链滴

MySQL 部分重要参数说明

作者：14032
原文链接：https://ld246.com/article/1557389666445
来源网站：链滴

+1 特别推荐

- 掘金小册： MySQL 是怎样运行的：从根儿上理解 MySQL
- 极客时间： MySQL实战45讲

innodb_buffer_pool_size

- 查看系统变量

```
+-------------------------------------+------------------+
| Variable_name                       | Value            |
+-------------------------------------+------------------+
| innodb_buffer_pool_chunk_size       | 134217728(128M)  |
| innodb_buffer_pool_instances        | 1                |
| innodb_buffer_pool_size             | 536870912(512M)  |
+-------------------------------------+------------------+
```

- show engine INNODB status\G;

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 549715968 # innodb_buffer_pool_size 总大小
Dictionary memory allocated 133466
Buffer pool size   32768 # 可容纳缓存页数
Free buffers       31362 # 剩余空闲缓存页
Database pages     1359 # 代表LRU链表中的页的数量，包含young和old两个区域的节点数量
Old database pages 521 # LRU链表old区域的节点数量
Modified db pages  87 # 脏页数量
Pending reads      0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 1171, created 188, written 717
32.52 reads/s, 7.33 creates/s, 32.24 writes/s
Buffer pool hit rate 996 / 1000, # 命中率
young-making rate 0 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 1359, unzip_LRU len: 0
I/O sum[0]:cur[200], unzip sum[0]:cur[0]
```

innodb_log_file_size

```
mysql> show variables like '%innodb_log_file%';
+---------------------------+--------------+
| Variable_name             | Value        |
+---------------------------+--------------+
| innodb_log_file_size      | 50331648(48M)|
| innodb_log_files_in_group | 2            |
```

```
+--------------------------+--------------+
```

● innodb_log_file_size

该参数指定了每个redo日志文件的大小，在MySQL 5.7.25这个版本中的默认值为48MB，

● innodb_log_files_in_group

该参数指定redo日志文件的个数，默认值为2，最大值为100。

sync_binlog

```
mysql> show variables like '%sync_binlog%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| sync_binlog   | 200   |
+---------------+-------+
```

sync_binlog=0 的时候，表示每次提交事务都只 write，不 fsync；

sync_binlog=1 的时候，表示每次提交事务都会执行 fsync；

sync_binlog=N(N>1) 的时候，表示每次提交事务都 write，但累积 N 个事务后才 fsync。

但是，将 sync_binlog 设置为 N，对应的风险是：如果主机发生异常重启，会丢失最近 N 个事务的 b nlog 日志。

innodb_flush_log_at_trx_commit

```
mysql> show variables like '%innodb_flush_log_at_trx_commit%';
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| innodb_flush_log_at_trx_commit | 2     |
+--------------------------------+-------+
```

● 0：当该系统变量值为0时，表示在事务提交时不立即向磁盘中同步redo日志，这个任务是交给台线程做的。 **这样很明显会加快请求处理速度，但是如果事务提交后服务器挂了，后台线程没有及将redo日志刷新到磁盘，那么该事务对页面的修改会丢失。**

● 1：当该系统变量值为1时，表示在事务提交时需要将redo日志同步到磁盘，可以保证事务的持性。1也是innodb_flush_log_at_trx_commit的默认值。

● 2：当该系统变量值为2时，表示在事务提交时需要将redo日志写到操作系统的缓冲区中，但并需要保证将日志真正的刷新到磁盘。 **这种情况下如果数据库挂了，操作系统没挂的话，事务的持久还是可以保证的，但是操作系统也挂了的话，那就不能保证持久性了。**

sort_buffer_size

```
mysql> show variables like 'sort_buffer_size%';
+-----------------+--------+
| Variable_name   | Value  |
```

```
+------------------+--------+
| sort_buffer_size | 262144 |
+------------------+--------+
```

- OPTIMIZER_TRACE 查看完整的优化过程

```
# 开启 OPTIMIZER_TRACE
set optimizer_trace = 'enabled=on';
select * from sys_log order by create_date;
select * from information_schema.optimizer_trace\G;

.....省略
"filesort_summary": {
  "rows": 30815,
  "examined_rows": 30815,
  "number_of_tmp_files": 8, # 表示排序过程中使用的临时文件数
  "sort_buffer_size": 1048560,
  "sort_mode": "<sort_key, rowid>" # 下面做说明
}
```

- 加大 sort_buffer_size 之后，再查看下 OPTIMIZER_TRACE

```
.....省略
"filesort_summary": {
  "rows": 30815,
  "examined_rows": 30815,
  "number_of_tmp_files": 0, # 表示排序直接在内存中完成
  "sort_buffer_size": 20971472,
  "sort_mode": "<sort_key, rowid>" # 下面做说明
}
```

join_buffer_size

```
mysql> show variables like 'join_buffer_size%';
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| join_buffer_size | 262144 |
+------------------+--------+
```

- 嵌套循环连接（Nested-Loop Join）

驱动表只访问一次，但被驱动表却可能被多次访问，访问次数取决于对驱动表执行单表查询后的结果中的记录条数的连接执行方式称之为嵌套循环连接（Nested-Loop Join），这是最简单，也是最笨的一种连接查询算法

- 基于块的嵌套循环连接（Block Nested-Loop Join）

join buffer就是执行连接查询前申请的一块固定大小的内存，先把若干条驱动表结果集中的记录装在个join

buffer中，然后开始扫描被驱动表，每一条被驱动表的记录一次性和join

buffer中的多条驱动表记录做匹配，因为匹配的过程都是在内存中完成的

max_length_for_sort_data

```
mysql> show variables like 'max_length_for_sort_data%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| max_length_for_sort_data | 1024  |
+-------------------------+-------+
```

max_length_for_sort_data，是 MySQL

中专门控制用于排序的行数据的长度的一个参数。它的意思是，如果单行的长度超过这个值，MySQL就认为单行太大，要换一个算法。

- sort_mode
  - sort_key, rowid

这表明排序缓冲区(sort_buffer)只放入排序列和主键ID。按排序键值排序，主键ID用于从表中读取该其他值(回表)。

  - sort_key, additional_fields

这表明排序缓冲区(sort_buffer)包含查询所引用的排序键值和列。按排序键值排序，列值直接从其中取。

  - sort_key, packed_additional_fields

这表明排序过程中对字符串做了紧凑处理，在排序过程中按照实际长度来分配空间。

- 改变查询列数，再查看下 OPTIMIZER_TRACE

```
set optimizer_trace = 'enabled=on';
select id,create_date,type from sys_log order by create_date;
select * from information_schema.optimizer_trace\G;

.....省略
"filesort_summary": {
  "rows": 30815,
  "examined_rows": 30815,
  "number_of_tmp_files": 0,
  "sort_buffer_size": 20971440,
  "sort_mode": "<sort_key, packed_additional_fields>" # 注意看这里
}
```

12