首页推荐

MySQL锁定位实践指南

原创 Digital Observer All China Database Union 2025-06-29 613

作者：Digital Observer（施嘉伟）
Oracle **ACE Pro**: Database
PostgreSQL **ACE** Partner
11年数据库行业经验，现主要从事数据库服务工作
拥有Oracle **OCM**、DB2 10.1 Fundamentals、MySQL 8.0 OCP、WebLogic 12c OCA、KCP、PCTP、PCSD、**PGCM**、OCI、PolarDB技术专家、达梦师资认证、数据安全咨询高级等认证
ITPUB认证专家、PolarDB开源社区技术顾问、HaloDB技术顾问、TiDB社区技术布道师、青学会MOP技术社区专家顾问、国内某高校企业实践指导教师
公众号：Digital Observer；CSDN：施嘉伟；ITPUB：sjw1933；墨天轮：Digital Observer；PGFans：施嘉伟。

MySQL锁定位实践指南

在数据库日常运维中，锁问题常常成为性能瓶颈和系统卡顿的根源。本文系统总结了MySQL中常见的锁类型及其排查方法，涵盖全局读锁、表锁、元数据锁（MDL）及行锁，并提供标准化的诊断脚本，适用于MySQL 5.6、5.7与8.0多个版本。

一、全局读锁（Global Read Lock）

全局读锁通常由 `FLUSH TABLES WITH READ LOCK` 添加，常用于逻辑备份或主从切换。另一种风险情形则是权限设置不合理，具备 `RELOAD` 权限的账号可能会误操作导致加锁。

排查方法

MySQL 5.7 起支持通过 `performance_schema.metadata_locks` 表查看 Server 层级锁信息，包括全局读锁。

查询示例（未开启 performance_schema）

```
select /* default performance_schema level*/
concat('kill ',l.id,';') as kill_command,
e.THREAD_ID,
e.event_name,
e.CURRENT_SCHEMA,
e.SQL_TEXT,
round(e.TIMER_WAIT / 1000 / 1000 / 1000 /1000,2) as "TIMER_WAIT(s)",
l.host,
l.db,
l.state,
DATE_SUB(NOW(), INTERVAL(SELECT VARIABLE_VALUE FROM performance_schema.global_status WHERE
from performance_schema.events_statements_history e inner join information_schema.PROCESSLI
on e.THREAD_ID = sys.ps_thread_id(l.id)
and e.event_name = 'statement/sql/flush'
order by e.TIMER_START;
```

查询示例（开启 performance_schema）

Digital Observer
LV.3
关注
122 文章
90 粉丝
38K+ 浏览量
获得了 232 次点赞
内容获得 50 次评论
获得了 427 次收藏

TA的专栏
RMAN备份恢复实战
收录 9 篇内容
PostgreSQL
收录 11 篇内容

热门文章
Oracle 11g升级到19c问题汇总 2024-10-11 1859浏览
免费获得 Oracle MOS 账号的秘密，技术爱好者必看！ 2025-05-28 1227浏览
Oracle数据库 Truncate慢分析 2024-09-09 1028浏览
Oracle DB replay实践 2024-11-14 869浏览
Oracle RAC单节点高负载问题诊断与解决记录 2024-12-17 821浏览

在线实训环境入口
MySQL在线实训环境
查看详情 »

最新文章
与金仓结缘：从 KCP 到 KCM 的成长之路 6天前 20浏览
达梦数据库备份与恢复：dexp 和 dimp 工具的使用与优化 2025-07-30 144浏览
SQL Server安全配置全面检查与优化方案

```
select * from performance_schema.metadata_locks where owner_thread_id != sys.ps_thread_id(c
```

二、表锁（Table Lock）

表锁通常由显式语句如 `LOCK TABLE t READ` 加入，用于控制表级别的并发访问。

排查方法

```
select * from performance_schema.metadata_locks where owner_thread_id != sys.ps_thread_id(c
```

三、MDL锁（Metadata Lock）

MDL（元数据锁）在访问表对象时自动加锁，用于保证读写操作的元数据一致性。若遇到 DDL 阻塞等情况，往往与此类锁有关。

排查方法

未开启 sys 扩展（适用于 MySQL 5.7/8.0）

```
use performance_schema;
select /* 默认 performance_schema级别*/
p.THREAD_ID,
concat('kill ',l.id,';') as kill_command,
p.event_name,
p.TIMER_START,
round(p.TIMER_WAIT / 1000 / 1000 / 1000 /1000,2) as "TIMER_WAIT(s)",
p.CURRENT_SCHEMA,
p.SQL_TEXT,
l.host,
l.DB,
l.STATE,
l.INFO as "mdl_blocking_info",
DATE_SUB(NOW(), INTERVAL(SELECT VARIABLE_VALUE FROM performance_schema.global_status WHERE
from performance_schema.events_statements_history p
         inner join information_schema.PROCESSLIST l
         on p.THREAD_ID = sys.ps_thread_id(l.id)
         and l.state = 'Waiting for table metadata lock'
order by p.TIMER_START;
```

开启 sys 扩展时

```
select * from sys.schema_table_lock_waits;
```

四、行锁（Row Lock）

行锁是InnoDB的核心特性之一，支持高并发访问。常见的行锁类型包括意向锁、Next-Key锁、间隙锁等，表现为以下几类：

- `IX`：意向排他锁（表级）
- `X`：Next-Key锁（锁定记录本身及前间隙，排他）
- `S`：Next-Key共享锁
- `X,REC_NOT_GAP`：锁定记录本身（排他）
- `S,REC_NOT_GAP`：锁定记录本身（共享）

