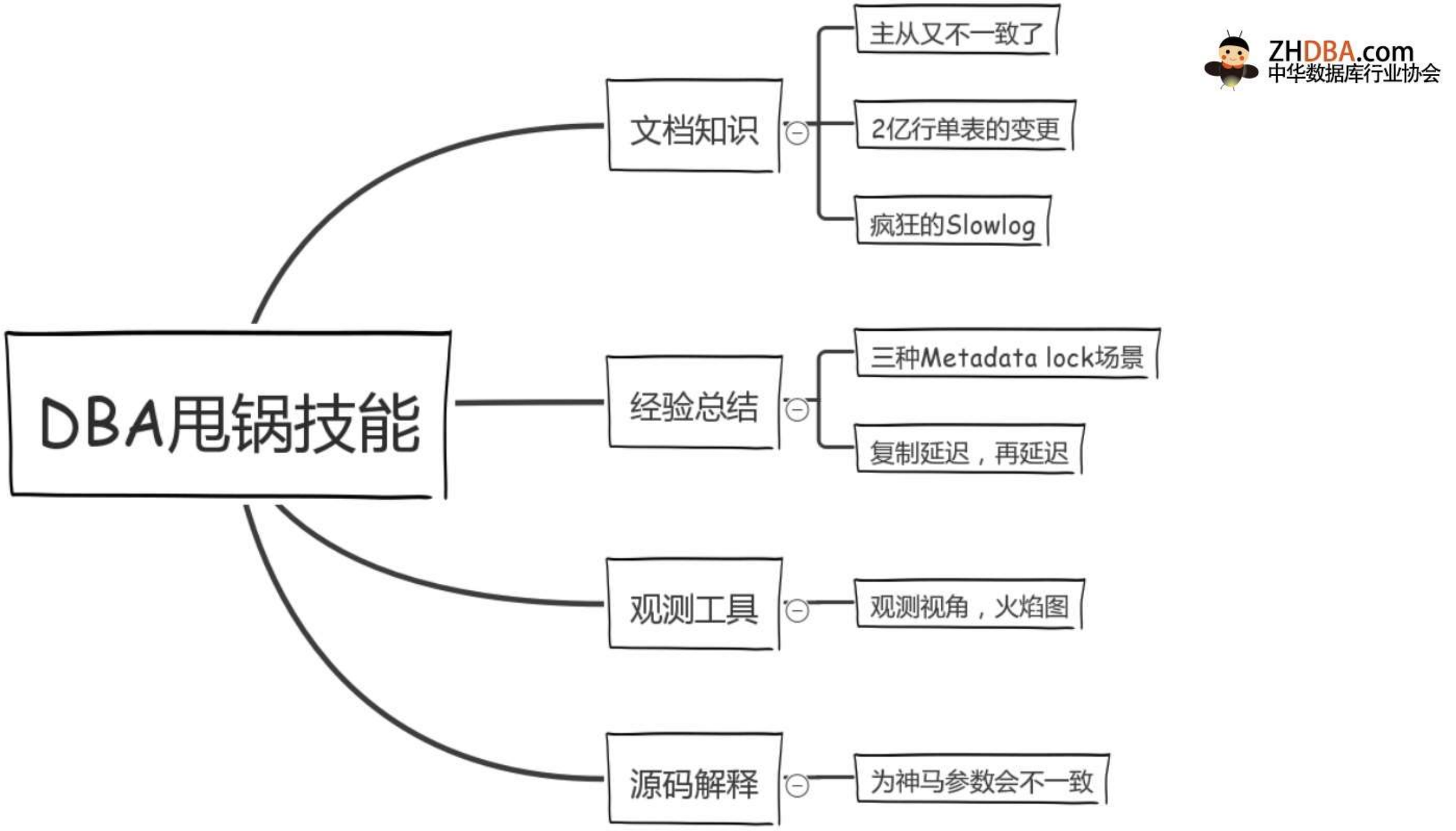
先后在阿里云，爱可生担任数据库运维；

目前为爱可生数据库产品负责人，MySQL技术专家；

上海爱可生信息技术股份有限公司（证券代码：832768）是国内领先的企业级数据处理技术和开源数据库整体解决方案提供商，致力于为产业互联网打造大数据和云计算应用所需的便捷、高效的数据处理核心能力，已成为国内数据处理领域的重要新兴技术力量。

中华数据库与运维大会技术交流群

ZHDBA.com
中华数据库行业协会



1.1 主从又不一致了

老生常谈的参数

ZHDBA.com
中华数据库行业协会

一致性相关

innodb_flush_log_at_trx_commit=1

sync_binlog=1

复制安全

relay_log_info_repository = TABLE

master_info_repository = TABLE

sync_relay_log = 1

sync_master_info = 1

影响复制安全的参数

故事背景

MySQL 5.6.24

- 主实例Binlog磁盘满
- 磁盘清理

影响面

- 主从实例不同步，数据不一致现象；
- 重做所有从实例

原因

[ERROR] Could not open ./master-bin.000064 for logging (error 13). Turning logging off for the whole duration Of the MySQL server process. To turn it on again: fix the cause,shutdown the MySQL server and restart it.

影响复制安全的参数

分析

binlog_error_action

- ignore_error
- abort_server

验证：ABORT_SERVER

Error message: Binary logging not possible. Either disk is full or file system is read only while rotating the binlog. Aborting the server

影响复制安全的参数

结论

- 5.6.22版本增加了Abort_server选项
- 5.6版本默认值为IGNORE_ERROR
- 5.7版本默认值为ABORT_SERVER
- 该参数支持动态修改
- 推荐生产环境配置ABORT_SERVER选项

1.2 2亿行单表的故事

一张2亿行单表的故事

ZHDBA.com
中华数据库行业协会

故事背景

CPU：64c Mem：256G Disk：SSD

2亿行70G单表上做了alter table加字段操作，报错如下：

```
mysql> alter table friend add column tag varchar(255)  comment ‘好友对应的tags’;
InnoDB: Error: Write to file (merge) failed at offset 3039821824.
InnoDB: 1048576 bytes should have been written, only 471040 were written.
InnoDB: Operating system error number 0.
InnoDB: Check that your OS and file system support files of this size.
```

