# 22 | MySQL有哪些“饮鸩止渴”提高性能的方法？

**短连接风暴**

正常的短连接就是连接到数据库后，执行很少的SQL语句就断开，下次需要的时候在重连，如果使用的是短连接，在业务高峰期的时候，可能出现连接数暴涨的情况。

在第一篇说过MySQL建立连接的过程，成本很高，除了正常的网络连接三次握手外，还需要登录权限判断和获得这个数据的读取权限。

短连接模型存在一个风险，就是一旦数据库处理的慢一些，连接数就会暴涨

Max\_connections参数，用来控制一个MySQL实例同时存在的连接数的上线，超过这个值，系统就会拒绝下来的连接请求，并报错提示“Too Many connections”。对于被拒绝连接的请求，从业务角度来说是数据库不可用。

在机器负载比较高的时候，处理现有请求的时间变长，每个连接保持的时间也更长，这是，再有新建连接的，就可能会超过max\_connections的限制。

碰到这种情况时，一个比较自然的想法，就是调高max\_connections的值，但这样的是有风险的。因为设计max\_connections这个参数的目的是保护MySQL,如果把它改的太大，让更多的连接都可以进来，那么系统的负载可能会进一步加大，大量的资源消耗在权限验证等逻辑上，结果可能适得其反，已经连接的线程拿不到CPU资源去执行业务的SQL请求。

**第一种方法：先处理掉那么占着连接但是不工作的线程**

max\_connections的计算，不是看谁在running,是只要连接着就占一个计数位置，对于那些不需要保持的连接，可以用kill connection主动踢掉，这个行为跟事先设置wait\_timeout的效果是一样的，设置wait\_timeout参数表示的是，一个线程空闲wait\_timeout这么多秒之后，就会被MySQL直接断开。

在show processlist的结果是，踢掉显示为sleep的线程，可能是有损的

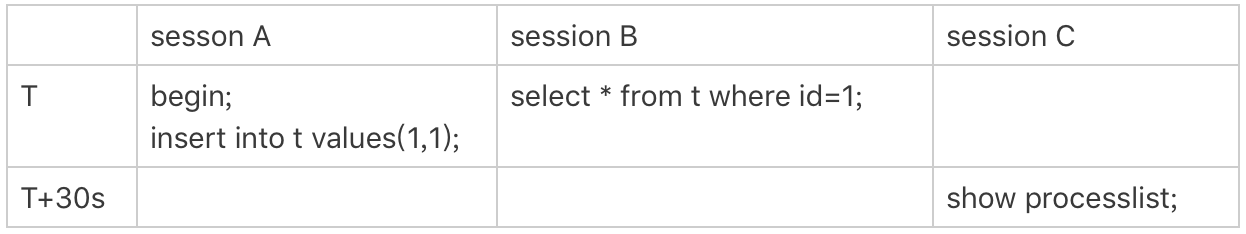


图 1 sleep线程的两种状态

在上面的例子，如果断开session A的连接，此时sessionA还没有提交，所以MySQL只能按照回滚事务来处理；而断开sessionB的连接，就没什么大影响，所以按照优先级来说，应该先断开像sessionB这样的事务外空闲的连接

怎么判断哪些是事务外空闲的呢？Session C在T时刻后30秒执行show processlist,结果如图

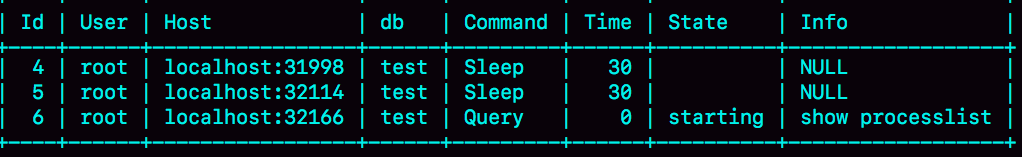


图 2 sleep 线程的两种状态，show processlist结果

图中id=4 和id=5的两个会话都是sleep状态，而要看事务具体状态的话，查information\_schema库的innodb\_trx表

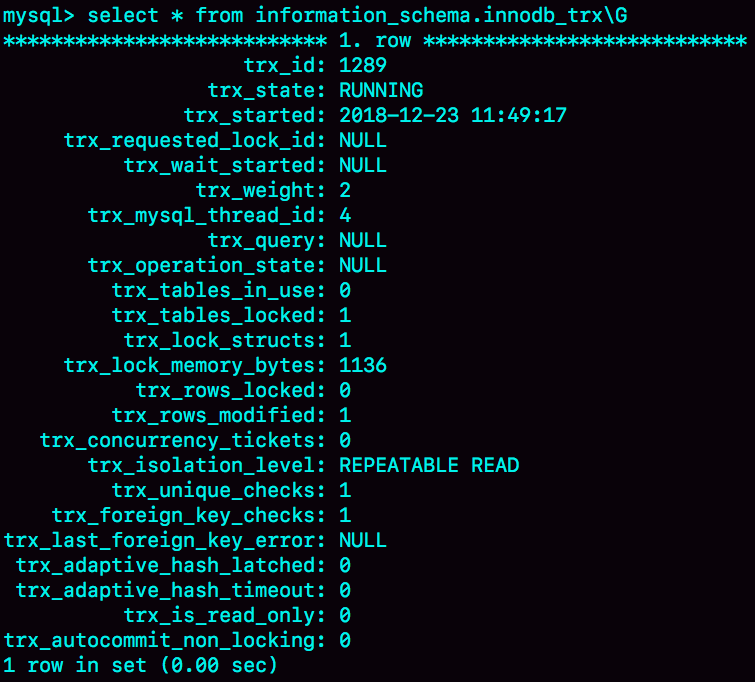


图3 information\_schema.innodb\_trx查询事务状态

这个结果trx\_mysql\_thread\_id=4，表示id=4的线程还在事务中

因此，如果是连接数过多，可以优先断开事务外空闲太久的连接；如果这样不够，在考虑断开事务内空闲太久的连接

从服务器断开连接使用的是kill\_connection + id的命令，一个客户端处于sleep状态时，它的连接被服务器主动断开，这客户端不会马上知道，直到客户端在发下一个请求的时候，才会收到这样的报错.