2025-07-30 100浏览
Oracle 常见的33个等待事件
2025-07-29 332浏览
浅谈信创数据库改造重难点
2025-07-29 105浏览

目录

- `X,GAP` / `S,GAP`：纯间隙锁
- `X,GAP,INSERT_INTENTION`：插入意向锁

排查方法

MySQL 5.7 / 8.0（未启用 sys 扩展）

```
-- 默认开启performance_schema的情况，5.7和8.0都能用。
select *
from (
         select distinct c.THREAD_ID,
                         x.sql_kill_blocking_connection
                         x.blocking_lock_mode,
                         x.waiting_lock_mode,
                         c.event_name,
                         c.sql_text
                         x.waiting_query
                         c.CURRENT_SCHEMA,
                         c.OBJECT_NAME,
                         DATE_SUB(NOW(), INTERVAL (SELECT VARIABLE_VALUE
                                                   FROM performance_schema.global_status
                                                   WHERE VARIABLE_NAME = 'UPTIME') -
                                                  c.TIMER_START / 1000 / 1000 / 1000 / 1000
                         x.wait_age_secs
                         x.locked_table,
                         x.locked_index,
                         x.locked_type
from performance_schema.events_statements_history c
         inner join (
    select t.THREAD_ID,
           ilw.sql_kill_blocking_connection,
                         ilw.waiting_lock_mode,
                         ilw.blocking_lock_mode,
                         ilw.wait_age_secs,
                         ilw.locked_table,
                         ilw.waiting_query,
                         ilw.locked_index,
                         ilw.locked_type
             from sys.innodb_lock_waits ilw
                      inner join performance_schema.threads t on t.PROCESSLIST_ID = ilw.blc
                             on x.THREAD_ID = c.THREAD_ID) xx
order by xx.blocking_session_sql_start_time;
```

MySQL 5.6（未启用 performance_schema）

```
SELECT r.trx_wait_started as wait_started,
TIMEDIFF(now(), r.trx_wait_started) as wait_age,
TIMESTAMPDIFF(SECOND，r.trx_wait_started,Now() AS wait_age_secs,
rl.lock_table as locked_table,
r1.lock_index as locked_index,
r1.lock_type As locked_type,
r.trx_id as waiting_trx_id,
r.trx_started as waiting_trx_started,
TIMEDIFF(NOW(), r.trx_started) as waiting_trx_age,
r.trx_rows_locked as waiting_trx_rows_locked,
r.trx_rows_modified as waiting_trx_rows_modified,
r.trx_mysql_thread_id as waiting_pid,
sys.format_statement(r.trx_query) As waiting_query,
r1.lock_id As waiting_lock_id,
r1.lock_mode as waiting_lock_mode,
b.trx_id as blocking_trx_id,
b.trx_mysql_thread_id as blocking_pid,
sys.format_statement(b.trx_query) as blocking_query,
bl.lock_id as blocking_lock_id,
bl.lock_mode As blocking_lock_mode,
b.trx_started As blocking_trx_started,
TIMEDIFF(NOW(),b.trx_started) as blocking_trx_age,
b.trx_rows_locked as blocking_trx_rows_locked,
b.trx_rows_modified as blocking_trx_rows_modified,
concat('KILL QUERY', b.trx_mysql_thread_id) as sql_kill_blocking_query,
concat('KILL ', b.trx_mysql_thread_id) as sql_kill_blocking_connection
 from information_schema.innodb_lock_waits w
 inner join information_schema.innodb_trx b on b.trx_id = w.blocking_trx_id
 inner join information_schema.innodb_trx r on r.trx_id = w.requesting_trx_id
 inner join information_schema.innodb_locks bl on bl.lock_id = w.blocking_lock_id
 inner join information_schema.innodb_locks rl on rl.lock_id = w.requested_lock_id
order by r.trx_wait_started;
```

通过上述脚本与方法，可以对MySQL不同层级的锁进行精准排查与定位。建议在生产环境中提前配置好 `performance_schema` 和 `sys` 库，以提升锁问题的可观测性和处理效率，保障数据库系统的稳定运行。

墨力计划 mysql mysql数据库 锁

最后修改时间：2025-07-07 19:59:08

「喜欢这篇文章，您的关注和赞赏是给作者最好的鼓励」

关注作者 赞赏

评论

分享你的看法，一起交流吧～

相关阅读

ACDU周度精选 | 本周数据库圈热点 + 技术干货分享（2025/7/25期）
墨天轮小助手 468次阅读 2025-07-25 15:54:18

ACDU周度精选 | 本周数据库圈热点 + 技术干货分享（2025/7/17期）
墨天轮小助手 435次阅读 2025-07-17 15:31:18

墨天轮「实操看我的」数据库主题征文活动启动
墨天轮编辑部 379次阅读 2025-07-22 16:11:27

深度解析MySQL的半连接转换
听见风的声音 204次阅读 2025-07-14 10:23:00

MySQL 9.4.0 正式发布，支持 RHEL 10 和 Oracle Linux 10
严少安 199次阅读 2025-07-23 01:21:32

索引条件下推和分区-一条SQL语句执行计划的分析
听见风的声音 196次阅读 2025-07-23 09:22:58

null和子查询--not in和not exists怎么选择？
听见风的声音 182次阅读 2025-07-21 08:54:19

MySQL数据库SQL优化案例(走错索引)
陈举超 164次阅读 2025-07-17 21:24:40

使用 MySQL Clone 插件为MGR集群添加节点
黄山谷 162次阅读 2025-07-23 22:04:19

MySQL 8.0.40：字符集革命、窗口函数效能与DDL原子性实践
shunwah 140次阅读 2025-07-15 15:27:19