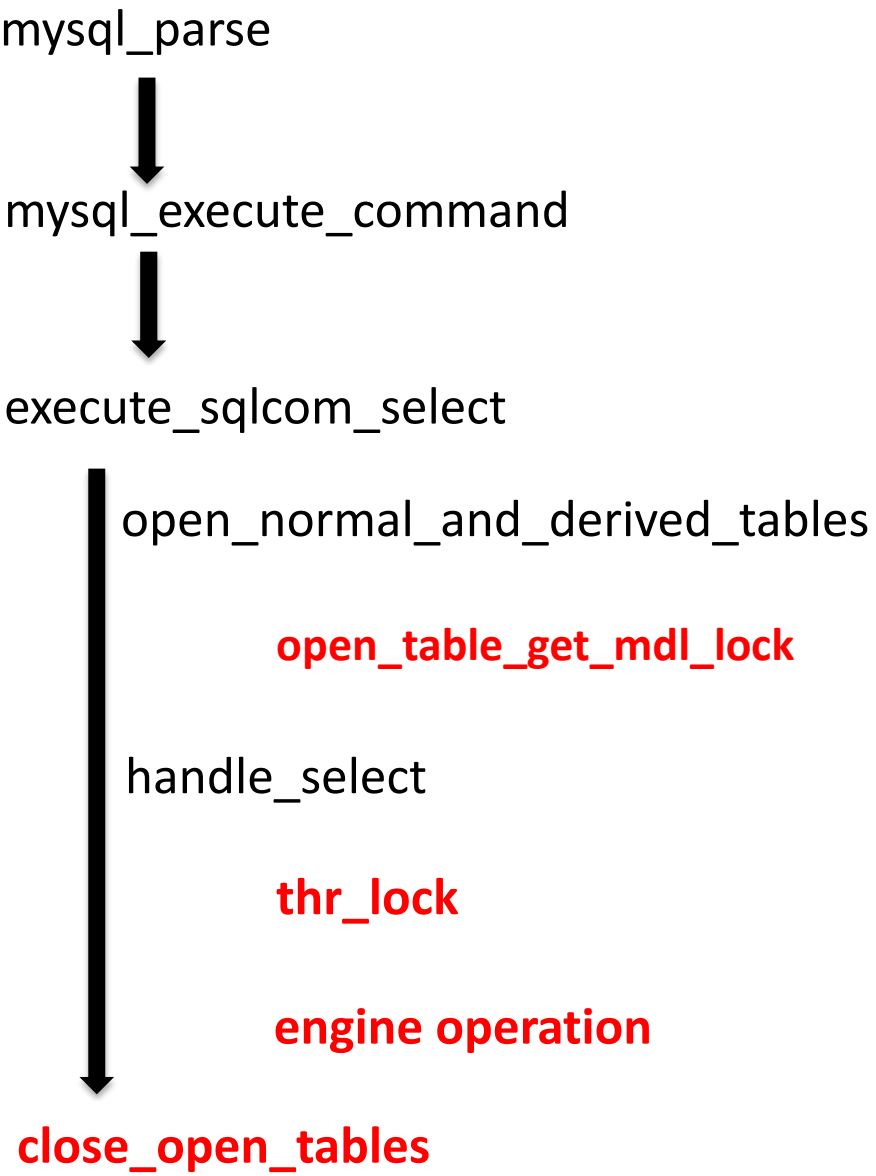
Gdevops

全球敏捷运维峰会

MySQL大并发量性能优化实战

演讲人：张青林

简单查询语句的执行过程



PS 的影响范围

Agenda

- ✓ metadata_locks_hash_instances 的作用 & 原理 & 实现
- ✓ table_open_cache_instances 的作用 & 原理 & 实现
- ✓ performance_schmea 的作用 & 原理 & 实现
- ✓ INNODB MVCC 的实现