影响面

不要跟我谈影响面

泡面都凉了！

一张2亿行单表的故事

分析

问题直接原因：/tmp满了，重新配置了tmpdir

2亿3000W条记录:

```
mysql> alter table friend add column tag varchar(255)
comment '好友对应的tags';
Query OK, 0 rows affected (1 hour 48 min 36.33 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

一张2亿行单表的故事

问题1：何时产生临时表？

alter table
- table copy -> #sql-
- INPLACE -> #sql-ib/tmpdir

复杂排序
- Order by
- Group by

参考文献：https://dev.mysql.com/doc/refman/5.7/en/temporary-files.html

一张2亿行单表的故事

问题2：alter table添加列属于哪种操作

Operation	In-Place?	Rebuilds Table?	Permits Concurrent DML?	Only Modifies Metadata?	Notes
Add column	Yes*	Yes*	Yes*	No	Concurrent DML is not permitted when adding an auto-increment column. Data is reorganized substantially, making it an expensive operation. ALGORITHM=INPLACE is supported for adding a virtual generated column but not for adding a stored generated column. Adding a virtual generated column does not require a table rebuild.
Drop column	Yes	Yes*	Yes	No	Data is reorganized substantially, making it an expensive operation. ALGORITHM=INPLACE is supported for dropping a generated column. Dropping a virtual generated column does not require a table rebuild.
Change column data type	No*	Yes	No	No	VARCHAR size may be increased using online ALTER TABLE. See Modifying Column Properties for more information.

参考文献：https://dev.mysql.com/doc/refman/5.6/en/innodb-create-index-overview.html

一张2亿行单表的故事

问题3:是否有其他的DDL方式

mysql：online ddl

pt-osc：percona online schema change

gh-ost：GitHub's online schema migration for MySQL

参考文献：

https://dev.mysql.com/doc/refman/5.6/en/innodb-online-ddl.html
https://www.percona.com/doc/percona-toolkit/2.1/pt-online-schema-change.html
https://github.com/github/gh-ost

一张2亿行单表的故事

总结

1. tmpdir在生产上线时提前规划，建议单独配置足够容量的目录

2. 操作ddl前建议根据alter table的类型，对照官方文档确认是inplace 还是table copy

一张2亿行单表的故事

总结

3. DDL操作方式的原则

表数据量大小

最直接影响选择ddl方式的因素，建议超过100万行的表，通过posc或gh-ost操作

是否可以接受延迟

接受延迟的环境，表小于100万，用原生

不接受延迟的环境，使用posc或gh-ost

1.3 疯狂的Slowlog

疯狂的SlowLog

谁见过long_query_time设置不生效的情况吗?

设置10s，100s；慢日志照样记录0.002s的sql

上次看到log_queries_not_using_indexes很有用，把它设成on了

受log_queries_not_using_indexes影响了

万万没想到所有的表只有主键

疯狂的SlowLog

To include queries that do not use indexes for row lookups in the statements written to the slow query log, enable the `log_queries_not_using_indexes` system variable. When such queries are logged, the slow query log may grow quickly. It is possible to put a rate limit on these queries by setting the `log_throttle_queries_not_using_indexes` system variable. By default, this variable is 0, which means there is no limit. Positive values impose a per-minute limit on logging of queries that do not use indexes. The first such query opens a 60-second window within which the server logs queries up to the given limit, then suppresses additional queries. If there are suppressed queries w... the window ends, the server logs a summary that indicates how many there were and the aggregate... them. The next 60-second window begins when the server logs the next query that does not use i...

grow quickly

The server uses the controlling parameters in the following order to determine whether to write a que... query log:

1. The query must either not be an administrative statement, or `log_slow_admin_statements` must be enabled.
2. The query must have taken at least `long_query_time` seconds, or `log_queries_not_using_indexes` must be enabled and the query used no indexes for row lookups.
3. The query must have examined at least `min_examined_row_limit` rows.
4. The query must not be suppressed according to the `log_throttle_queries_not_using_indexes` setting.

疯狂的SlowLog

正确的姿势应该是这样的

```
#-----------------MySQL Log Setting-----------------#
log_error                                =   mysql-error.log
log_bin                                  =   mysql-bin.log
slow_query_log_file                      =   mysql-slow.log
relay_log                                =   mysql-relay.log
log_slave_updates                        =   1
sync_binlog                              =   1
relay_log_recovery                       =   1
binlog_format                            =   row
expire_logs_days                         =   30
slow_query_log                           =   1
long_query_time                          =   2
log_queries_not_using_indexes            =   1
log_throttle_queries_not_using_indexes   =   10
log_slow_admin_statements                =   1
log_slow_slave_statements                =   1
min_examined_row_limit                   =   1000
```

2.1 偶遇Metadata lock

偶遇metadata lock

Metadata lock wait 出现的场景

- 有大事务正在使用表
- 存在有未提交的事物
- 存在有未提交的事务,且是失败操作

解决方案

- 找到未释放metadata lock的session，并kill
- Kill ddl语句，等待下一次运维时间变更

偶遇metadata lock

ZHDBA.com 中华数据库行业协会

Metadata lock wait 出现的场景1

有大事务正在使用表

```
mysql> mysql> show processlist;
+----+------+-----------+-----------+---------+------+--------------------------------+-----------------------------------------------------+
| Id | User | Host      | db        | Command | Time | State                          | Info                                                |
+----+------+-----------+-----------+---------+------+--------------------------------+-----------------------------------------------------+
| 11 | root | localhost | NULL      | Query   |    0 | starting                       | show processlist                                    |
| 12 | root | localhost | actionsky | Query   |    3 | Waiting for table metadata lock | select * from action                                |
| 16 | root | localhost | actionsky | Query   |   16 | User sleep                     | select * from action where name1 like "%a" or sleep(2000) |
| 18 | root | localhost | actionsky | Query   |    9 | Waiting for table metadata lock | alter table action add (name2 char(20))             |
+----+------+-----------+-----------+---------+------+--------------------------------+-----------------------------------------------------+
4 rows in set (0.00 sec)
```

ZHDBA.com
中华数据库行业协会

偶遇metadata lock

Metadata lock wait 出现的场景2

存在有未提交的事务

```
Query OK, 0 rows affected (0.00 sec)

mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from actionsky.action;
+-------+--------+
| id    | name1  |
+-------+--------+
|     1 | simple |
+-------+--------+
1 row in set (0.00 sec)
```

1

```
@node1:/ (ssh)
mysql>
mysql>
                    mysql> alter table action add (name2 char(20));
```

2

```
@node1:/ (ssh)
                 |
| 12 | root | localhost | actionsky | Sleep |  482 |                                  | NULL
                 |
| 19 | root | localhost | actionsky | Sleep |   79 |                                  | NULL
                 |
| 21 | root | localhost | actionsky | Query |   28 | Waiting for table metadata lock | alter table action ad
d (name2 char(20)) |
+----+------+-----------+-----------+-------+------+---------------------------------+---------------------
-------------------+
4 rows in set (0.00 sec)
```

3

```
mysql> select * from information_schema.innodb_trx\G
*************************** 1. row ***************************
                    trx_id: 421531794836192
                 trx_state: RUNNING
               trx_started: 2017-05-14 22:29:39
     trx_requested_lock_id: NULL
          trx_wait_started: NULL
                trx_weight: 0
       trx_mysql_thread_id: 19
                 trx_query: NULL
       trx_operation_state: NULL
         trx_tables_in_use: 0
         trx_tables_locked: 0
          trx_lock_structs: 0
     trx_lock_memory_bytes: 1136
           trx_rows_locked: 0
         trx_rows_modified: 0
   trx_concurrency_tickets: 0
       trx_isolation_level: REPEATABLE READ
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
 trx_adaptive_hash_latched: 0
 trx_adaptive_hash_timeout: 0
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
1 row in set (0.00 sec)

mysql>
```

4

偶遇metadata lock

Metadata lock wait 出现的场景3

存在有未提交的事务,且是失败操作

```
Query OK, 0 rows affected (0.00 sec)

mysql> select b from actionsky ;
ERROR 1146 (42S02): Table 'actionsky.actionsky' doesn't exist
mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> select b from action ;
ERROR 1054 (42S22): Unknown column 'b' in 'field list'
mysql>
```

1

@node1:/(ssh)

```
Database changed
mysql> alter table action add (name2 char(20));
```

2

@node1:/(ssh)

```
+----+------+-----------+-----------+---------+------+--------------------------------+----------------------+
| Id | User | Host      | db        | Command | Time | State                          | Info                 |
+----+------+-----------+-----------+---------+------+--------------------------------+----------------------+
| 11 | root | localhost | NULL      | Query   |    0 | starting                       | show processlist     |
| 12 | root | localhost | actionsky | Sleep   |   87 |                                | NULL                 |
| 19 | root | localhost | actionsky | Sleep   |  192 |                                | NULL                 |
| 23 | root | localhost | actionsky | Query   |  159 | Waiting for table metadata lock | alter table action ad
d (name2 char(20)) |
+----+------+-----------+-----------+---------+------+--------------------------------+----------------------+
4 rows in set (0.00 sec)
```

3

```
mysql> select * from information_schema.innodb_trx\G
Empty set (0.00 sec)

mysql>
```

4

@node1:/(ssh)

