

MySQL 升级后查询性能跳水，排序竟成“罪魁祸首”？

原创 龚唐杰 爱可生开源社区 2025年03月03日 19:30 上海

作者：龚唐杰，爱可生 DBA 团队成员，主要负责 MySQL 技术支持，擅长 MySQL、PG、国产数据库。

爱可生开源社区出品，原创内容未经授权不得随意使用，转载请联系小编并注明来源。

本文约 1100 字，预计阅读需要 3 分钟。

1 背景及分析

近期，某客户完成对数据库 MySQL 5.7 到 8.0 的版本升级，升级后查询性能显著变慢。

原来是 MySQL 8.0 对某些 ORDER BY 相关的参数修改，导致了优化器不生效。下面我们进入本次的 SQL 优化分析，也建议升级后有类似情况的读者自检。

分析过程

首先，查看慢日志及对应的表结构。

- 慢 SQL：`select * from xx where xx order by xx limit xx .`
- 执行计划：发现 `order by` 的字段没有索引，若加上索引可从原来的 4 秒变为毫秒级别。

在升级前（MySQL 5.7），该字段没有索引，查询只需 1 秒左右，需要找到这个原因。

通过执行计划（`profile`，`trace` 等方式）对比了升级前后的区别，发现只有 `profile` 会有明显的区别。其中，MySQL 5.7 的耗时主要在 `Creating sort index` 阶段，而 MySQL 8.0 的耗时都是在执行阶段。

在 MySQL 8.0 中 SELECT 少数字段时间也在 1 秒左右。随着 SELECT 查询的字段增多，时间也越来越长。当 `select *` 时能达到 4 秒，而 MySQL 5.7 中不管多少字段都是 1 秒左右。

根据以上信息可以推测，变慢主要在排序环节，需要进一步了解 MySQL 8.0 的排序方式发生了哪些改变。

通过 [MySQL 官网文档^{\[1\]}](#) 可知：

- MySQL 8.0.20 之前的版本：排序跟 `max_length_for_sort_data` 参数有关。当需要排序的行的大小大于参数设置对应的值时（byte），会使用 `row_id` 排序，反之使用全字段

排序。通过测试，在 MySQL 5.7 版本时，设置参数的值若大于所有列对应的大小，`select * 查询`也需要耗时 4 秒左右。

- MySQL 8.0.20 及之后的版本：`max_length_for_sort_data` 参数被废弃，不再生效。

官网中对该参数的调整说明

分析完毕，下面我们将进行验证。

2 验证测试

在本次升级涉及的具体版本是 MySQL 5.7.44 和 MySQL 8.0.30。根据上面的分析过程，推断 MySQL 5.7.44 版在涉及到排序查询时会受到 `max_length_for_sort_data` 的影响，而 MySQL 8.0.30 则不会。

数据准备

在 MySQL 5.7 和 8.0 版本库中建表并插入 400W 行数据。

```
CREATE TABLE `t` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `create_date` datetime DEFAULT NULL,  
  `status` int DEFAULT NULL,  
  `col1` varchar(50) DEFAULT NULL,  
  `col2` varchar(50) DEFAULT NULL,  
  `col3` varchar(50) DEFAULT NULL,  
  `col4` varchar(50) DEFAULT NULL,  
  `col5` varchar(50) DEFAULT NULL,  
  `col6` varchar(50) DEFAULT NULL,  
  `col7` varchar(50) DEFAULT NULL,  
  `col8` varchar(50) DEFAULT NULL,  
  `col9` varchar(50) DEFAULT NULL,  
  `col10` varchar(50) DEFAULT NULL,  
  `col11` varchar(255) DEFAULT NULL,  
  `col12` varchar(255) DEFAULT NULL,  
  `col13` varchar(255) DEFAULT NULL,  
  `col14` varchar(255) DEFAULT NULL,  
  `col15` varchar(255) DEFAULT NULL,  
  `col16` varchar(255) DEFAULT NULL,  
  `col17` varchar(255) DEFAULT NULL,  
  `col18` varchar(255) DEFAULT NULL,  
  `col19` varchar(255) DEFAULT NULL,  
  `col20` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB;
```

-- 插入数据过程略

```
select count(*) from `t`;
+-----+
| count( * ) |
+-----+
|      4194304 |
+-----+
1 row in set (0.11 sec)
```

在 MySQL 5.7 和 8.0 版本环境中执行（参数配置一致），分别执行查询三个字段和查询所有字段两种 SELECT 语句。

```
-- 查询三个字段
select id,create_date,status from t where status=1 order by create_date desc limit 1;

-- 查询所有字段
select * from t where status=1 order by create_date desc limit 1;
```

MySQL 5.7 的两种 SELECT 语句执行时间均为 1 秒左右。

MySQL 5.7 两种查询对比

MySQL 8.0 查询三个字段 1 秒左右，查询所有字段则为 4 秒左右。

MySQL 8.0 两种查询对比

在 MySQL 8.0.30 查询的字段越多，时间越长。

对比查询 7 个字段和 11 个字段

若在 MySQL 5.7.44 中，把 `max_length_for_sort_data` 参数的值设置大于所有列的大小时，查询时间也会变慢（全字段排序）。

3 结论及优化方案

MySQL 8.0.20 及之后的版本，针对无索引的排序方式发现改变。不会再通过 `max_length_for_sort_data` 参数来判断，而是通过查询的字段和排序的字段大小动态来进行排序。所以在查询列较多时会导致比 MySQL 5.7 更慢。

最好的解决方式是给排序字段加上索引：)

参考资料

- [1] order-by-optimization: <https://dev.mysql.com/doc/refman/8.0/en/order-by-optimization.html>

本文关键字：#MySQL# #排序# #SQL优化#

故障分析 | 如何解决由触发器导致 MySQL 内存溢出？

故障分析 | 查询 `ps.data_locks` 导致 MySQL hang 住

技术分享 | 深入理解 MySQL 中的 `SQL_MODE`

技术分享 | MySQL Undo 工作机制历史演变

技术分享 | MySQL 隐式转换必知必会

技术分享 | MySQL VARCHAR 最佳长度评估实践


故障分析 | TCP 缓存超负荷导致的 MySQL 连接中断



Github : <https://github.com/actiontech/sqlc>

 文档 : <https://actiontech.github.io/sqlc-docs/>

 官网 : <https://opensource.actionsky.com/sqlc/>

 微信群 : 请添加小助手加入 ActionOpenSource

 商业支持：<https://www.actionsky.com/sql>e

扫描下方二维码，进行在线咨询预约：

此外，您也可以直接联系我们的商业支持团队获取更多信息，联系方式如下：

400-820-6580 / 13916131869 / 18930110869

MySQL 231 排序 2 SQL优化 15

MySQL · 目录

上一篇

技术分享 | 某二手交易平台数据安全实践（建议对照自检）

下一篇

MySQL 核心模块揭秘 | 专栏合集