MySQL metadata_locks_hash_instances

✓ 什么是 MDL Lock，其主要作用是什么？

MDL Lock (Metadata Lock)，即 Server 层的数据字典锁，是为了保护 DB，table, trigger, procedure 等数据对象而设计的，先看一下一个 bug#989：

```
connect (master,127.0.0.1,root,,test,$MASTER_MYPORT,);
connect (master1,127.0.0.1,root,,test,$MASTER_MYPORT,);

connection master;
drop table if exists t1,t2,t3;

create table t1 (a int) type=innodb;
reset master;
begin;
insert into t1 values(1);

connection master1;

drop table t1;

connection master;

commit;
let $VERSION=`select version()`;
--replace_result $VERSION VERSION
show binlog events;
```

MDL Lock 的实现原理

mysql_execute_command

...

acquire_lock(...mdl_requests...)

try_acquire_lock_impl

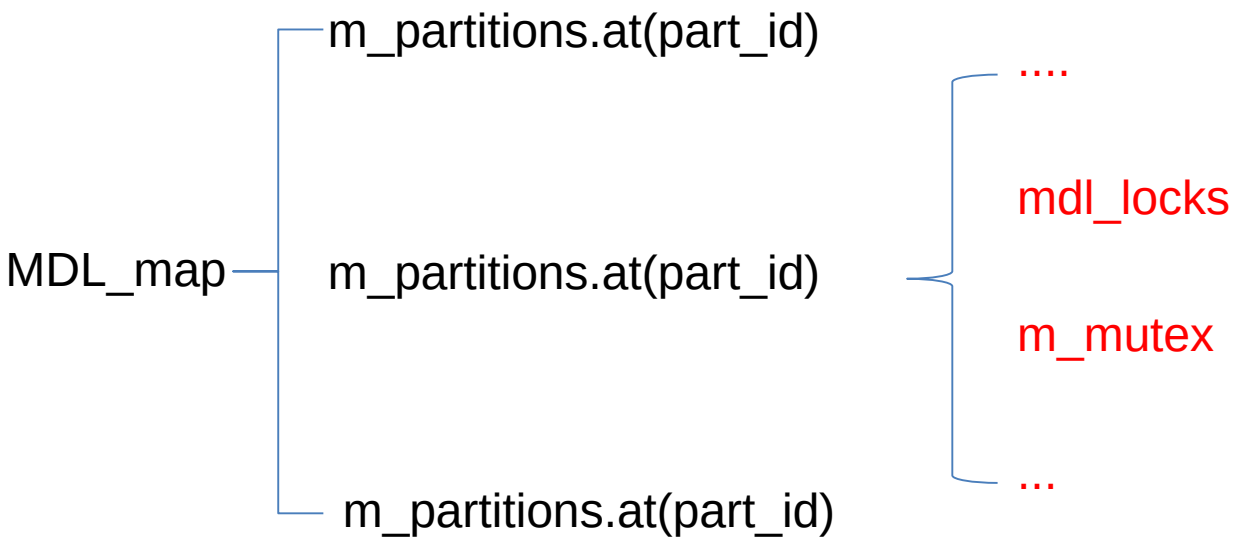
find_or_insert

MDL_map_partition::find_or_insert

MDL_lock::create

....

THD -> MDL_context -> MDL_tickets -> MDL_lock



MDL_key {mdl_namespace, dbname, object_name}

问题：MDL Lock 中存在的瓶颈，MDL Lock 瓶颈的缓解方法， MDL Lock 中在各个版本中的实现方法

MySQL table_open_cache_instances

- ✓ 什么是 table_cache，为什么需要 table_cache ?

table_cache是缓存 table 对象的容器，table 是MySQL中用户线程操作表对象的内存对象，table 中包括了用户的相关操作记录，对应的 engine 的文件句柄以及内部对象管理的控制信息等；

- ✓ MySQL 打开表 & 关闭表的过程

- ✓ MySQL 打开表的流程中存在的性能瓶颈

当 MySQL 的并发链接数较高的时候，会有多个用户线程被分配到同一个 table_cache 中，造成多个线程竞争 table_cache->m_lock 的问题，引起 thread_running 飙升；

