MySql的优化探讨

2020/8/4 王国庆

# 1、innodb存储引擎

建表时使用ENGINE关键字决定存储引擎

Create table innodb\_test(id int,name varchar(16),ENGINE INNODB)

5.58版本以后默认使用INNODB的引擎，之前是MYISAM

Create table innodb\_test2(id int,name varchar(16))

Innodb\_file\_pre\_table = ON 独立表空间，OFF系统表空间，

## 2、锁

在innodb中只有利用索引的更新，删除操作，才可以使用行级锁。

在实际开发的时候，如果遇到写操作，一定要确保update/delete语句的条件。

要能够使用索引否则程序不具备并发性。

# 3、性能

在事务的控制中，存在“事务区”来保证事务的完整性，在事务提交以后，这些事务区的数据会写到磁盘上，同时事务操作日志（log）也需要向磁盘中写入，logBuffer将每秒一次的的写入log file中，并且flush（刷到磁盘中区），该模式为系统默认。

设置innodb的线程并发数，默认值0表示不被限制，若要设置则和服务器的cpu的核心数相同，或者是CPU核心数的2倍。

show global VARIABLES like 'innodb\_thread\_concurrency';

可以了解到总的innodb buffer pool 获得的内存有多少，字典信息的buffer 多少， 以及到底目前innodb buffer pool 中使用了多少内存，以及一个重要的师表buffer hit (其他的数据库也有这样的信息，体现到底目前的内存状态）

# 4、Btree 索引

/\*

Navicat Premium Data Transfer

Source Server : RDS

Source Server Type : MySQL

Source Server Version : 50718

Source Host : localhost:3306

Source Schema : explain\_test

Target Server Type : MySQL

Target Server Version : 50718

File Encoding : 65001

Date: 09/05/2019 11:28:53

\*/

SET NAMES utf8mb4;

SET FOREIGN\_KEY\_CHECKS = 0;

-- ----------------------------

-- Table structure for t\_content

-- ----------------------------

DROP TABLE IF EXISTS `t\_content`;

CREATE TABLE `t\_content` (

`content\_id` bigint(20) NOT NULL COMMENT '内容id',

`channel\_id` bigint(20) NULL DEFAULT NULL COMMENT '频道编号',

`forum\_id` bigint(11) NULL DEFAULT NULL COMMENT '圈子编号',

`uid` bigint(255) NULL DEFAULT NULL COMMENT '会员id',

`status` int(255) NULL DEFAULT NULL COMMENT '状态 4-过审',

`comment\_count` int(11) NULL DEFAULT NULL COMMENT '评论数量',

`bookmark\_count` int(255) NULL DEFAULT NULL COMMENT '收藏数量',

`content\_text` text CHARACTER SET utf8mb4 COLLATE utf8mb4\_general\_ci NULL COMMENT '标题或正文',

`like\_count` int(255) NULL DEFAULT NULL COMMENT '点赞数量',

`hate\_count` int(255) NULL DEFAULT NULL COMMENT '讨厌数量',

`share\_url` varchar(1024) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '分享url',

`share\_count` int(255) NULL DEFAULT NULL COMMENT '分享数量',

`passtime` varchar(32) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '发布时间',

`content\_type` varchar(32) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '内容类型 text/image/gif/video',

`source\_id` bigint(20) NULL DEFAULT NULL COMMENT '来源编号',

`create\_time` datetime(0) NULL DEFAULT NULL COMMENT '数据创建时间',

PRIMARY KEY (`content\_id`) USING BTREE

) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE = utf8mb4\_general\_ci ROW\_FORMAT = Dynamic;

-- ----------------------------

-- Records of t\_content

-- ----------------------------

INSERT INTO `t\_content` VALUES (17140110, 2, 58854, 12601190, 4, 4, 61, '高手在民间：牛人手工打造铜斑蛇手杖，整个制作过程太惊人了！', 29, 1, 'http://a.f.budejie.com/share/17140110.html?wx.qq.com', 1, '2018-12-27 15:58:02', 'video', 14159, '2018-12-27 16:01:28');

