


囧...执行analyze table意外导致waiting for table flush

老叶茶馆

+ 关注

发布于 2021-02-23 15:05:36 👁 837

文章被收录于专栏： [MySQL修行 | 老叶茶馆](#)

 运行总次数：1

代码可运行

关联问题

↻ 换一批 | ✕

❓ 执行analyze table时遇到waiting for table flush的原因是什么？

❓ 如何解决analyze table导致的waiting for table flush问题？

❓ waiting for table flush状态是什么意思？

作者：八怪（高鹏） 中亦科技 [数据库](#) 专家

一、问题描述

问题现场如下：

javascript | 运行次数：0

▶ [AI代码解释](#)

```
1  mysql> show processlist;
2  +-----+-----+-----+-----+-----+-----+
3  | Id | User          | Host          | db   | Command | Time | State
4  +-----+-----+-----+-----+-----+-----+
5  | 5 | event_scheduler | localhost    | NULL | Daemon  | 34025 | Waiting on em
6  | 11 | root           | localhost    | NULL | Query   | 1528  | User sleep
7  | 12 | root           | localhost    | NULL | Query   | 0     | starting
8  | 23 | root           | localhost    | NULL | Query   | 33    | Waiting for t
9  | 24 | root           | localhost    | NULL | Query   | 7     | Waiting for t
10 +-----+-----+-----+-----+-----+-----+
11
```

javascript | 运行次数：0

AI代码解释

1 | 乍一看来，很是奇怪，这里没有出现我们经常遇到的 flush table/flush table with read lock

https://www.jianshu.com/p/b141585cd844

以前记录的文章中的案例2，但是其实并不一样，这里是由于analyze table语句造成的。

复现过程非常简单（必须是社区版本，我使用的8.0.21），如下：

session 1

javascript | 运行次数：0

AI代码解释

1 | mysql> select sleep(1000) from test.e01; (要有几条数据)

2 | 这条语句肯定结束不了

session 2

javascript | 运行次数：0

AI代码解释

1 | mysql> analyze table test.e01;

2 | +-----+-----+-----+-----+

3 | | Table | Op | Msg_type | Msg_text |

4 | +-----+-----+-----+-----+

5 | | test.e01 | analyze | status | OK |

6 | +-----+-----+-----+-----+

session 3

javascript | 运行次数：0

AI代码解释

1 | mysql> select * from test.e01;

2 | # 被堵塞了

此时堵塞的情形就是Waiting for table flush

二、analyze触发了什么

analyze table 除了更新我们的统计数据，实际上最后做了一个操作如下（栈）：

javascript | 运行次数：0

▶

AI代码解释

```
1  #0  TABLE_SHARE::clear_version (this=0x7ffeec00ee38) at /opt/mysql/mysql-8.
2  #1  0x0000000003680a4d in <lambda(std::unordered_map<std::__cxx11::basic_st
3      my_it=...) at /opt/mysql/mysql-8.0.21/sql/sql_base.cc:9831
4  #2  0x0000000003680d0c in tdc_remove_table (thd=0x7ffed8094550, remove_type
5      at /opt/mysql/mysql-8.0.21/sql/sql_base.cc:9844
6  #3  0x0000000003d890a9 in mysql_admin_table (thd=0x7ffed8094550, tables=0x7
7      open_for_modify=true, repair_table_use_frm=false, extra_open_options=0,
8      (int (handler::*)(handler * const, THD *, HA_CHECK_OPT *)) 0x3acc64 <h
9      need_to_acquire_shared_backup_lock=true) at /opt/mysql/mysql-8.0.21/sql
```

javascript | 运行次数：0

▶

javascript | 运行次数：0

▶

AI代码解释

```
1  | 大概看一下做了什么，如下：
```

javascript | 运行次数：0

▶

AI代码解释

```
1  |         if (share->ref_count() > 0) {
2  |
3  |             if (remove_type != TDC_RT_REMOVE_NOT_OWN_KEEP_SHARE)
4  |                 share->clear_version();
5  |             table_cache_manager.free_table(thd, remove_type, share);
6  |         }
```