- ✓ 解决 table_cache 问题的方法

对参数 table_open_cache_instances 进行调整，1 -> 64

- ✓ table_open_cache_instances & table_definition_cache 关系

每一个 table 内存对像中会有一个指象表对像的指针，即 table_share, table_share 也有一个缓存大小的设置，即 table_definition_cache， 同时， table_definition_cache 也是 innodb 层缓存数据字典的设置，如果没有对 table_definition_cache 进行设置，则会根据 table_open_cache的值进行计算；

- ✓ table cache 相关问题的状态信息

```
show status like '%table_cache%';   flush tables;
```

table 对象的生命周期

open_and_process_table

open_table

...

tc= table_cache_manager.get_cache(thd)

tc->lock()

table= tc->get_table()

tc->unlock

...

close_thread_table

release_table

link_unused_table

...

el->used_tables.remove(table);

el->free_tables.push_front(table);

link_unused_table(table);

free_unused_tables_if_necessary

...

- ✓ cache_manager & cache_instance 控制的是 thd 使用表的分区信息；
- ✓ Table_cache_element table_cache 共同管理 table 缓存对象；

MySQL performance_schema

什么是 performance_schema，和 information_schmea 有什么区别？

performance schema 是 MySQL 的内部诊断器，用于记录 MySQL 在运行期间的各种信息，如表锁情况、mutex 竞争情况、执行语句的情况等,和 Information Schema 类似的是拥用的信息都是内存信息，而不是存在磁盘上的。和 information_schema 有以下不同点：

- information_schema 中的信息都是从 MySQL 的内存对象中读出来的，只有在需要的时候才会读取这些信息，如 processlist, profile, innodb_trx 等，不需要额外的存储或者函数调用，而 performance schema 则是通过一系列的回调函数来将这些信息额外的存储起来，使用的时候进行显示，因此 performance schema 消耗更多的 CPU & memory 资源；
- Information_schema 中的表是没有表定义文件的，表结构是在内存中写死的，而 performation_schema 中的表是有表定义文件的；
- 实现方式不同，Information_schema 只是对内存的 traverse copy, 而 performance_schema 则使用固定的接口来进行实现；
- 作用不同，Information_schema 主要是轻量级的使用，即使在使用的时候也很少会有性能影响，performance_schema 则是 MySQL 的内部监控系统，可以很好的定位性能问题，但也很影响性能；

Performance schema 的作用

- 对各种资源监控开关进行设置
- 监控mutex, rw_lock, conds等资源的使用情况
- 监控内存使用的情况 (5.7)
- 监控SQL语句的使用情况并对相关的语句进行统计
- 监控内部运行的线程
- 监控文件资源的使用情况并对相关的资源进行统计