INSERT INTO `t\_content` VALUES (17157166, 1, 407, 17266360, 4, 18, 50, '会吧你笑死你的😀😀😀👍🏽', 137, 5, 'http://a.f.budejie.com/share/17157166.html?wx.qq.com', 7, '2018-12-26 21:36:01', 'video', 13704, '2018-12-26 22:01:15');

INSERT INTO `t\_content` VALUES (17166511, 1, 53647, 16575883, 4, 9, 46, '总说魔术不靠谱，这次一镜到底服不服？', 75, 6, 'http://a.f.budejie.com/share/17166511.html?wx.qq.com', 0, '2018-12-27 22:46:02', 'video', 14329, '2018-12-27 23:01:09');

INSERT INTO `t\_content` VALUES (17173520, 1, 407, 16350454, 4, 9, 56, '星爷合集', 97, 14, 'http://a.f.budejie.com/share/17173520.html?wx.qq.com', 3, '2018-12-26 20:28:01', 'video', 13679, '2018-12-26 21:01:24');

INSERT INTO `t\_content` VALUES (17183391, 1, 55, 17009384, 4, 52, 4, '牛逼！！', 44, 91, 'http://a.f.budejie.com/share/17183391.html?wx.qq.com', 0, '2019-01-11 20:02:01', 'video', 23279, '2019-01-11 21:02:08');

INSERT INTO `t\_content` VALUES (17188678, 1, 58854, 16506384, 4, 18, 60, '9岁神童堪称小梅西 单骑闯关花式射门无所不能', 92, 3, 'http://a.f.budejie.com/share/17188678.html?wx.qq.com', 4, '2018-12-27 15:17:02', 'video', 14154, '2018-12-27 16:01:27');

INSERT INTO `t\_content` VALUES (17190356, 1, 473, 13134036, 4, 87, 39, '无论新老司机 这种情况必须撞上去！', 229, 5, 'http://a.f.budejie.com/share/17190356.html?wx.qq.com', 7, '2018-12-27 22:02:01', 'video', 14329, '2018-12-27 23:01:09');

INSERT INTO `t\_content` VALUES (17218676, 1, 407, 14206986, 4, 5, 63, '灌篮高手最新大结局', 75, 4, 'http://a.f.budejie.com/share/17218676.html?wx.qq.com', 2, '2018-12-25 19:46:02', 'video', 13054, '2018-12-25 20:01:20');

INSERT INTO `t\_content` VALUES (17238570, 1, 124, 17025371, 4, 5, 32, '此狗已成精，抖眉都会', 73, 6, 'http://a.f.budejie.com/share/17238570.html?wx.qq.com', 2, '2019-01-11 15:17:02', 'video', 23154, '2019-01-11 16:01:58');

INSERT INTO `t\_content` VALUES (17240964, 5, 58191, 7546850, 4, 102, 2129, '要不是我认识四大天王，差点就信了哈哈哈哈', 1776, 24, 'http://a.f.budejie.com/share/17240964.html?wx.qq.com', 252, '2018-11-17 12:24:02', 'video', 4977, '2018-12-12 08:00:23');

INSERT INTO `t\_content` VALUES (17270741, 1, 58191, 17424441, 4, 3, 36, '单身狗的毛病', 57, 1, 'http://a.f.budejie.com/share/17270741.html?wx.qq.com', 0, '2019-01-01 11:49:02', 'video', 17054, '2019-01-01 12:01:34');

INSERT INTO `t\_content` VALUES (17271888, 1, 58191, 17418210, 4, 24, 35, '话说内地护士于台湾护士之间的差距', 173, 12, 'http://a.f.budejie.com/share/17271888.html?wx.qq.com', 9, '2018-12-31 19:11:01', 'video', 16654, '2018-12-31 20:01:43');

