

MySQL性能分析的“秘密武器”，深度剖析SQL问题

原创

szrsu

2025-01-23

680



MySQL出现性能问题，该怎样深入分析，有什么工具可以使用？本文将分享两个实用的工具，让你事半功倍！

profile工具

使用profile可以分析当前查询 SQL 语句执行的资源消耗情况，更清楚地了解SQL执行的过程。默认情况下，该参数处于关闭状态，并保存最近15次运行结果（受profiling_history_size这个参数影响）。通过这个工具，可以用来对比优化前和优化后的SQL性能。

```
--查看当前版本是否支持
mysql> select @@have_profiling;
+-----+
| @@have_profiling |
+-----+
| YES              |
+-----+

--查看是否开启
mysql> select @@profiling;
+-----+
| @@profiling |
+-----+
|          0 |
+-----+

--查看保留记录数
mysql> select @@profiling_history_size ;
+-----+
| @@profiling_history_size |
+-----+
|              15 |
+-----+
```

这个功能虽说在MySQL5.7中已经被废弃了，官方有提到以后版本将会去掉，将会被Performance Schema替换掉，但目前MySQL新版本依然可以使用，还是非常实用的功能。

profile具体如何使用，可以查看profile使用帮助：

```
mysql> help profile
```

开启profile

使用下面命令开启：

```
mysql> set profiling=1;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> select @@profiling;
+-----+
| @@profiling |
+-----+
|          1 |
+-----+
1 row in set, 1 warning (0.00 sec)
```

执行要分析的sql

-SQL1（性能差的）

```
mysql> select * from t21 where id in (select id from t21 where id=7369 union select id from
+-----+-----+-----+-----+-----+-----+
| id | name | age | face_value | position | hire_date |
+-----+-----+-----+-----+-----+-----+
| 7369 | szr7369 | 43 | 89 | position | 2025-01-20 |
| 7844 | szr7844 | 55 | 88 | position | 2025-01-20 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.25 sec)
```

-SQL2（优化后的）

szrsu

关注

216

文章

208

粉丝

351K+

浏览量

获得了 2725 次点赞

内容获得 727 次评论

获得了 1472 次收藏

热门文章

- 不重层linux系统扫描新增磁盘

2022-09-19

7551浏览
- docker 主机连接两个网络，桥接网络上网（双网卡）

2022-10-12

6747浏览
- MySQL中清除binlog的几种方法

2023-01-16

6555浏览
- postgresql 主从切换过程

2023-01-07

5598浏览
- 深入理解iostat命令，哪些指标比较重要？

2022-12-07

5216浏览

在线实训环境入口

MySQL

MySQL在线实训环境

查看详情

最新文章

- 揪出内鬼，快速定位用户登录失败的相关信息

2025-02-07

351浏览
- GoldenGate高手秘籍：启用GoldenGate TRACE调试，疑难杂症无所遁形！

2025-01-20

527浏览
- 努力终有回报，我也成为Oracle ACE啦！这是过去一年的努力，最好的成果与见...

2025-01-16

318浏览
- OGG复制进程延迟高，别怕，我来告诉你怎么解决！

2025-01-03

329浏览
- 级联OGG DDL复制的坑，关键参数你不可不知道！

2024-12-31

241浏览

目录

- profile工具

开启profile
- 执行要分析的sql

查看sql执行详情

停止profile
- trace工具

开启trace

执行要分析的sql

查看trace详情

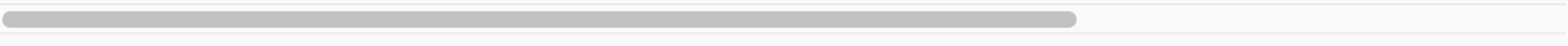
关闭trace
- 总结


```
mysql> select * from t21 where id in (7369,7844);
+-----+-----+-----+-----+-----+-----+
| id   | name   | age  | face_value | position | hire_date |
+-----+-----+-----+-----+-----+-----+
| 7369 | szr7369 | 43   | 89         | position | 2025-01-20 |
| 7844 | szr7844 | 55   | 88         | position | 2025-01-20 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

查看sql执行详情

–执行show profiles，查看SQL语句执行的耗时，进行对比

```
mysql> show profiles;
+-----+-----+-----+
| Query_ID | Duration | Query
+-----+-----+-----+
| 1 | 0.25657650 | select * from t21 where id in (select id from t21 where id=7369 u
| 2 | 0.00036950 | select * from t21 where id in (7369,7844)
+-----+-----+-----+
2 rows in set, 1 warning (0.00 sec)
```



可以看到两条语句的执行时间差异。

通过show profile for query query_id 语句可以查看到指定SQL执行过程中每个线程的状态和消耗的时间：

