CPU又100%了

原创 阿陈98 呆呆的私房菜 2025年03月15日 00:00 广东

Whoami：5年+金融、政府、医疗领域工作经验的DBA
Certificate：PGCM、OCP、YCP
Skill：Oracle、Mysql、PostgreSQL、国产数据库
Platform：CSDN、墨天轮、公众号（呆呆的私房菜）

阅读本文可以了解可以了解Mysql CPU占用率高的时候如何进行定位和优化，帮助DBA们在处理故障时提供排查和优化思路。

01 ▸ 模拟测试数据

- 1. 建表和插入数据

```
CREATE TABLE `large_data_table` (
  `id` BIGINT(20) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '主键',
  `user_id` VARCHAR(36) NOT NULL DEFAULT '' COMMENT '用户ID（UUID格式）',
  `name` VARCHAR(50) NOT NULL DEFAULT '' COMMENT '用户名',
  `age` TINYINT(3) UNSIGNED NOT NULL DEFAULT 0 COMMENT '年龄',
  `province_id` INT(10) UNSIGNED NOT NULL COMMENT '省份ID',
  `city_id` INT(10) UNSIGNED NOT NULL COMMENT '城市ID',
  `create_time` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '创
  `update_time` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
  PRIMARY KEY (`id`),
  KEY `idx_user_id` (`user_id`),
  KEY `idx_create_time` (`create_time`),
  KEY `idx_province_city` (`province_id`, `city_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
  AUTO_INCREMENT=1
  COMMENT='亿级数据表';


-- 插入1亿条数据
DELIMITER $$
CREATE PROCEDURE generate_100m_data()
BEGIN
  DECLARE i INT DEFAULT 1;
  WHILE i <= 100000000 DO
    INSERT INTO large_data_table (user_id, name, age, province_id, cit
```

```
    VALUES (
      UUID(),
      CONCAT('用户', i),
      FLOOR(18 + (RAND() * 62)),  -- 年龄范围18-80岁
      FLOOR(1 + (RAND() * 33)),    -- 省份ID范围1-34
      FLOOR(100 + (RAND() * 3400)) -- 城市ID范围101-3420
    );
    SET i = i + 1;
    -- 每1万条提交一次事务
    IF i % 10000 = 0 THEN
      COMMIT;
      START TRANSACTION;
    END IF;
  END WHILE;
END
$$
DELIMITER ;

-- 调用存储过程
CALL generate_100m_data();
```

- 2. 模拟cpu高消耗sql

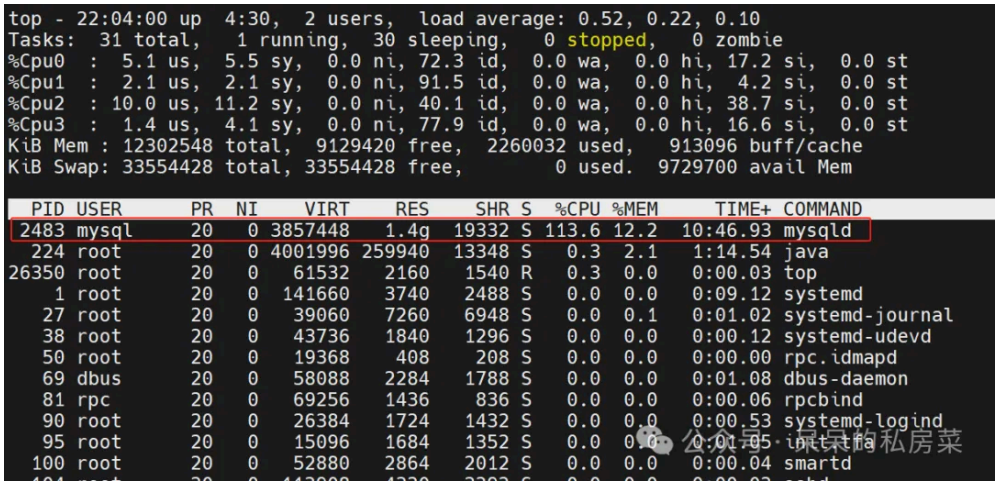
```
select * from large_data_table order by rand() limit 100;
```