从数据端主动断开连接可能是有损的，尤其是有的应用端收到这个错误后，不重新连接，而是直接用这个已经不能用的句柄重试查询，这会导致从应用端看上去，“MySQL一直没有恢复”

**第二种方法：减少连接过程的消耗**

跳过权限验证的方法是：重启数据库，并使用--skip-grant-tables参数启动，这样整个MySQL会跳过所有的权限验证阶段，包括连接过程和执行过程在内。

这种方法风险极高，是不建议使用的方案

MySQL8.0版本里，如果启用--skip-grant-table参数，MySQL会默认把--skip-networking参数打开，表示这时数据库只能被本地的客户端连接，

**慢查询性能问题**

Mysql中，会引发性能问题的慢查询，大体有三种可能

1. 索引没设计好
2. SLQ语句没写好
3. MySQL选错索引

**导致慢查询的第一种可能，索引没设计好**

一般是通过紧急创建索引来解决，MySQL版本以后，创建索引都支持Online DDL了，对于那种高峰期数据库已经被这个语句打挂了，最高效的做法都是直接执行alter table语句。

比较理想的是能够在备库先执行，假设现在的服务是一主一备，主库A,备库B

1.在备库B上执行set sql\_log\_bin=off，也就是不写binlog,然后执行alter table语句加上索引

2.执行主备切换

3.这个时候主库是B，备库是A,在A上执行set sql\_log\_bin=off然后执行alter table语句加上索引

**导致慢查询的第二种可能，语句没写好**

比如，犯了第18篇中提到的错误，导致语句没有用上索引，可以通过改写SQL语句来处理，MySQL5.7提供了query\_rewrite功能，可以把输入的一种语句改写成另外以总模式

比如：语句被错误的写成select \* from t where id + 1= 10000, 可以增加一个改写规则

mysql> insert into query\_rewrite.rewrite\_rules(pattern, replacement, pattern\_database) values ("select \* from t where id + 1 = ?", "select \* from t where id = ? - 1", "db1");

call query\_rewrite.flush\_rewrite\_rules();

这里call query\_rewirte.flush\_rewrite\_rules()这个存储过程，是让插入的新规则生效，也就是我们说的“查询重写”

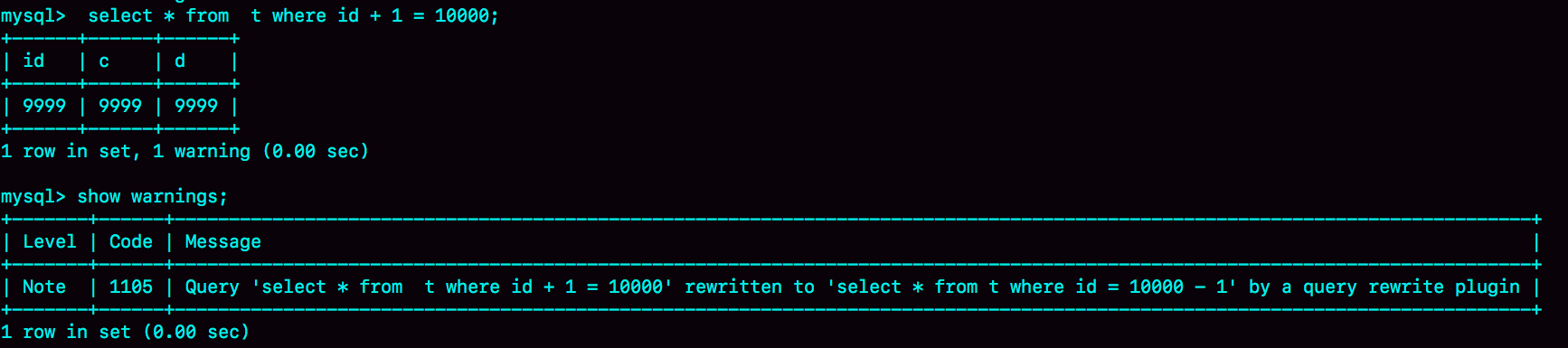


图4 查询重写效果

**导致慢查询的第三种可能是第10篇的Mysql 选错索引**

此时应急方案就是给这个语句加上force index;

同样的，使用查询重写功能，给原来的语句加上force index，也可以解决问题

可以避免上述的过程

1.上线前，在测试环境，把慢查询日志（slow log） 打开，并且把long\_query\_time设置0

确保每个语句都会被记录如慢查询日志

1. 在测试表里插入模拟线上的数据，走一遍回归测试
2. 观察慢日志里没类语句的输出，特别留意Rows\_examined字段是否与预期一致

**QPS突增问题**