```
mysql> show profile for query 1;
+-----+-----+
| Status | Duration |
+-----+-----+
| executing | 0.000002 |
| executing | 0.000002 |
| executing | 0.000002 |
.....
| end | 0.000008 |
| query end | 0.000003 |
| waiting for handler commit | 0.000025 |
| closing tables | 0.000012 |
| freeing items | 0.000150 |
| logging slow query | 0.000151 |
| cleaning up | 0.000016 |
+-----+-----+

mysql> show profile for query 2;
+-----+-----+
| Status | Duration |
+-----+-----+
| starting | 0.000070 |
| Executing hook on transaction | 0.000003 |
| starting | 0.000006 |
| checking permissions | 0.000005 |
| Opening tables | 0.000038 |
| init | 0.000003 |
| System lock | 0.000006 |
| optimizing | 0.000008 |
| statistics | 0.000035 |
| preparing | 0.000016 |
| executing | 0.000049 |
| end | 0.000003 |
| query end | 0.000002 |
| waiting for handler commit | 0.000008 |
| closing tables | 0.000006 |
| freeing items | 0.000105 |
| cleaning up | 0.000008 |
+-----+-----+
17 rows in set, 1 warning (0.00 sec)
```

复制

在获取到最消耗时间的线程状态后，MySQL支持进一步选择all、cpu、block io 、context switch、page faults等明细类型类查看MySQL在使用什么资源上耗费了过高的时间。

–指定SQL的CPU开销

```
mysql> show profile cpu for query 1;
```

–指定SQL的内存开销

```
mysql> show profile memory for query 1;
```

–指定SQL的IO开销


```
mysql> show profile block io for query 1;
```

-同时查看不同资源开销

```
mysql> show profile block io,cpu for query 1;
```

MySQL所有的profile都被记录到了information_schema.profilng表，用下面的语句查询某条SQL开销并且按照耗时倒叙排序：

###设置要查询的 profile id，然后执行如下SQL即可复制

```
SET @query_id = 1;
SELECT
    STATE,
    SUM(DURATION) AS Total_R,
    ROUND(100 * SUM(DURATION) / (SELECT
        SUM(DURATION)
        FROM
            INFORMATION_SCHEMA.PROFILING
        WHERE
            QUERY_ID = @query_id),
    2) AS Pct_R,
    COUNT(*) AS Calls,
    SUM(DURATION) / COUNT(*) AS 'R/Call'
FROM
    INFORMATION_SCHEMA.PROFILING
WHERE
    QUERY_ID = @query_id
GROUP BY STATE
ORDER BY Total_R DESC;
```

STATE	Total_R	Pct_R	Calls	R/Call
executing	0.000236	49.68	93	0.0000025376
freeing items	0.000136	28.63	1	0.0001360000
logging slow query	0.000043	9.05	1	0.0000430000
waiting for handler commit	0.000020	4.21	1	0.0000200000
cleaning up	0.000019	4.00	1	0.0000190000
closing tables	0.000012	2.53	1	0.0000120000
end	0.000006	1.26	1	0.0000060000
query end	0.000003	0.63	1	0.0000030000

8 rows in set, 2 warnings (0.00 sec)

这里可以看出，最大的开销就是在执行（executing）。

注：state字段含义

starting：开始
checking permissions：检查权限
Opening tables：打开表
init：初始化
System lock：系统锁
optimizing：优化
statistics：统计
preparing：准备
create tmp table：创建临时表(如group时储存中间结果)
executing：执行
converting HEAP to MyISAM：查询结果太大时，把结果放在磁盘
Copying to tmp table on disk：把内存临时表复制到磁盘
Sending data：发送数据
Sorting result：排序
end：结束
query end：查询结束
removing tmp table：关闭表/去除TMP表
freeing items：释放物品
cleaning up：清理

停止profile

停止profile,可以设置profiling参数，或者在session退出之后，profiling会被自动关闭。