INSERT INTO `t\_content` VALUES (17274907, 1, 407, 10455199, 4, 10, 1074, '《师父》里长达7分钟的巷战，咏春传人单挑各大武林高手。短兵相接，剑拔弩张！看着非常过瘾！', 68, 7, 'http://a.f.budejie.com/share/17274907.html?wx.qq.com', 0, '2019-02-09 23:46:02', 'video', 40730, '2019-02-10 00:03:04');

explain是解释计划,说明SQL的执行情况

explain select \* from t\_content where content\_id = 17076710;  
drop index idx\_uid on t\_content;  
create index idx\_uid on t\_content(uid);  
explain select \* from t\_content where uid=16940130;

drop index idx\_uid on t\_content;  
create index idx\_uid on t\_content(uid);

复合索引查询查询条件必须包含左侧列

EXPLAIN select \* from t\_content where uid = 14206986 ;

精准匹配，允许使用Btree索引，

EXPLAIN select \* from t\_content where uid = 14206986 ;

当排序出现了索引左侧列，则允许使用索引排序

索引使用情况

SELECT  
object\_type,object\_schema,object\_name,index\_name,  
count\_read,count\_fetch, count\_insert,  
count\_update,count\_delete  
FROM  
performance\_schema.table\_io\_waits\_summary\_by\_index\_usage  
ORDER BY  
sum\_timer\_wait desc;

索引重新统计

analyze table t\_content;

#optimize 优化表空间,释放表空间

#锁表,一定要在维护期间,否则会造成IO阻塞

optimize table t\_content;

# 5、查询案例分析：

在企业的生产环境中，往往会遇到某个功能或者接口需要很久才能返回结果，我们就应该去确定是不是由于慢查询导致的，定位慢sql有两种解决方案：

1. 查看慢查询日志确定已经执行完的慢查询
2. Show processlist 查看正在执行的慢查询

## 5.1 慢查询日志

如果需要定位到慢查询，一般的方法是通过慢查询日志来查询的，MySQL 的慢查询日志用来记录在 MySQL 中响应时间超过参数 long\_query\_time（单位秒，默认值 10）设置的值并且扫描记录数不小于 min\_examined\_row\_limit（默认值 0）的语句，能够帮我们找到执行完的慢查询，方便我们对这些 SQL 进行优化。

A：确定慢查询日志路径：

show global variables like "datadir"

Varable\_name Value

Datadir /var/lib/mysql/

B： 确定慢日志的文件名：

show global variables like "slow\_query\_log\_file"

Varable\_name Valu

Show\_query\_log\_file /var/lib/mysql/abcd3ef.-show.log

根据上面的查询结果，可以直接查看/var/lib/mysql/6017bde482bf-slow.log 文件获取已经执行完的慢查询

[root@mysqltest ~]# tail -n5 /var/lib/mysql/6017bde482bf-slow.log

Time: 2019-02-21T09:15:06.255554+08:00

User@Host: root[root] @ localhost [] Id: 8591152

Query\_time: 10.000260 Lock\_time: 0.000000 Rows\_sent: 1 Rows\_examined: 0

SET timestamp=1558401306;

select sleep(10);

这里对上方的执行结果详细描述一下

tail -n5：只查看慢查询文件的最后 5 行

Time：慢查询发生的时间

User@Host：客户端用户和 IP

Query\_time：查询时间

Lock\_time：等待表锁的时间

Rows\_sent：语句返回的行数

ows\_examined：语句执行期间从存储引擎读取的行数

上面这种方式是用系统自带的慢查询日志查看的，如果觉得系统自带的慢查询日志不方便查看，可以使用统一日志中心的慢sql日志[，.通过ELK收集mysql慢查询日志](https://www.kancloud.cn/owenwangwen/open-capacity-platform/1132513)，通过filebeat kafka logstash es搜集慢sql。

## 5.2 通过processlist

## 有时慢查询正在执行，已经导致数据库负载偏高了，而由于慢查询还没执行完，因此慢查询日志还看不到任何语句。此时可以使用 show processlist 命令判断正在执行的慢查询。show processlist 显示哪些线程正在运行。如果有 PROCESS 权限，则可以看到所有线程。否则，只能看到当前会话的线程。

