关于 MySQL 慢查询的介绍及作业

慢查询

什么是慢查询

MySQL 的慢查询日志是 MySQL 提供的一种日志记录，它用来记录在 MySQL 中响应时间超过阀值的语句，阈值指的是运行时间超过 long_query_time 值的 SQL，则会被记录到慢查询日志中。long_query_time 的默认值为 10，意思是运行 10 秒 以上的语句。默认情况下，MySQL 数据库并不启动慢查询日志，需要我们手动来设置这个参数。

慢查询需要知道的 “点”

- 企业级开发中，慢查询日志是会打开的。但是这同样会带来一定的性能影响。
- 慢查询日志支持将日志记录写入文件，也支持将日志记录写入数据库表
-
- 默认的阈值（long_query_time）是 10，这个显然不可用，通常，对于用户级应用而言，我们将它设置为 0.2
- 慢查询相关的变量

查看变量的 SQL 语句

SQL	*含义*	*备注*
show variables like ‘slow_query_log’;	是否开启慢查询日志	ON 表示开启；OFF 表示关闭
show variables like ‘slow_query_log_file’;	慢查询日志存储路径	根据 MySQL 版本不同而不同
show variables like ‘long_query_time’;	慢查询阈值	默认值是 10s，使用慢查询则一定需要更改
show variables like ‘log_queries_not_using_indexes’;	未使用索引的查询也被记录到慢查询日志中	OFF 表示关闭，通常不会被开启
show variables like ‘log_output’;	慢查询日志存储方式	默认值是 FILE，表示将日志存入文件；修改为 TABLE，表示将日志存入数据库（mysql.slow_log 表）
show global status like ‘Slow_queries’;	查看有多少条慢查询记录	

关于变量的说明

- ** 修改变量可以使用命令：set global long_query_time = 0.2; （更常见的做法是修改 MySQL 的配置 my.cnf） **
-
- ** 日志记录到系统的专用日志表中，要比记录到文件耗费更多的系统资源。所以，不要将慢查询日志记录到表中。 **
- 慢查询日志的分析工具（优化慢查询）

mysqldumpslow

MySQL 内置了工具 mysqldumpslow 用于解析 MySQL 慢查询日志，并打印其内容摘要。

- 使用语法

```
mysqldumpslow [options] [log_file ...]
```

- 常用选项（options）解释

```
-g pattern：只显示与模式匹配的语句，大小写不敏感。

-r：反转排序顺序。

-s sort_type：如何排序输出，可选的 sort_type 如下


  t：按查询总时间排序。

  l：按查询总锁定时间排序。

  r：按总发送行排序。

  c：按计数排序。

  at：按查询时间或平均查询时间排序。

  al：按平均锁定时间排序。

  ar：按平均行发送排序。


默认情况下，mysqldumpslow 按平均查询时间（相当于-s at）排序。


-t N：是 top n 的意思，即返回前面多少条的数据。

-v：详细模式。
```

- 使用示例

```
# 显示 2 条结果，且按照查询总时间排序，且过滤 group by 语句

mysqldumpslow -t 2 -s t -g "group by" slow_query_log_file


# 按照时间排序的前 10 条里面含有左连接的查询语句

mysqldumpslow -s t -t 10 -g “left join” slow_query_log_file


# 返回记录集最多的 10 个 SQL

mysqldumpslow -s r -t 10 slow_query_log_file


# 可以结合 more 一起使用，避免一次显示过多 SQL 语句

mysqldumpslow -s r -t 20 slow_query_log_file | more


# 访问次数最多的 10 个 SQL

mysqldumpslow -s c -t 10 slow_query_log_file


...
```

- mysqldumpslow 结果信息

```
Count: 这种类型的语句执行了几次

Time: 这种类型的语句执行的最大时间

Lock: 这种类型语句执行时等待锁的时间
```

```
Rows: 单次返回的结果数


Count: 2  Time=3.21s (7s)  Lock=0.00s (0s)  Rows=1.0 (2), root[root]@localhost

代表的含义是：执行了 2 次，最大时间是 3.21s，总共花费时间 7s，等待锁的时间是 0s，单次返回的结果数是 1 条记录，2 次总共返回 2 条记录。
```

EXPLAIN

MySQL 提供了 EXPLAIN 命令，可以对慢查询（SELECT）进行分析，并输出 SELECT 执行的详细信息。我们可以针对输出的信息对慢查询语句进行合理的优化。

- 使用方法

```
explain select * from ad_unit_it where it_tag like '%球';
```

- EXPLAIN 输出信息及解释

```
mysql> explain select * from ad_unit_it where it_tag like '%球'G*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: ad_unit_it
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 3
     filtered: 33.33
        Extra: Using where
id:             查询的唯一标识符
select_type:    查询的类型
table:          查询的表
partitions:     匹配的分区
type:           join 类型
possible_keys:  查询中可能用到的索引
key:            查询中使用到的索引
key_len:        查询优化器使用了的索引字节数
ref:            哪个字段或常量与 key 一起被使用
rows:           当前的查询一共扫描了多少行（估值）
filtered:       查询条件过滤的数据百分比
Extra:          额外信息
```

-

select_type: 最常见的查询类型是 SIMPLE， 这表示查询中没有子查询，也没有 UNION 查询

-
-

type: 这个字段是判断查询是否高效的重要提示。可以判断查询是全表扫描还是索引扫描。例如: all 表示全表扫描，性能最差；

range 表示使用索引范围扫描，通常是

where 条件中带有数学比对的；index 表示全索引扫描，扫描索引而不扫描数据

-
-

possible_keys: 表示查询时可能会使用到的索引，但是并不表示一定会使用。真正的使用了哪些索引，由 key 决定

-
-

rows: MySQL 优化器会估算此次查询需要扫描的数据记录数（行数），这个值越小，查询效率越高

-
-

Extra: 这是查询语句所对应的“额外信息”， 常见的有 ：

-
- Using filesort: 表示 MySQL 需额外的排序操作，不能通过索引顺序达到排序效果。这样的查询应该是需要避免的，CPU 消耗很高。
- Using where: 在查找使用索引的情况下，需要回表去查询所需的数据
- Using index: 表示查询在索引树中就可查找所需数据，不用扫描表数据文件
- Using temporary: 查询过程会使用到临时表，常见于 order by、group by、join 等场景，性能较低

为什么会产生慢查询 ？

-

两张比较大的表进行 JOIN，但是没有给表的相应字段加索引

-
-

表存在索引，但是查询的条件过多，且字段顺序与索引顺序不一致

-

对很多查询结果进行 GROUP BY

-
-

...

-

任务

作业 1：你在工作中遇到过哪些慢查询 ？你是怎么优化的呢 ？

说明：大家可以将自己的答案发布到右侧的问答区，老师会针对大家回答的情况给予点评。

作业 2：你的应用设置的 MySQL 慢查询阈值时间是多少呢 ？为什么设置成这个值呢 ？

说明：大家可以将自己的答案发布到右侧的问答区，老师会针对大家回答的情况给予点评。

作业 3：优化慢查询的方案，绝大多数都是给查询条件的字段添加索引，所以，要搞清楚索引的原理与设计 ！

说明：大家可以将自己的答案发布到右侧的问答区，老师会针对大家回答的情况给予点评。