```
mysql> select * from performance_schema.events_statements_curr
ent where Errors=1\G;
*************************** 1. row ***************************
              THREAD_ID: 46
               EVENT_ID: 20
           END_EVENT_ID: 20
             EVENT_NAME: statement/sql/select
                 SOURCE: socket_connection.cc:98
            TIMER_START: 283295202799802000
              TIMER_END: 283295203072498000
             TIMER_WAIT: 272696000
              LOCK_TIME: 0
               SQL_TEXT: select b from action
                 DIGEST: a660d6e6d85b1fecd6e12efd6b067681
            DIGEST_TEXT: SELECT `b` FROM ACTION
         CURRENT_SCHEMA: actionsky
            OBJECT_TYPE: NULL
          OBJECT_SCHEMA: NULL
            OBJECT_NAME: NULL
  OBJECT_INSTANCE_BEGIN: NULL
            MYSQL_ERRNO: 1054
      RETURNED_SQLSTATE: 42S22
          MESSAGE_TEXT: Unknown column 'b' in 'field list'
                ERRORS: 1
              WARNINGS: 0
          ROWS_AFFECTED: 0
              ROWS_SENT: 0
          ROWS_EXAMINED: 0
CREATED_TMP_DISK_TABLES: 0
     CREATED_TMP_TABLES: 0
```

5

2.2 复制延迟，再延迟

复制延迟的几种可能

- 主从服务器的配置或者压力不一，从实例的IO能力太弱
- 主库的TPS太高导致备库延迟
- 备份Waiting for global read lock
- DDL导致备库延迟（alter table,create index,optimize table,repair table）
- 运行大事务
- 对无主键的表进行删除或者更新

运行大事务

ZHDBA.com
中华数据库行业协会

背景

max_binlog_size=512M

```
-rw-r----- 1 mysql mysql  512M May 11 10:17 mysql-bin.000063
-rw-r----- 1 mysql mysql  512M May 11 15:59 mysql-bin.000064
-rw-r----- 1 mysql mysql 1220M May 12 22:23 mysql-bin.000065
```

缺少主键

背景

Binlog：Row模式，没有主键，表数据量3000万左右

主上执行

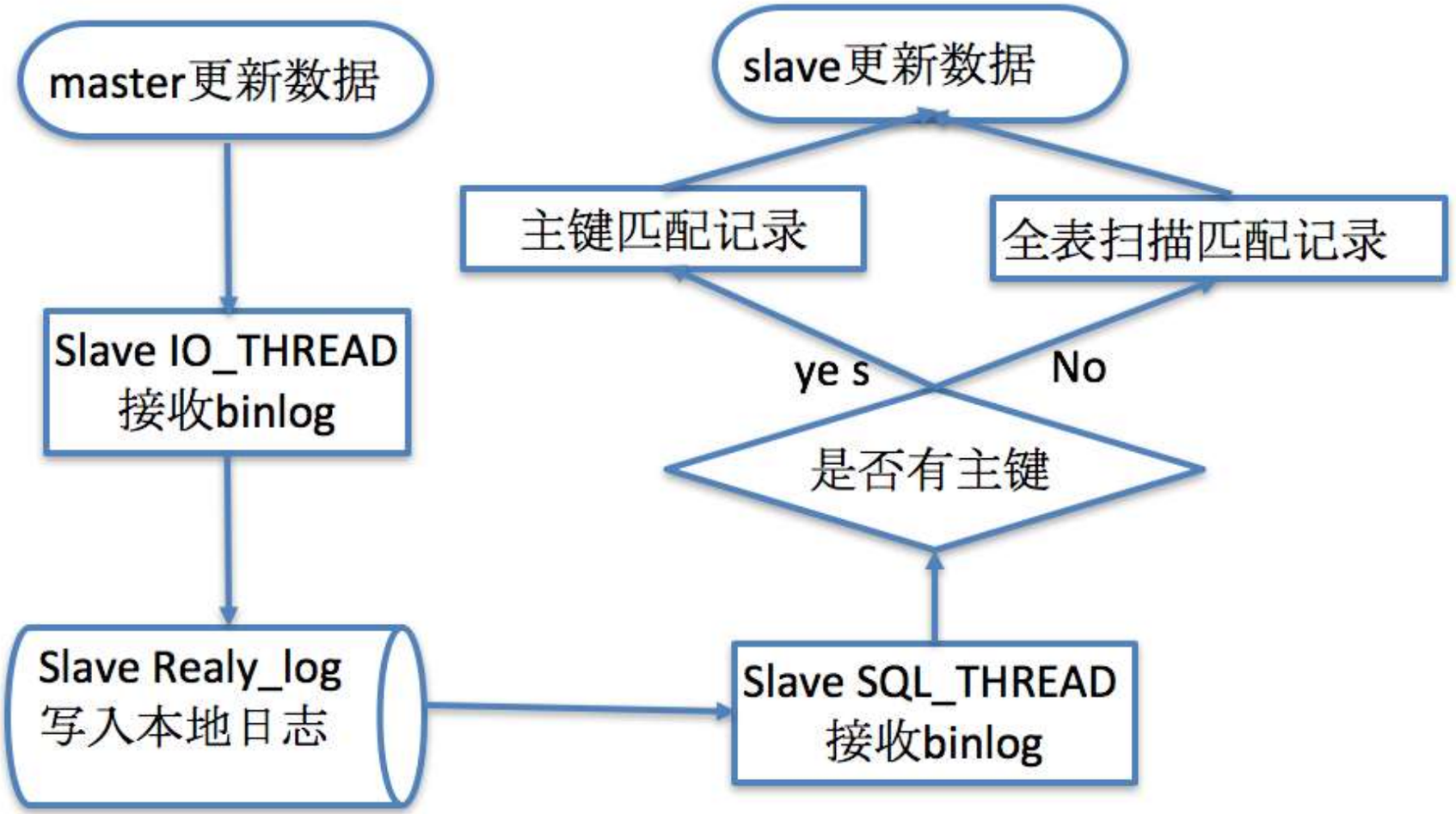
```
DELETE FROM `***_snsdb`.`***_gamedata_service`
```

缺少主键

mysqlbinlog -vv --base64-output=decode-rows mysqld-relay-bin.000065 --start-position=542989459|less