- 3. 客户开始反馈业务卡顿、CPU消耗100%了。。。

02 问题排查

- 1. top查看进程情况

```
top
```



- 通过top工具发现mysql进程CPU使用率已经超过100%，继续分析。

2. 查看内存使用情况

```
free -h
```

```
               total        used        free      shared  buff/cache   available
Mem:            11G        2.2G        8.7G         17M        891M        9.3G
Swap:           31G          0B         31G
```

- 通过 free 工具发现内存使用情况还好，继续分析。

3. 检查MYSQL线程情况

```
top -H -p `pidof mysqld`
```

```
top - 22:09:27 up  4:35,  2 users,  load average: 0.62, 0.38, 0.21
Threads:  43 total,   1 running,  42 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.8 us, 10.0 sy,  0.0 ni, 76.5 id,  0.0 wa,  0.0 hi, 11.6 si,  0.0 st
KiB Mem : 12302548 total,  9129032 free,  2260224 used,   913292 buff/cache
KiB Swap: 33554428 total, 33554428 free,        0 used.  9729384 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 2775 mysql     20   0 3857448   1.4g  19332 R 99.9 12.2   1:55.40 connection
 2503 mysql     20   0 3857448   1.4g  19332 S  1.3 12.2   0:16.81 ib_io_rd-1
 2505 mysql     20   0 3857448   1.4g  19332 S  1.3 12.2   0:17.02 ib_io_rd-3
 2506 mysql     20   0 3857448   1.4g  19332 S  1.3 12.2   0:16.71 ib_io_rd-4
 2504 mysql     20   0 3857448   1.4g  19332 S  0.7 12.2   0:17.77 ib_io_rd-2
 2521 mysql     20   0 3857448   1.4g  19332 S  0.7 12.2   2:06.82 ib_log_files_g
 2509 mysql     20   0 3857448   1.4g  19332 S  0.3 12.2   0:02.15 ib_io_wr-3
 2518 mysql     20   0 3857448   1.4g  19332 S  0.3 12.2   0:12.59 ib_log_flush
 2545 mysql     20   0 3857448   1.4g  19332 S  0.3 12.2   0:00.02 xpl_worker-1
 2552 mysql     20   0 3857448   1.4g  19332 S  0.3 12.2   0:14.06 ib_clone_gtid
 2483 mysql     20   0 3857448   1.4g  19332 S  0.0 12.2   0:00.30 mysqld
 2501 mysql     20   0 3857448   1.4g  19332 S  0.0 12.2   0:02.10 ib_io_ibuf
```


- 通过 top 工具可以明显发现 2775 线程CPU使用率异常。

4. 查看线程详细情况

```
select a.user, a.host, a.db, b.thread_os_id, b.thread_id, a.id processl
from information_schema.processlist a, performance_schema.threads b
where a.id = b.processlist_id
and b.thread_os_id = 2775;
```

```
| user | host            | db   | thread_os_id | thread_id | processlist_id | command | time | state     | info                                                                |
| [redacted] | [redacted]:57988 | chen |         2775 |        58 |             10 | Query   |   61 | executing | select * from large_data_table order by rand() limit 100000 |
1 row in set (0.00 sec)
```

```
select *
from performance_schema.events_statements_current
where thread_id in (select thread_id from performance_schema.threads wh
```

```
*************************** 1. row ***************************
           THREAD_ID: 58
            EVENT_ID: 21
        END_EVENT_ID: NULL
          EVENT_NAME: statement/sql/select
              SOURCE: init_net_server_extension.cc:95
         TIMER_START: 16867033233563000
           TIMER_END: 16926732779960000
          TIMER_WAIT: 59699546397000
           LOCK_TIME: 2000000
            SQL_TEXT: select * from large_data_table order by rand() limit 100000
              DIGEST: 56abae4486173328c73435cd6bc2bd3d65d798bbcee18dc4e786b2984adf30d3
         DIGEST_TEXT: SELECT * FROM `large_data_table` ORDER BY `rand` ( ) LIMIT ?
      CURRENT_SCHEMA: chen
         OBJECT_TYPE: NULL
       OBJECT_SCHEMA: NULL
        OBJECT_NAME: NULL
  OBJECT_INSTANCE_BEGIN: NULL
        MYSQL_ERRNO: 0
     RETURNED_SQLSTATE: NULL
          MESSAGE_TEXT: NULL
              ERRORS: 0
            WARNINGS: 0
        ROWS_AFFECTED: 0
           ROWS_SENT: 0
       ROWS_EXAMINED: 0
CREATED_TMP_DISK_TABLES: 0
     CREATED_TMP_TABLES: 0
       SELECT_FULL_JOIN: 0
 SELECT_FULL_RANGE_JOIN: 0
         SELECT_RANGE: 0
     SELECT_RANGE_CHECK: 0
            SELECT_SCAN: 1
      SORT_MERGE_PASSES: 0
             SORT_RANGE: 0
              SORT_ROWS: 0
              SORT_SCAN: 1
          NO_INDEX_USED: 1
     NO_GOOD_INDEX_USED: 0
       NESTING_EVENT_ID: NULL
     NESTING_EVENT_TYPE: NULL
    NESTING_EVENT_LEVEL: 0
          STATEMENT_ID: 36
              CPU_TIME: 0
       EXECUTION_ENGINE: PRIMARY
1 row in set (0.00 sec)
```