Performance schema 实现过程及问题

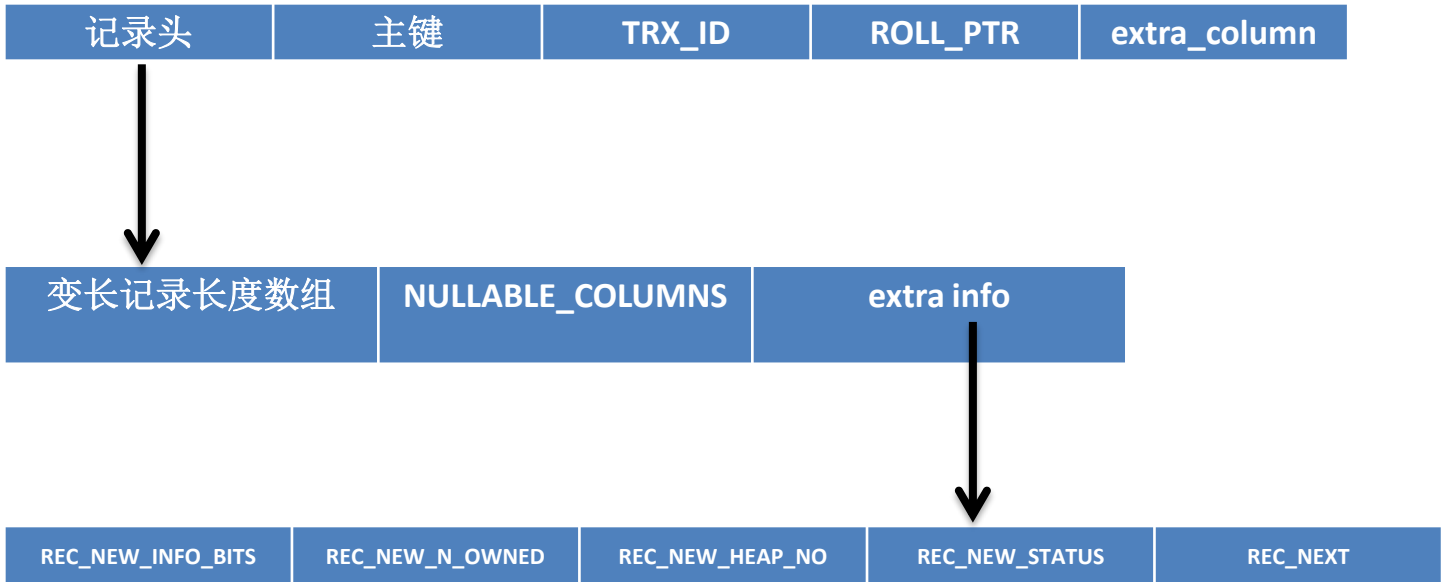
- 创建对应的监控表
- 实现 performance_schema 引擎的相关接口；
- 在相应的地方设置回调函数，在相应的操作中添加或减少对应的监控信息；
- 查询的时候调用相应的接口访问已经额外存储的数据；
- 由于需要存储一份额外的数据在内存中，所以会有额外的内存消耗；
- Perfornamce schema 会加重计算代价，对性能也有一定的影响；

MySQL MVCC 的实现

- ✓ MySQL Innodb record 物理结构
- ✓ MySQL Innodb MVCC 实现原理
- ✓ MySQL Innodb MVCC 实现过程中存在的问题
- ✓ MySQL Innodb MVCC 对于问题的解决过程
- ✓ TDB5.6 Innodb MVCC 解决问题的方案设计