```
#160526  8:00:10 server id 9749  end_log_pos 549683948  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549684890  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549686239  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549687786  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549688727  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549690205  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549691161  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549692381  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549694156  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549695112  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549696793  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549697910  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549699223  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549700287  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549701959  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549703860  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549705070  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549706026  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549707145  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549708259  Delete_rows: table id 19991
#160526  8:00:10 server id 9749  end_log_pos 549709293  Delete_rows: table id 19991
```

缺少主键

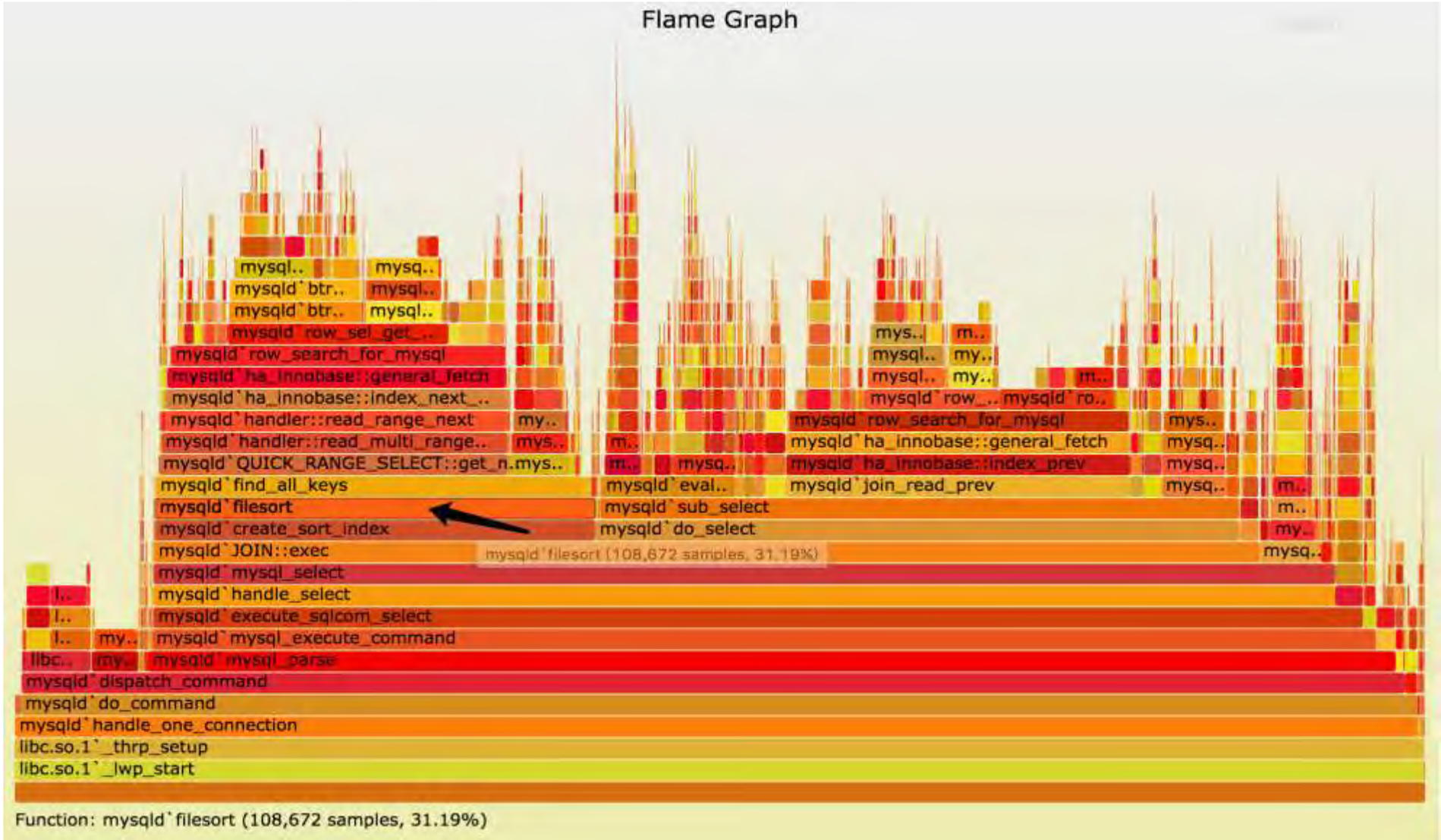


3.0 观测视角，火焰图

背景知识

Different types of flame graphs:

- On-CPU
- Off-CPU
- Hot/Cold
- Differential
- Memory



背景知识

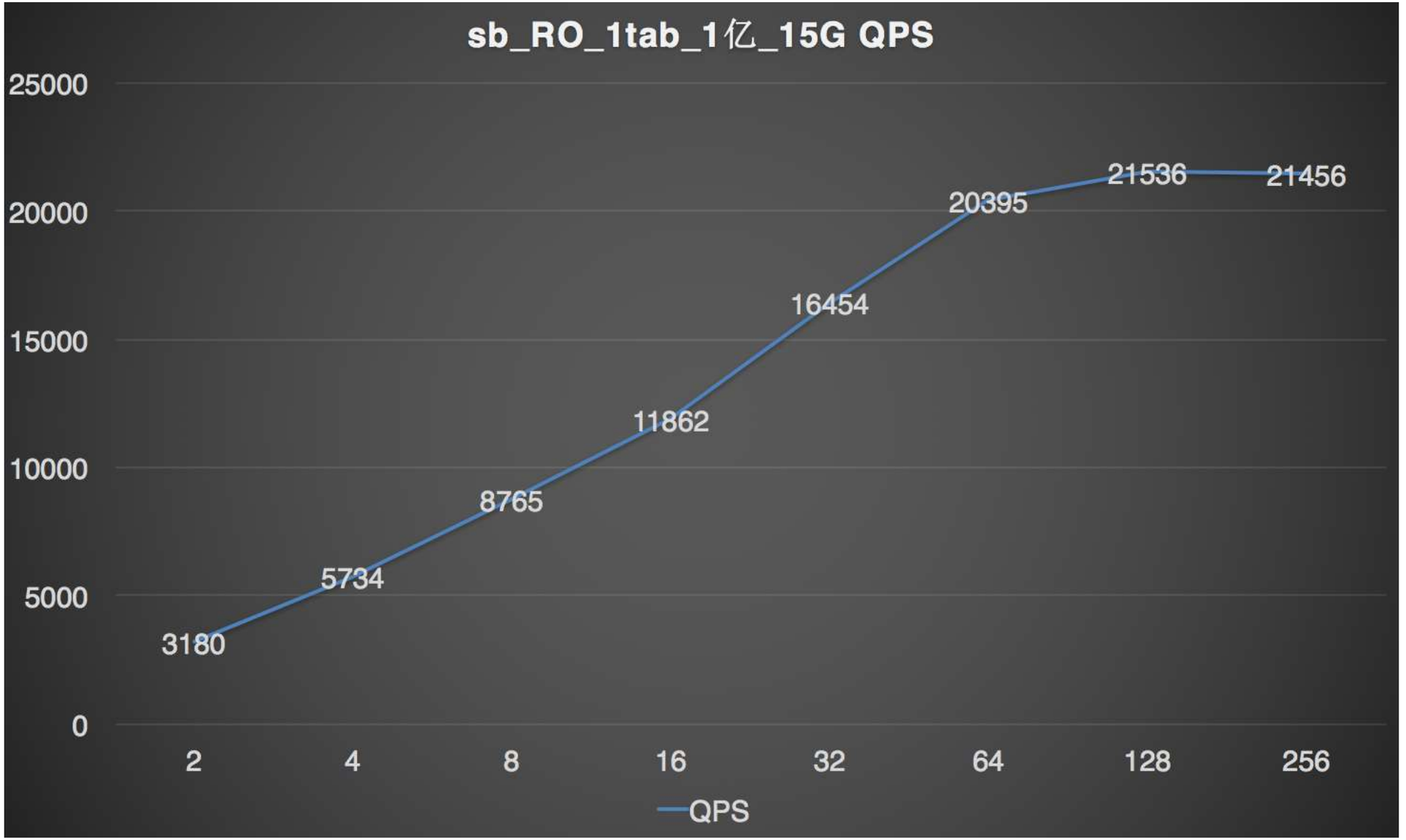
Profiling tools:

Linux:	perf, SystemTap, …
Solaris, illumos, FreeBSD:	DTrace
Mac OS X:	DTrace and Instruments
Windows:	Xperf.exe

参考文献：

Mysqld DTrace Probe Reference
https://dev.mysql.com/doc/refman/5.7/en/dba-dtrace-mysqld-ref.html
用Systemtap探索MySQL
http://www.actionsky.com/docs/archives/168

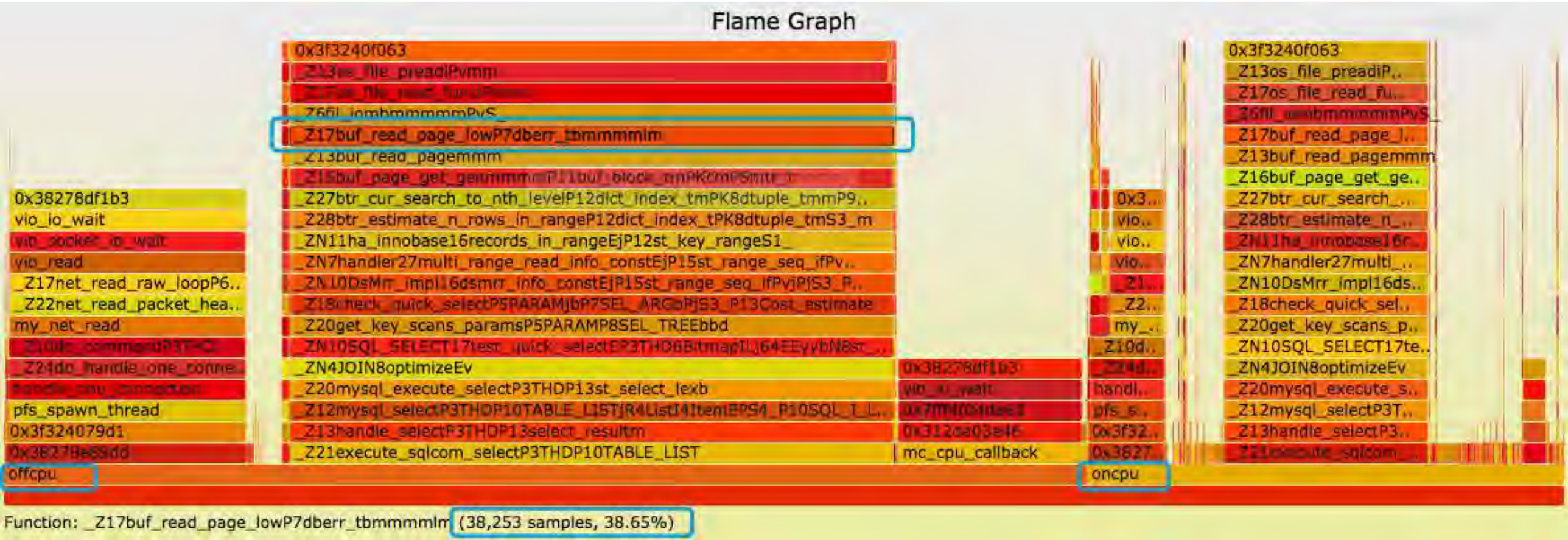
场景1：基准测试



Mysql5.7测试：
总规模1亿行
数据量大小15G
单表只读
4Gbuffer