ok，抓到慢日志，定位是哪个应用的SQL，确认是否可以kill

```
kill 2775;
```

- **5. 如果是事后才发现MYSQL有CPU高的情况，那怎么办？只能试试用慢日志分析一下咯。**

```
-- 慢日志相关参数
show variables like 'long_query_time';
show variables like 'slow_query_log';

-- 查看慢日志，找到具体故障时间点的sql
less /mysql/mysql8/slowlog/mysql-slow.log
```

```
# Time: 2025-03-14T22:22:37.348426+08:00
# User@Host: repl[repl] @  [10.28.12.21]  Id:    10
# Query_time: 89.699412  Lock_time: 0.000003 Rows_sent: 100000  Rows_examined: 25089999
SET timestamp=1741962067;
select * from large_data_table order by rand() limit 100000;
```

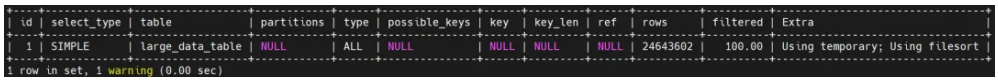

03 ▸ 问题处理

- 1. 常规的手段是先 kill 会话，然后分析SQL执行计划并进行优化或SQL重构。

```
kill 2775;

```

```
3 explain select * from large_data_table order by rand() limit 100000;
```



```
+----+-------------+------------------+------------+------+---------------+------+---------+------+----------+----------+-----------------------------+
| id | select_type | table            | partitions | type | possible_keys | key  | key_len | ref  | rows     | filtered | Extra                       |
+----+-------------+------------------+------------+------+---------------+------+---------+------+----------+----------+-----------------------------+
|  1 | SIMPLE      | large_data_table | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 24643602 |   100.00 | Using temporary; Using filesort |
+----+-------------+------------------+------------+------+---------------+------+---------+------+----------+----------+-----------------------------+
1 row in set, 1 warning (0.00 sec)
```

- 2. 关注索引的使用情况，一般where、join、max()、min()、order by、group by等字句用到的字段要创建相应的索引；同时要关注二级索引的正确使用；
- 3. 数据库参数优化：优化 key_buffer_size 、 table_cache 、 innodb_buffer_pool_size 、 innodb_log_file_size 等参数的大小；

本文内容就到这啦，相信本篇的内容能为您排查MYSQL思路带来一些参考和帮助。我们下篇再见！

呆呆的私房菜

记录PostgreSQL、Oracle、MySQL等数据库相关内容。

80篇原创内容

公众号

点击上方公众号，关注我吧！

阿陈98

喜欢作者

MySQL专栏 · 目录

上一篇
高效SQL怎么来的？

下一篇
MySQL严重故障场景数据恢复