这里判断了是否当前table share正在使用，如果正在使用（很显然我们这个table share是不能直接释放的，因为select一直持有它），那么将share版本的设置为0（share->clear_version()。实际上这个版本由全局变量refresh_version初始化），目的在于下次如果有使用表定义的时候需要重新打开table share。

然后释放了当前没有使用的table cache（类型TDC_RT_REMOVE_UNUSED）。

三、再次访问表堵塞

当再次访问表的时候（open_table），会去判断如下是否有老的table share存在，如果存在则需要等待释放：

javascript | 运行次数：0

▶

AI代码解释

```
1  share_found:
2      if (!(flags & MYSQL_OPEN_IGNORE_FLUSH)) {
3          if (share->has_old_version()) {
4
5              release_table_share(share);
6          ...
7          wait_result =
8              tdc_wait_for_old_version(thd, table_list->db, table_list->table_r
9              ot_ctx->get_timeout(), deadlock_weight);
```

首先如果存在判断是否存在的老版本，判断是通过table share的版本和当前全局版本refresh_version进行对比。

前面我们知道这里table share的版本已经设置为0，因此这里必然进入release_table_share环节，然后等待持有者的释放（案例窗口1的select查询），然后再次获取table share。

等待栈如下：

javascript | 运行次数：0

▶

AI代码解释

```
1  (gdb) bt
2  #0  tdc_wait_for_old_version (thd=0x7ffed8094550, db=0x7ffed802e5c0 "test",
3      at /opt/mysql/mysql-8.0.21/sql/sql_base.cc:2705
4  #1  0x000000003671d7e in open_table (thd=0x7ffed8094550, table_list=0x7ffe
5  #2  0x000000003675c36 in open_and_process_table (thd=0x7ffed8094550, lex=0
6      has_prelocking_list=false, ot_ctx=0x7fff600cf1a0) at /opt/mysql/mysql-8
7  #3  0x000000003677102 in open_tables (thd=0x7ffed8094550, start=0x7fff600c
8      at /opt/mysql/mysql-8.0.21/sql/sql_base.cc:5664
9  #4  0x000000003678993 in open_tables_for_query (thd=0x7ffed8094550, tables
```

四、Percona修复该问题

Percona在上文中已经提到问题如下：

- This happens because before the fix, `ANALYZE TABLE` worked as follows:
- 1. **Opens table statistics:** concurrent DML operations (`INSERT/UPDATE/DELETE/SELECT`) are allowed
 - 2. **Updates table statistics:** concurrent DML operations are allowed
 - 3. **Update finished**
 - 4. **Invalidates table entry in the table definition cache:** concurrent DML operations are forbidden
 - 1. What happens here is `ANALYZE TABLE` marks the currently open table share instances as invalid. This does not affect running queries: they will complete as usual. But all incoming queries will not start until they can re-open table share instance. And this will not happen until all currently running queries

complete.

5. Invalidates query cache: concurrent DML operations are forbidden

Last two operations are usually fast, but they cannot finish if another query touched either the table share instance or acquired query cache mutex. And, in its turn, it cannot allow for incoming queries to start.

一个关键的修改点如下

官方版本：

```
01031:      // Row is ended below.
01032:  }
01033:
01034:  if (table->table)
01035:  {
01036:      if (table->table->s->tmp_table)
01037:      {
01038:          /*
01039:           * If the table was not opened successfully, do not try to get
01040:           * status information. (Bug#47633)
01041:           */
01042:          if (open_for_modify && !open_error)
01043:              table->table->file->info(HA_STATUS_CONST);
01044:      }
01045:      else if (open_for_modify || fatal_error)
01046:      {
01047:          tdc_remove_table(thd, TDC_RT_REMOVE_UNUSED,
01048:                          table->db, table->table_name, FALSE);
01049:          /*
01050:           * May be something modified. Consequently, we have to
01051:           * invalidate the query cache.
01052:           */
01053:          table->table= 0; // For query cache
01054:          query_cache.invalidate(thd, table, FALSE);
01055:      }
01056:      else
01057:      {
01058:          /*
01059:           * Reset which partitions that should be processed
01060:           * if ALTER TABLE t ANALYZE/CHECK/.. PARTITION ..
01061:           * CACHE INDEX/LOAD INDEX for specified partitions
01062:           */
01063:          if (table->table->part_info &&
01064:              lex->alter_info.flags & Alter_info::ALTER_ADMIN_PARTITION)
01065:          {
01066:              set_all_part_state(table->table->part_info, PART_NORMAL);
01067:          }
```