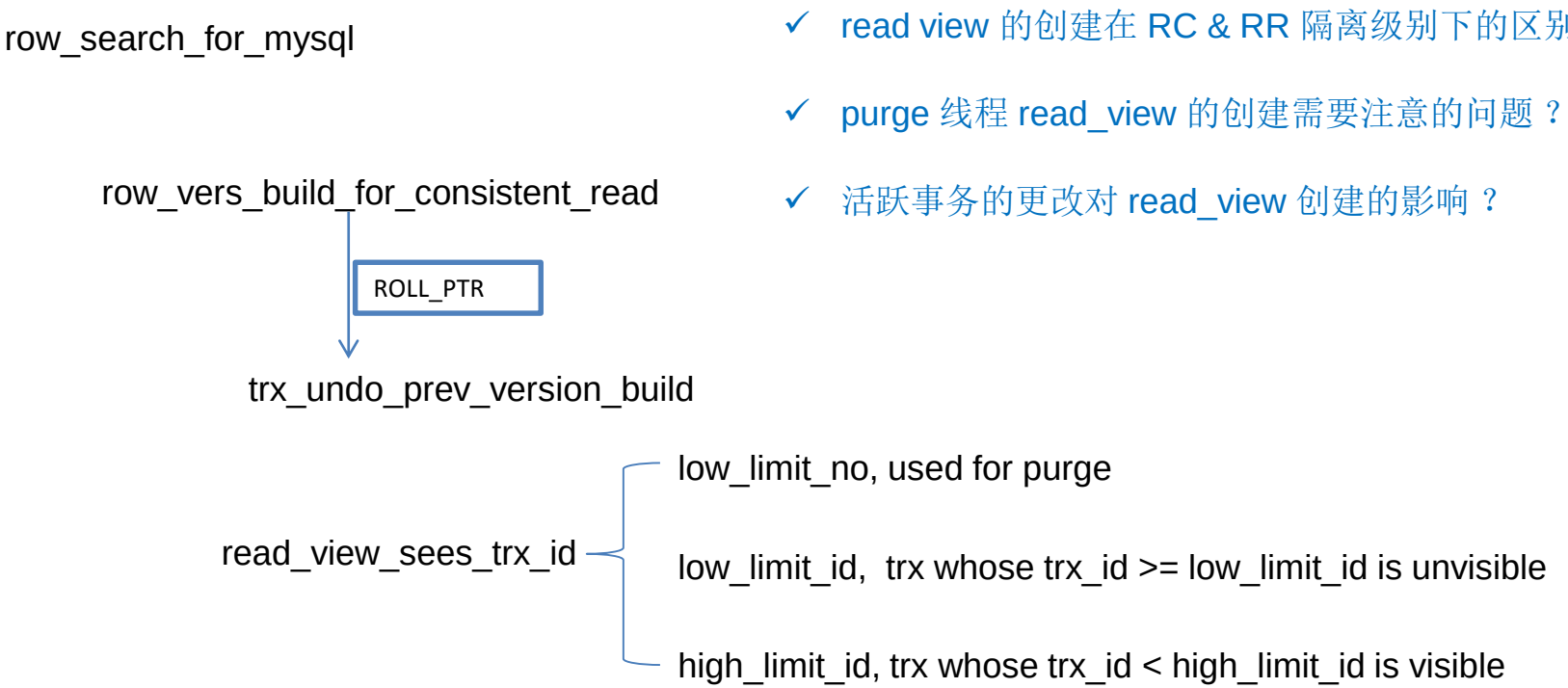
MySQL Innodb record 物理结构

Record 格式：



MySQL Innodb MVCC 实现原理

MVCC 部分调用栈：



- ✓ read view 的创建在 RC & RR 隔离级别下的区别？
- ✓ purge 线程 read_view 的创建需要注意的问题？
- ✓ 活跃事务的更改对 read_view 创建的影响？

MySQL Innodb MVCC 存在的问题

参考视图创建函数 read_view_open_now_low，可以看到：

- ✓ 整个创建过程一直持有 trx_sys->mutex 锁；
- ✓ 需要遍历 trx_sys->trx_list (5.5) 或 trx_sys->rw_list (5.6)；
- ✓ read_view 的内存在每次创建中被分配，事务提交后被释放；
- ✓ 并发较大，活跃事务链表过长时，会在 trx_sys->mutex 上有较大的消耗；

详情见：https://bugs.mysql.com/bug.php?id=49169

From rem0rec.ic

MySQL Innodb MVCC 对于问题的解决过程

解决过程比较区折，但主要是从几个方面来进行优化，从5.6 ->5.7 的 release note & work log 也可以清晰的看到，简单介绍如下：

- ✓ 减少 read_view 创建时的事务链表的长度（语法上支持或设置 session 变量）
- ✓ 减少 read_view 创建时的不必要的内存分配
- ✓ 减少遍历 rw_trx_list 链表的次数
- ✓ 减少 read_view 的分配 & 释放

MySQL Innodb MVCC 对于问题的解决过程

MySQL 5.7 的具体细节可以参考：(5.7 中相关的 merge 会更清晰)

- ✓ WL#6047： http://dev.mysql.com/worklog/task/?id=6047
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/5209

- ✓ WL#6578： http://dev.mysql.com/worklog/task/?id=6578 （部分类似于 Percona 5.6）
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/6203
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/6204
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/6205
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/6224
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/6236
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/6788

- ✓ WL#6906： http://dev.mysql.com/worklog/task/?id=6906 (trx 的 缓冲池机制，效果不大)
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/5744
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/5750
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/5753
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/5756
  http://bazaar.launchpad.net/~mysql/mysql-server/5.7/revision/5786

CDB 5.6 Innodb MVCC 解决问题的方案设计

结合 percona & MySQL 5.7 两种方案进行实现，主要思想如下：

- ✓ backport percona 5.6 中对 bug#49169 的修复；
- ✓ 实现 MySQL 5.7 中移除只读链表的想法；
- ✓ 其它在编码 & 测试过程中待改近的地方

经过以上的优化，对于只读性能应该可以有比较大的提升，目前进度在移除只读链表中

Gdevops

全球敏捷运维峰会

THANK YOU！