```
mysql> set profiling=off;
```

trace工具

explain 可以查看 SQL 执行计划，但是无法知道它为什么做这个决策，如果想确定多种索引方案之间是如何选择的或者排序时选择的是哪种排序模式，有什么好的办法吗？从MySQL5.6开始，可以使用trace工具主要看的是MySQL大致的优化和计算成本过程，深入分析优化器如何选择执行计划，帮助我们更好理解优化器的行为。

默认情况下trace是关闭的，对于MySQL的性能会有一些的影响，在生产环境建议只在需要分析SQL语句的执行计划生成过程时才开启，使用完毕后记得关闭trace命令。

开启trace

查看trace相关的配置参数：

mysql> show variables like '%trace%';

复制

Variable_name	Value
optimizer_trace	enabled=off,one_line=off
optimizer_trace_features	greedy_search=on,range_optimizer=on,dynamic_range=on,rep
optimizer_trace_limit	1
optimizer_trace_max_mem_size	1048576 ##trace最大能够使用的内存大小，避免解析过程中因为默认内存达
optimizer_trace_offset	-1

5 rows in set, 1 warning (0.01 sec)

开启trace：

--打开trace，设置格式为 JSON
mysql> set optimizer_trace="enabled=on",end_markers_in_json=on;
Query OK, 0 rows affected (0.00 sec)

执行要分析的sql

mysql> select * from t24 where height=182;

查看trace详情

查询information_schema.optimizer_trace就可以知道MySQL生成了怎样的执行计划：

mysql> select * from information_schema.optimizer_trace\G;

trace文件分析大致分为三个过程：

准备阶段：对应文本中的 join_preparation

优化阶段：对应文本中的 join_optimization

执行阶段：对应文本中的 join_execution

使用时，重点关注优化阶段和执行阶段。

***** 1. row *****
QUERY: select * from t24 where height=182
TRACE: {

"steps": [
 {
 # 第一步：prepare准备阶段
 "join_preparation": {
 "select#": 1,
 "steps": [
 {
 # 进行SQL语句的格式化操作
 "expanded_query": "/* select#1 */ select `t24`.`id` AS `id`,`t24`.`name` AS `nc
 }
] /* steps */
 } /* join_preparation */
 },
 {
 "join_optimization": {
 "select#": 1,
 "steps": [
 {
 "condition_processing": {
 "condition": "WHERE",
 "original_condition": "(`t24`.`height` = 182)",
 "steps": [
 {
 # 第一步优化：等值优化
 "transformation": "equality_propagation",
 #优化之后的输出结果
 "resulting_condition": "multiple equal(182, `t24`.`height`)"
 },
 {
 # 第二步优化：常量比值优化
 "transformation": "constant_propagation",
 #优化之后的输出结果
 "resulting_condition": "multiple equal(182, `t24`.`height`)"
 },
]
 }
 }
]
 }
 }
]


```
        # 条件移除优化，去掉一些不必要的条件
        "transformation": "trivial_condition_removal",
        "resulting_condition": "multiple equal(182, `t24`.`height`)"
    }
] /* steps */
} /* condition_processing */
},
{
    "substitute_generated_columns": {
    } /* substitute_generated_columns */
},
{
#表的依赖分析
    "table_dependencies": [
        {
            #表名
            "table": "`t24`",
            #判断行记录是否有可能为空
            "row_may_be_null": false,
            "map_bit": 0,
            "depends_on_map_bits": [
            ] /* depends_on_map_bits */
        }
    ] /* table_dependencies */
},
{
#使用的优化的索引
    "ref_optimizer_key_uses": [
        {
            "table": "`t24`",
            "field": "height",
            "equals": "182",
            "null_rejecting": true
        }
    ] /* ref_optimizer_key_uses */
},
{
    "rows_estimation": [
        {
            "table": "`t24`",
            "range_analysis": {
            #全表扫描预估扫描
            "table_scan": {
                "rows": 12,
                "cost": 3.55
            } /* table_scan */,
            #可能用到的索引
            "potential_range_indexes": [
                {
                    #主键索引 => 不适用
                    "index": "PRIMARY",
                    "usable": false,
                    "cause": "not_applicable"
                },
                {
                    #索引idx_height => 适用
                    "index": "idx_height",
                    "usable": true,
                    "key_parts": [
                        "height",
                        "id"
                    ] /* key_parts */
                }
            ] /* potential_range_indexes */,
            "setup_range_conditions": [
            ] /* setup_range_conditions */,
            "group_index_range": {
                "chosen": false,
                "cause": "not_group_by_or_distinct"
            } /* group_index_range */,
            "skip_scan_range": {
                "potential_skip_scan_indexes": [
                    {
                        "index": "idx_height",
                        "usable": false,
                        "cause": "query_references_nonkey_column"
                    }
                ] /* potential_skip_scan_indexes */
            } /* skip_scan_range */,
            "analyzing_range_alternatives": {
                "range_scan_alternatives": [
                    {
                        "index": "idx_height",
                        "ranges": [
                            "height = 182"
                        ] /* ranges */,
                        "index_dives_for_eq_ranges": true,
                        "rowid_ordered": true,
                        "using_mrr": false,
```