其他默认

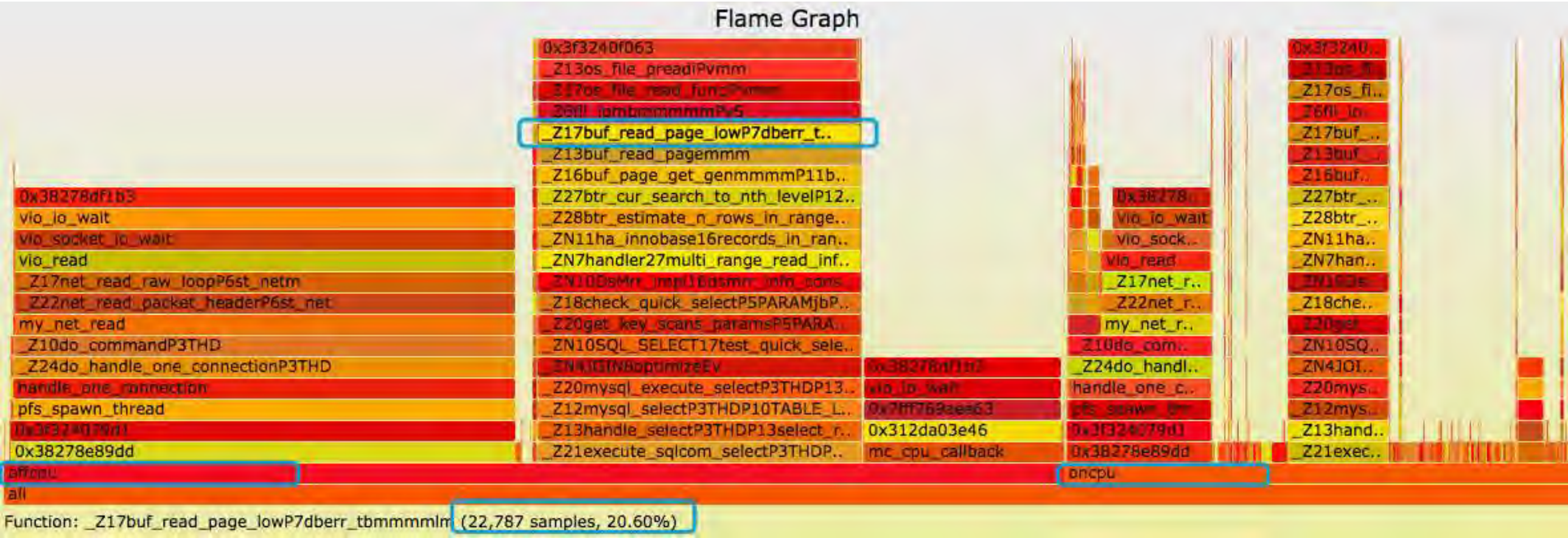
场景1：基准测试

原生mysql测试，总数据量1亿， 数据量大小15G, 4Gbuffer，其他默认

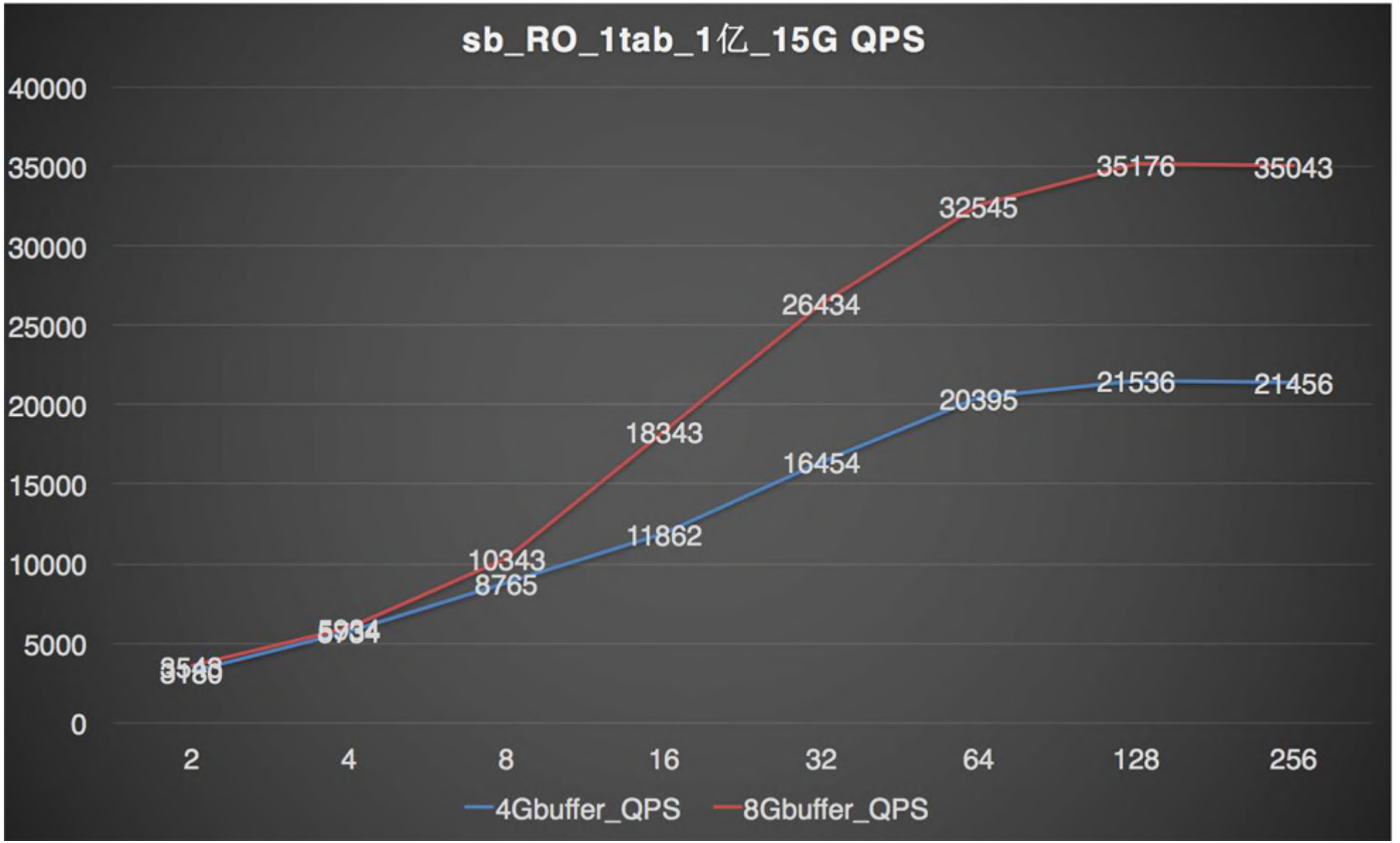


场景1：基准测试

原生mysql测试，总数据量1亿， 数据量大小15G, 8Gbuffer



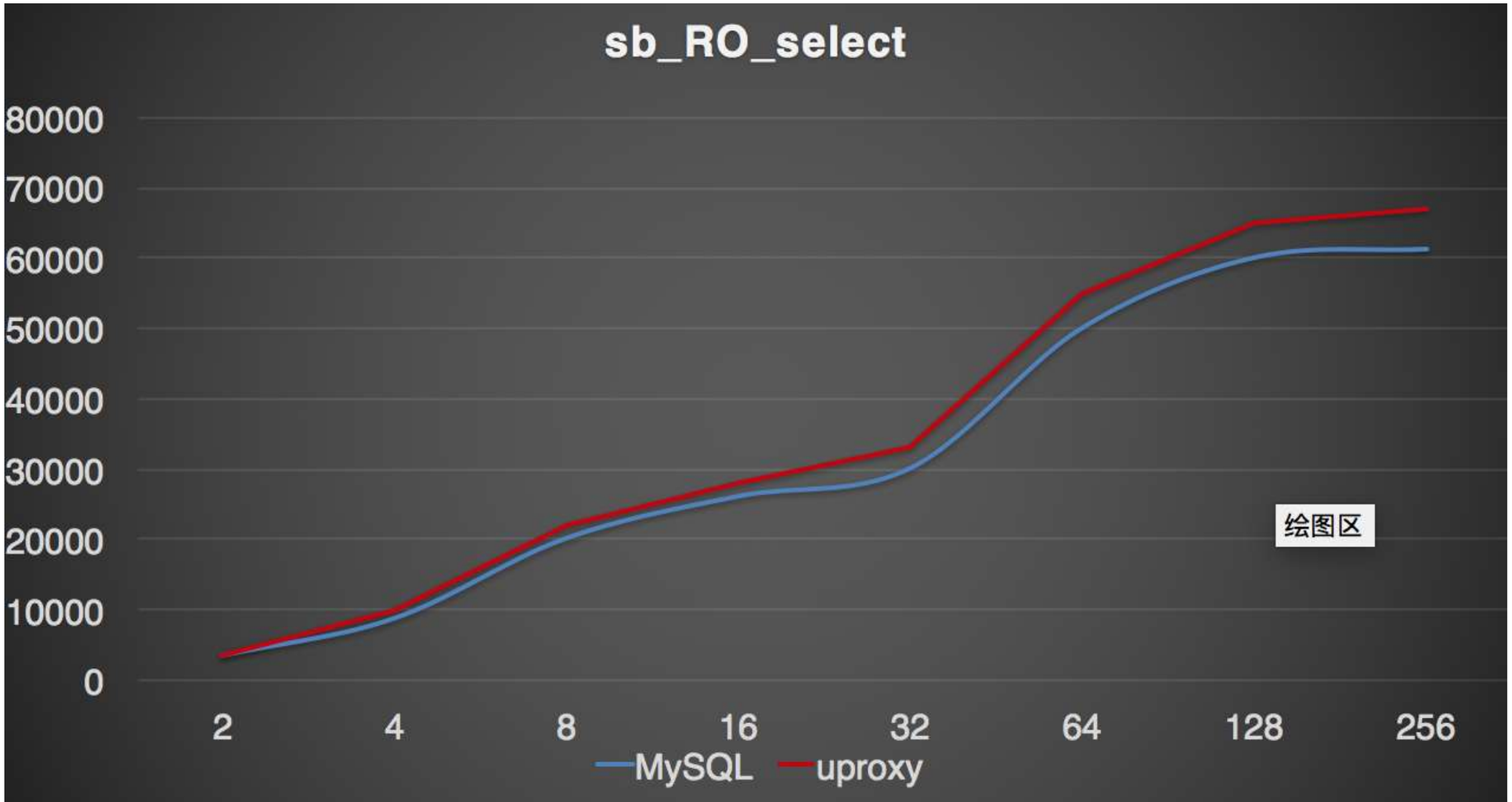
场景1：基准测试



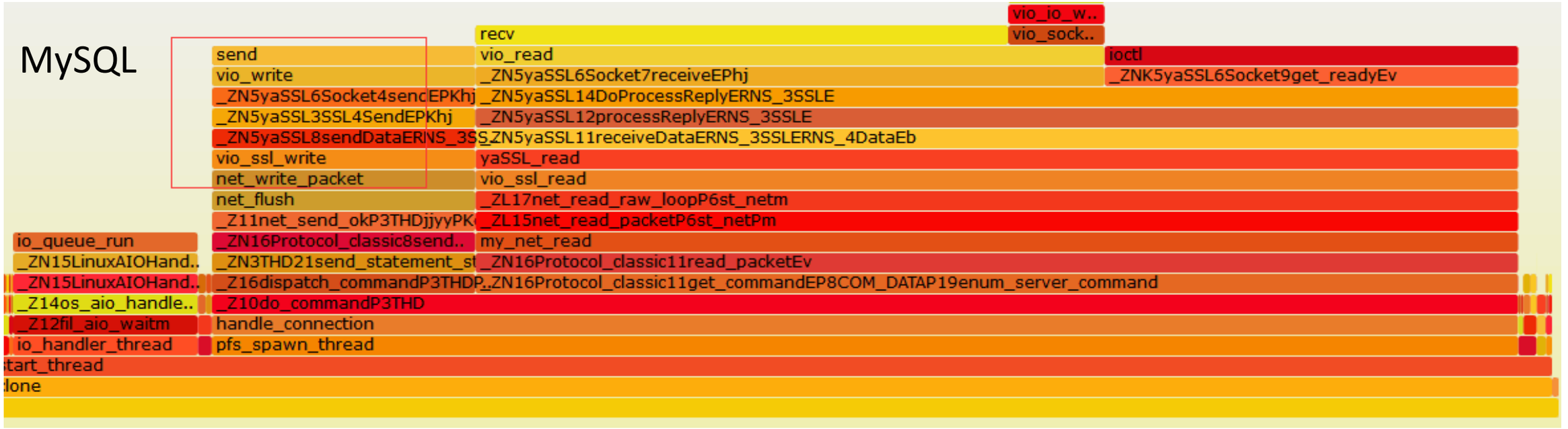
Mysql5.7测试：
总规模1亿行
数据量大小15G
单表只读
4G/8G buffer

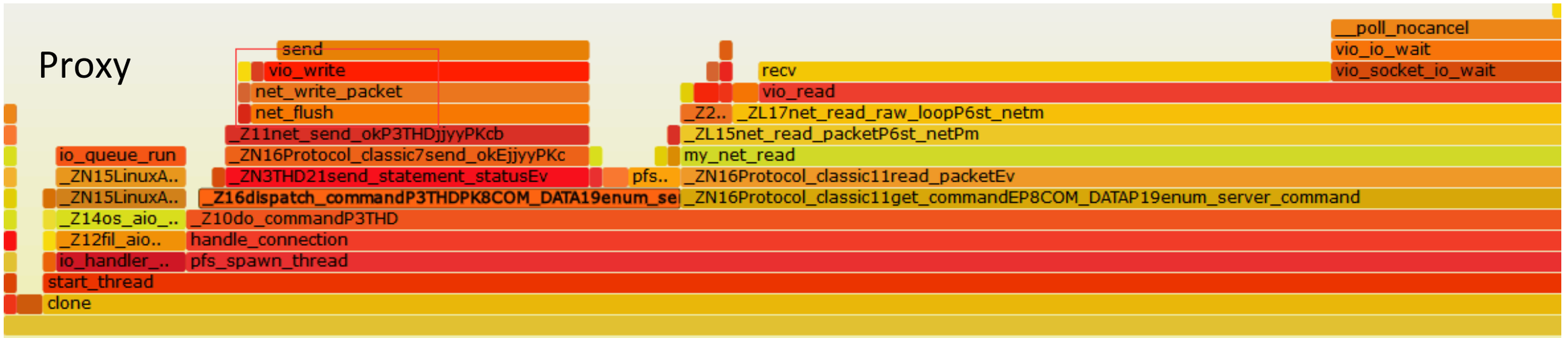
场景2：中间件性能损失测试

单实例5.7 VS 中间件+单实例5.7

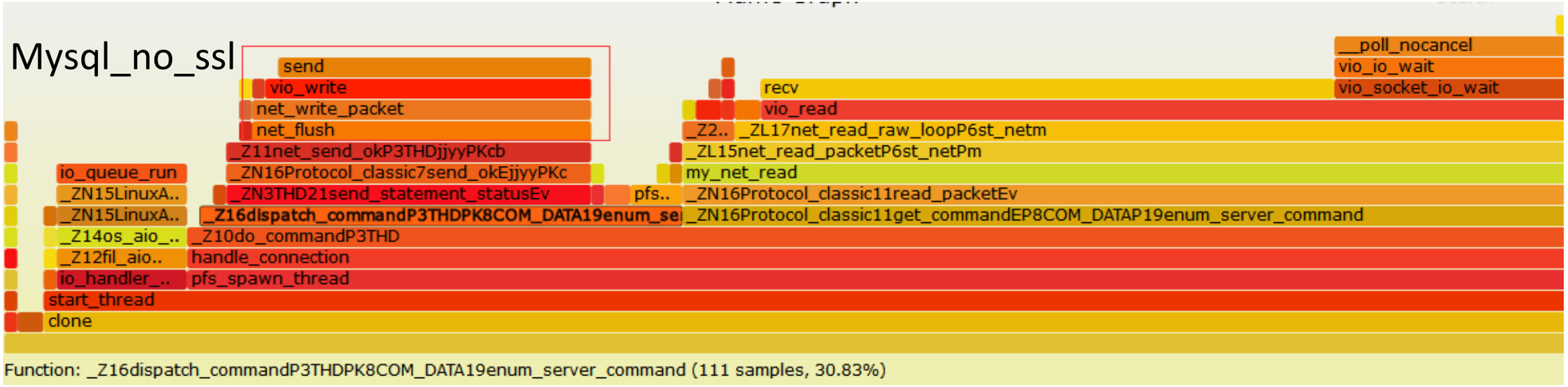


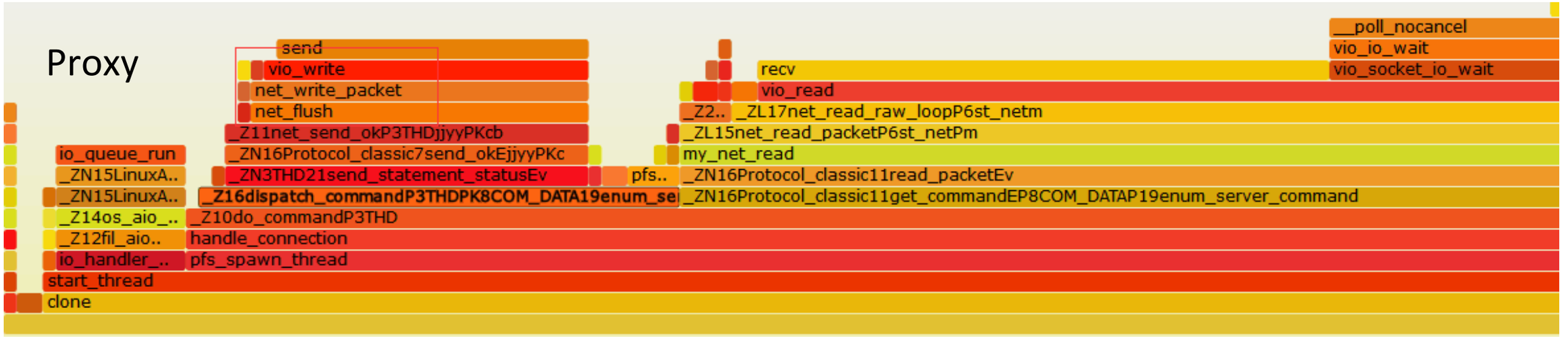
场景2：中间件性能损失测试





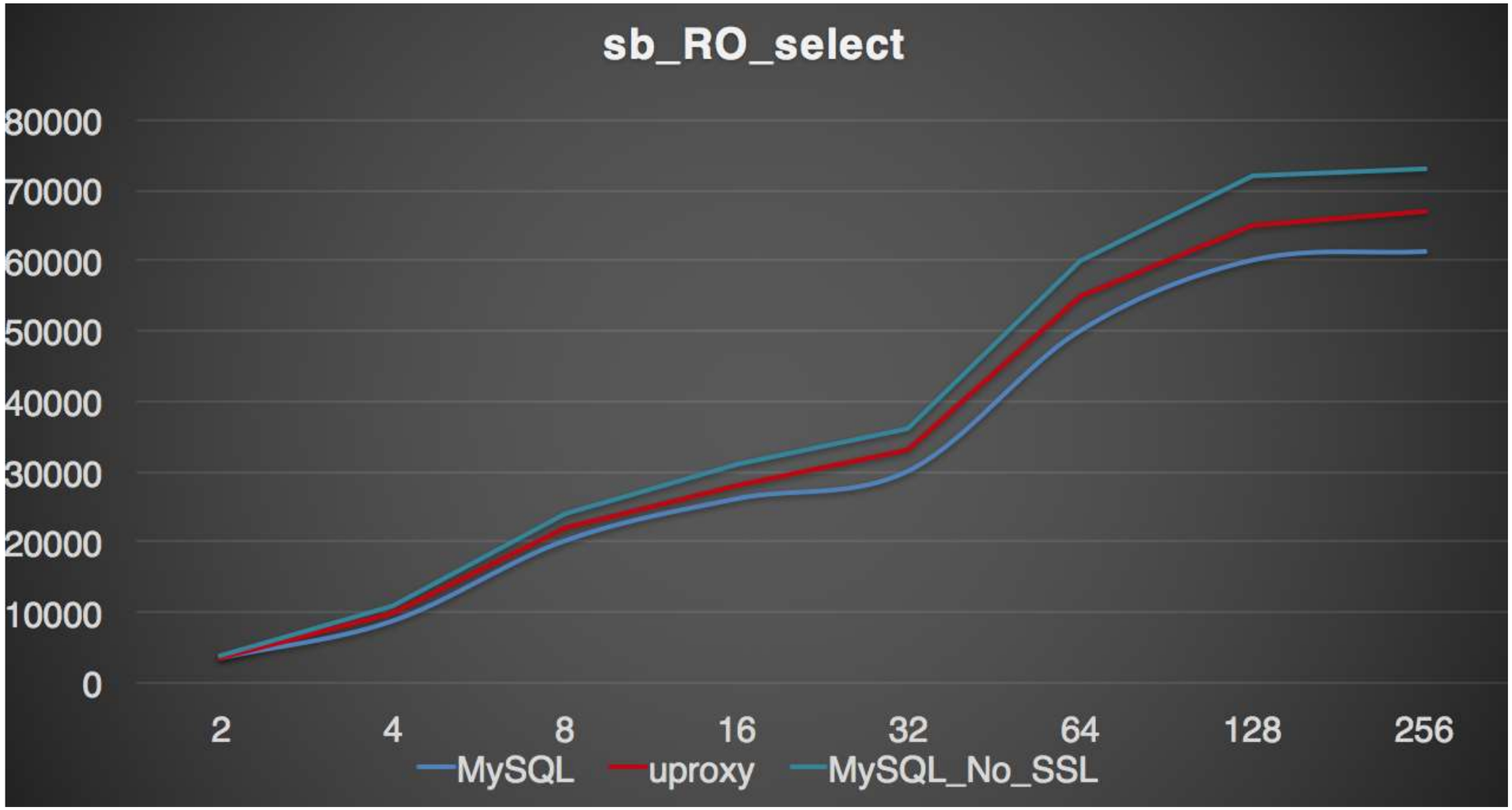
场景2：中间件性能损失测试





场景2：中间件性能损失测试

单实例5.7 VS 中间件+单实例5.7



4.0 为神马参数也不一致

MySQL配置参数和生效参数不一致

背景

客户端报错：