## mysql> show processlist`

## https://img.kancloud.cn/cc/3c/cc3c83d861b10b9d6d78477eba0e12be_959x205.png

## Time:表示执行时间

## Info：表示sql语句

## 5.3 使用explain分析慢查询

## 分析sql执行效率是优化sql的重要手段，通过上面讲的两种方法，定位到慢查询语句，我们就要分析sql的执行效率，可通过explain工具来分析慢查询。

## Explain可以获取mysql中sql语句的执行计划，比如语句是否使用了关联查询，是否使用了索引，扫描行数等，可以帮助我们更好的索引和写出更优的Sql。

## 使用方法：在查询语句前加上explain运行就可以了。

## 示例：

## mysql> explain select \* from t1 where b=100;

## id | select type | table | partitions | type | possible\_keys | key | key\_lenth | ref |rows| filterd|extra

## explain的结果各字段解释如下：

## 列名解释id查询编号

## **select\_type**查询类型：显示本行是简单还是复杂查询

## table涉及到的表

## partitions匹配的分区：查询将匹配记录所在的分区。仅当使用 partition 关键字时才显示该列。对于非分区表，该值为 NULL。

## **type**本次查询的表连接类型

## possible\_keys可能选择的索引**key**实际选择的索引

## key\_len被选择的索引长度：一般用于判断联合索引有多少列被选择了

## ref与索引比较的列**rows**预计需要扫描的行数，对

## InnoDB 来说，这个值是估值，并不一定准确filtered按条件筛选的行的百分比

## **Extra**附加信息

## 其中 explain 各列都有各种不同的值，这里介绍几个比较重要列常包含的值：包含 select\_typ、type 和 Extra。

## 5.4 show profile 分析慢查询

## 基本步骤：

## 确定当前的MySQL版本是否支持profile

## 确定profile是否关闭

## 如何关闭开启profile

## 在服务器端发送要执行的SQL

## 查看执行完SQL的query id

## 通过query id 查看sql的每个状态和耗时时间

## 停止profile

### 5.4.1 确定是否支持profile

select @@have\_profiling

### 5.4.2 查看profiling状态

select @@profiling;

0表示关闭，1表示开启，默认是关闭的。  
使用如下命令开启：

mysql> set profiling=1;

### 5.4.3执行要分析的语句

select @@profiling;

set profiling=1 ;

select \* from t1 where b=100;

show profiles

执行完成SQL后再执行：

show profiles；得到profile id。

show profile for query 1 ;

根据profile id查询指定SQL执行详情

### 5.4.4 查看sql的开销

查看指定SQL的CPU开销：show profile cpu for query 1;  
查询指定SQL的内存开销： show profile memory for query 1;

关闭profile

mysql> set profiling=off;

## 5.5 trace 分析sql优化器

通过trace，能够进一步了解为什么优化器选择A执行计划而不是选择B执行计划，或者知道某个排序使用的排序模式，帮助我们更好地理解优化器行为。如果需要使用，先开启 trace，设置格式为 JSON，再执行需要分析的 SQL，最后查看 trace 分析结果(在 information\_schema.OPTIMIZER\_TRACE 中）。  
使用步骤如下：

开启trace分析器  
执行要查询的sql  
查看分析结果  
关闭trace分析器

### 开启trace分析器

set session optimizer\_trace="enabled=on";

### 执行要查询的SQL

select \* from t1 where b=100;

查询分析结果

SELECT \* FROM information\_schema.OPTIMIZER\_TRACE

注意：在返回的steps数组中可以查看详细mysql都干了什么。

# 6总结

## 分享的关于定位慢 SQL 及使用 explain 分析慢 SQL 到这里就结束了。

## 总结如下： 如何定位慢 SQL:一种方法是查看慢查询日志另一种方法是 show process 查看正在执行的 SQL再讲到通过 explain 分析慢 SQL，explain 会返回很多字段，其中 select\_type、type、key、rows、Extra 是重点关注项。

## 最后推荐基于小米 soar 的开源 sql 分析与优化的 WEB 图形化工具，支持 soar 配置的添加、修改、复制，多配置切换，配置的导出、导入与导入功能。