```
        "index_only": false,
        "in_memory": 1,
        "rows": 12,
        "cost": 4.46,
        "chosen": false,
        "cause": "cost"
      }
    ] /* range_scan_alternatives */,
    "analyzing_roworder_intersect": {
      "usable": false,
      "cause": "too_few_roworder_scans"
    } /* analyzing_roworder_intersect */
  } /* analyzing_range_alternatives */
} /* range_analysis */
}
] /* rows_estimation */
},
{
#最后的索引选择方案
  "considered_execution_plans": [
    {
      "plan_prefix": [
    ] /* plan_prefix */,
      "table": "`t24`",
      #最优的访问路径
      "best_access_path": {
        "considered_access_paths": [
          {
            #使用索引的方式
            "access_type": "ref",
            "index": "idx_height",
            "rows": 12,
            "cost": 1.7,
            "chosen": true
          },
          {
            #扫描表的方式
            "rows_to_scan": 12,
            "access_type": "scan",
            "resulting_rows": 12,
            "cost": 1.45,
            "chosen": true
          }
        ] /* considered_access_paths */
      } /* best_access_path */,
      #最终选择的方案
      "condition_filtering_pct": 100,
      "rows_for_plan": 12,
      "cost_for_plan": 1.45,
      "chosen": true
    }
  ] /* considered_execution_plans */
},
{
  "attaching_conditions_to_tables": {
    "original_condition": "(`t24`.`height` = 182)",
    "attached_conditions_computation": [
    ] /* attached_conditions_computation */,
    "attached_conditions_summary": [
      {
        "table": "`t24`",
        "attached": "(`t24`.`height` = 182)"
      }
    ] /* attached_conditions_summary */
  } /* attaching_conditions_to_tables */
},
{
  "finalizing_table_conditions": [
    {
      "table": "`t24`",
      "original_table_condition": "(`t24`.`height` = 182)",
      "final_table_condition   ": "(`t24`.`height` = 182)"
    }
  ] /* finalizing_table_conditions */
},
{
  "refine_plan": [
    {
      "table": "`t24`"
    }
  ] /* refine_plan */
}
] /* steps */
} /* join_optimization */
},
{
  "join_execution": {
    "select#": 1,
    "steps": [
```



```
    ] /* steps */
  } /* join_execution */
}
] /* steps */
}
MISSING_BYTES_BEYOND_MAX_MEM_SIZE: 0
INSUFFICIENT_PRIVILEGES: 0
1 row in set (0.00 sec)
```

关闭trace

当分析完SQL，立即关闭trace：

```
mysql> set session optimizer_trace="enabled=off";
```

总结

通过profile工具可以清楚了解到SQL到底慢在哪个环节；通过trace工具查看优化器如何选择执行计划，获取每个可能的选择代价。

🔗 墨力计划 mysql

「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





【版权声明】本文为墨天轮用户原创内容，转载时必须标注文章的来源（墨天轮），文章链接，文章作者等基本信息，否则作者和墨天轮有权追究责任。如果您发现墨天轮中有涉嫌抄袭或者侵权的内容，欢迎发送邮件至：contact@modb.pro进行举报，并提供相关证据，一经查实，墨天轮将立刻删除相关内容。

评论

分享你的看法，一起交流吧~

 飞天  LV.5
MySQL性能分析的“秘密武器”，深度剖析SQL问题
3天前  点赞  评论

 鲁鲁  LV.5
MySQL性能分析的“秘密武器”，深度剖析SQL问题
20天前  点赞  评论

 小草  LV.6
MySQL性能分析的“秘密武器”，深度剖析SQL问题
24天前  点赞  评论

相关阅读

【干货】2024年下半年墨天轮最受欢迎的50篇技术文章+文档
墨天轮编辑部 1562次阅读 2025-02-13 10:42:44

2025年1月“墨力原创作者计划”获奖名单公布
墨天轮编辑部 348次阅读 2025-02-13 15:07:02

MySQL 主从节点切换指导
CuiHulong 302次阅读 2025-01-23 11:50:29

[MYSQL] 忘记root密码时, 不需要重后也能强制修改了!
大大刺猬 286次阅读 2025-02-06 11:12:15

mysql 内存使用率高问题排查
蔡璐 266次阅读 2025-02-06 10:02:23

MySQL 9.2.0 中的更新（2025-01-21，创新版本）
通讯员 151次阅读 2025-01-22 09:54:21

MySQL基础高频面试题－划重点、敲难点
锁钥 146次阅读 2025-02-03 07:52:28

MySQL 底层数据&日志刷新策略解读
CuiHulong 133次阅读 2025-02-11 10:56:13

[MYSQL] mysql主从延迟案例(有索引但无主键)
大大刺猬 107次阅读 2025-01-21 13:53:21

PG vs MySQL 统计信息收集的异同
进击的CJR 84次阅读 2025-02-05 17:29:34