Percona版本：


```
039:     }
040:
041:     if (table->table)
042:     {
043:         const bool skip_flush=
044:             (operator_func == &handler::ha_analyze)
045:             && (table->table->file->ha_table_flags() & HA_ONLINE_ANALYZE);
046:         if (table->table->s->tmp_table)
047:         {
048:             /*
049:              * If the table was not opened successfully, do not try to get
050:              * status information. (Bug#47633)
051:              */
052:             if (open_for_modify && !open_error)
053:                 table->table->file->info(HA_STATUS_CONST);
054:         }
055:         else if (!skip_flush && open_for_modify) || fatal_error)
056:         {
057:             tdc_remove_table(thd, TDC_RT_REMOVE_UNUSED,
058:                             table->db, table->table_name, FALSE);
059:             /*
060:              * May be something modified. Consequently, we have to
061:              * invalidate the query cache.
062:              */
063:             table->table= 0; // For query cache
064:             query_cache.invalidate(thd, table, FALSE);
065:         }
066:     }
067:     else
068:     {
069:         /*
```

如此修改后analyze不会进入tdc_remove_table函数，那么table share的版本不会设置为0。因此如果使用官方版本小心本问题。

最后说下，官方版本测试到8.0.21依旧存在这个问题，而Percona已经修复了，详情见下

<https://www.percona.com/blog/2018/03/27/analyze-table-is-no-longer-a-blocking-operation/>

全文完。

Enjoy MySQL[®] :)

https 网络安全

本文分享自 老叶茶馆 微信公众号，前往查看
如有侵权，请联系 cloudcommunity@tencent.com 删除。
本文参与 腾讯云自媒体同步曝光计划 ，欢迎热爱写作的你一起参与！

推荐阅读

编辑精选文章

- 📖 一篇文搞定消息队列选型
- 📖 腾讯写码6年，我总结的技术人核心竞...
- 📖 醍醐灌顶！异地多活架构设计看这篇就...

相关讨论

- 📖 Waiting For Connection (netbeans-x...
- 📖 android studio调试测试类时 waiting f...
- 📖 语音识别 录音文件识别 为什么一直返...

相关

- 📖
- 📖
- 📖

901 0

👁 2K 💬 0

1.9K 0

185 0

3K 0

881 0

711 0

利用Pytorch的C++前端(libtorch)读取预训练权重并进行预测

 1.2K

 0

DDCTF WriteUp (Web压轴+逆向)

 1.5K

 0

C++内存问题排查攻略

 558

 0

畅游数据库性能优化过程简析（上）

 3.6K

 0

show create table底层流程跟踪

 552

 0

【Chromium】如何提取Chromium的Base库（下）

 606

 0

LevelDB Put如何写入数据

 825

 0

Percona 8.0.30中"show engine innodb status"导致coredump排查及分析

 310

 0

show status和set gtid_mode 导致线程死锁案例

 1K


 0


【栽了】迄今为止吐槽最多的标准库函数

 559

 0

QT5在windows下调用OpenCV库出现: undefined reference to `xxxxx' 错误解决办法（适用MinGW编译器）。

 7K

 0

数据库诊断不了的，腾讯大神来“诊断”

 2.1K

 0

深入理解MySQL 5.7 GTID系列(五) gtid_executed&gtid_purged什么时候更新

 1.4K

 0