```
Can't open file: '.\test\mytable.frm' (errno: 24)
shell> perror 24
OS error code 24: Too many open files
```

mysql的配置文件是这么写的

```
open_files_limit =4000
max_connections = 500
table_open_cache = 1000
```

mysql动态生效的值是这样子的

```
open_files_limit = 1500
max_connections = 500
table_open_cache = 495
```

系统 ulimit -n是这样子设的

```
1500
```

MySQL配置参数和生效参数不一致

MySQL调整参数的方式

1. 根据配置(三个参数的配置值或默认值)计算
   Request_open_files(需要的文件描述符)
2. 获取有效的系统的限制值
   effective_open_files
3. 根据effective_open_files调整
   request_open_files
4. 根据调整后的request_open_files,
   计算实际生效的参数值(show variables)

MySQL配置参数和生效参数不一致

MySQL调整参数的方式

1. 根据配置(三个参数的配置值或默认值)计算 request_open_files(需要的文件描述符)

```
//mysql
  max_connections = 500
  table_open_cache = 1000
  open_files_limit=4000
//ulimit -n
  1500
```

5000！！！

request_open_files =max {

```
limit_1= max_connections + table_cache_size * 2 + 10;   2510
limit_2= max_connections * 5;                           2500
limit_3= open_files_limit ? open_files_limit : 5000;    5000
```

MySQL配置参数和生效参数不一致

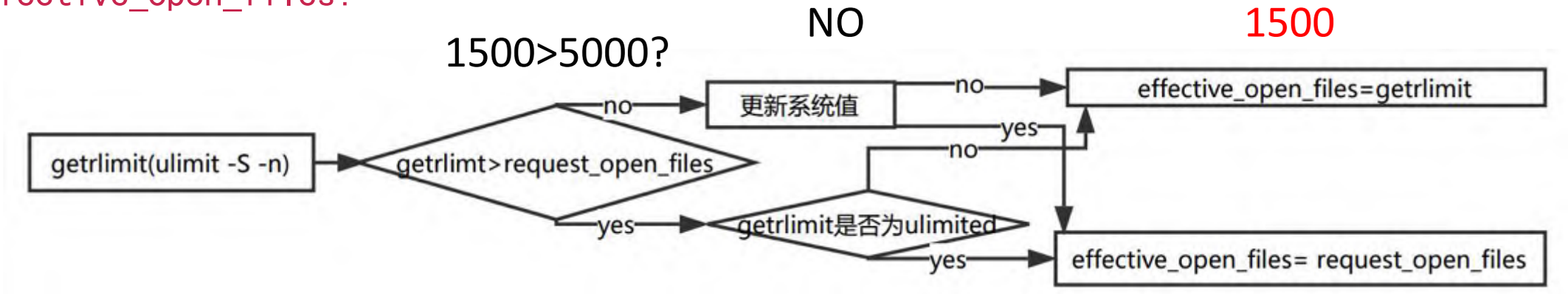
MySQL调整参数的方式

2. 获取有效的系统的限制值effective_open_files

```
//mysql
  max_connections = 500
  table_open_cache = 1000
  open_files_limit=4000

//ulimit -n
  1500
```

effective_open_files:



MySQL配置参数和生效参数不一致

MySQL调整参数的方式

3. 根据effective_open_files
   调整request_open_files

```
//mysql
  max_connections = 500
  table_open_cache = 1000
  open_files_limit=4000

//ulimit -n
  1500
```

1500 5000

requested_open_files= min(effective_open_files, request_open_files)

结论1:open_files_limit=1500

MySQL配置参数和生效参数不一致

MySQL调整参数的方式

4. 根据调整后的request_open_files，

   计算实际生效的参数值(show variables 可查看参数值)

```
//mysql
  max_connections = 500
  table_open_cache = 1000
  open_files_limit=4000
//ulimit -n
  1500
```

max_connections 默认值为400

Min（limit = requested_open_files - 10 - TABLE_OPEN_CACHE_MIN * 2 , max_connections）

1500-10-800=690 500

table_cache_size

Min（max(TABLE_OPEN_CACHE_MIN,(requested_open_files - 10 - max_connections)/2), table_cache_size）

400 (1500-10-500)/2=495 1000

结论： max_connections =500
table_open_cache=495

MySQL配置参数和生效参数不一致

mysql的配置文件是这么写的

```
open_files_limit =4000
max_connections = 500
table_open_cache = 1000
```

mysql动态生效的值是这样子的

```
open_files_limit = 1500
max_connections = 500
table_open_cache = 495
```

结论1:
open_files_limit=1500

结论2：
max_connections =500
table_open_cache=495

总结：

ulimit -S -n有关
生产环境建议尽量放开ulimit
管理平台/RDS/私有云多实例部署：需要精打细算

TECHNOLOGY
ACTION
爱可生

人生

最苦恼的事儿

莫过于

想甩锅遭遇【不可能】

爱可生·云树®

您身边的数据库贴身管家!

专注扛锅10多年!

影响主从不一致的另类参数

2亿行单表的变更问题的探讨

3种metadata lock场景分析

另一种观测视角，火焰图

slow log配置的正确姿势

动静态参数不一致源码解释

复制延迟问题分析总结

云树®·DMP—数据库集群管理平台

快速架构 | 诊断报告 | 自主优化 | 安全审计 | 全生命周期管理

自动化运维平台

高可用 | 安装部署 | 实例接管 | 读写分离 | 分库分表 | 备份恢复 | 配置管理 | 权限管理 | 资源隔离 | 灾备管理 | SQL上线 | WEB终端 | 监控告警 | 审计中心 | 日志分析

ZHDBA.COM
中华数据库行业协会

2017
中华数据库与运维大会
THANK YOU！

中华数据库与运维大会技术交流群

ZHDBA.com
中华数据库行业协会