Atitit 重要对mysql 性能优化状态参数

目录

[1. 查看MySQL运行情况 show global status; SHOW STATUS; 1](#_Toc1321)

[1.1. 二、查看INNODB数据库引擎运行状态 SHOW ENGINE INNODB STATUS; 2](#_Toc2312)

[1.2. 附1 SHOW STATUS;关键结果释义 2](#_Toc29651)

[2. 提升性能的建议:  4](#_Toc5294)

[3. 6， 临时表状态 4](#_Toc279)

[4. 7,open table 的情况 6](#_Toc14047)

[5. 9, 查询缓存(query cache) 7](#_Toc11435)

[5.1. 8, 进程使用情况 11](#_Toc11599)

[6. 10,排序使用情况 13](#_Toc20223)

[7. 11.文件打开数(open\_files) 14](#_Toc25098)

[8. 12。 表锁情况 15](#_Toc22469)

[9. 慢查询 18](#_Toc25712)

[10. 4, 连接数 判断够不够用 19](#_Toc17801)

[11. show global status like '%conn%'; 19](#_Toc23262)

[12. explain分析查询 20](#_Toc18225)

[13. ref 23](#_Toc15028)

# 查看MySQL运行情况 show global status; SHOW STATUS;

2, 查看MySQL服务器运行的各种状态值

mysql> show global status;

Mysql> show innodb status ——显示InnoDB存储引擎的状态

Mysql> show processlist ——查看当前SQL执行，包括执行状态、是否锁表等

## 二、查看INNODB数据库引擎运行状态 SHOW ENGINE INNODB STATUS;

三、查看当前正在进行的进程，对于有锁表等情况的排查很有用处   
SHOW PROCESSLIST; 默认显示前100条   
SHOW FULL PROCESSLIST; 显示所有

1. 查看当前已经被打开的表列表   
   SHOW OPEN TABLES;

## 附1 SHOW STATUS;关键结果释义

Aborted\_clients 由于客户没有正确关闭连接已经死掉，已经放弃的连接数量。  
Aborted\_connects 尝试已经失败的MySQL服务器的连接的次数。   
Connections 试图连接MySQL服务器的次数。  
Created\_tmp\_tables 当执行语句时，已经被创造了的隐含临时表的数量。   
Delayed\_insert\_threads 正在使用的延迟插入处理器线程的数量。   
Delayed\_writes 用INSERT DELAYED写入的行数。  
Delayed\_errors 用INSERT DELAYED写入的发生某些错误(可能重复键值)的行数。   
Flush\_commands 执行FLUSH命令的次数。   
Handler\_delete 请求从一张表中删除行的次数。   
Handler\_read\_first 请求读入表中第一行的次数。   
Handler\_read\_key 请求数字基于键读行。   
Handler\_read\_next 请求读入基于一个键的一行的次数。   
Handler\_read\_rnd 请求读入基于一个固定位置的一行的次数。   
Handler\_update 请求更新表中一行的次数。  
Handler\_write 请求向表中插入一行的次数。   
Key\_blocks\_used 用于关键字缓存的块的数量。  
Key\_read\_requests 请求从缓存读入一个键值的次数。  
Key\_reads 从磁盘物理读入一个键值的次数。  
Key\_write\_requests 请求将一个关键字块写入缓存次数。   
Key\_writes 将一个键值块物理写入磁盘的次数。   
Max\_used\_connections 同时使用的连接的最大数目。   
Not\_flushed\_key\_blocks 在键缓存中已经改变但是还没被清空到磁盘上的键块。   
Not\_flushed\_delayed\_rows 在INSERT DELAY队列中等待写入的行的数量。   
Open\_tables 打开表的数量。   
Open\_files 打开文件的数量。   
Open\_streams 打开流的数量(主要用于日志记载）   
Opened\_tables 已经打开的表的数量。   
Questions 发往服务器的查询的数量。  
Slow\_queries 要花超过long\_query\_time时间的查询数量。   
Threads\_connected 当前打开的连接的数量。   
Threads\_running 不在睡眠的线程数量。   
Uptime 服务器工作了多少秒。

# 提升性能的建议:

1.如果opened\_tables太大,应该把my.cnf中的table\_cache变大   
2.如果Key\_reads太大,则应该把my.cnf中key\_buffer\_size变大.可以用Key\_reads/Key\_read\_requests计算出cache失败率   
3.如果Handler\_read\_rnd太大,则你写的SQL语句里很多查询都是要扫描整个表,而没有发挥索引的键的作用   
4.如果Threads\_created太大,就要增加my.cnf中thread\_cache\_size的值.可以用Threads\_created/Connections计算cache命中率   
5.如果Created\_tmp\_disk\_tables太大,就要增加my.cnf中tmp\_table\_size的值,用基于内存的临时表代替基于磁盘的

# 6， 临时表状态

mysql> show global status like 'created\_tmp%';

+-------------------------+---------+

| Variable\_name | Value |

+-------------------------+---------+

| Created\_tmp\_disk\_tables | 4184337 |

| Created\_tmp\_files | 4124 |

| Created\_tmp\_tables | 4215028 |

+-------------------------+---------+

每次创建临时表，Created\_tmp\_tables增加，如果是在磁盘上创建临时表，Created\_tmp\_disk\_tables也增加,Created\_tmp\_files表示MySQL服务创建的临时文件文件数： Created\_tmp\_disk\_tables / Created\_tmp\_tables \* 100% ＝ 99% （理想值<= 25%）

mysql> show variables where Variable\_name in ('tmp\_table\_size', 'max\_heap\_table\_size');

+---------------------+-----------+

| Variable\_name | Value |

+---------------------+-----------+

| max\_heap\_table\_size | 134217728 |

| tmp\_table\_size | 134217728 |

+---------------------+-----------+

需要增加tmp\_table\_size

# 7,open table 的情况

mysql> show global status like 'open%tables%';

+---------------+-------+

| Variable\_name | Value |

+---------------+-------+

| Open\_tables | 1024 |

| Opened\_tables | 1465 |

+---------------+-------+

Open\_tables 表示打开表的数量，Opened\_tables表示打开过的表数量，如果Opened\_tables数量过大，说明配置中 table\_cache(5.1.3之后这个值叫做table\_open\_cache)值可能太小，我们查询一下服务器table\_cache值

mysql> show variables like 'table\_cache';

+---------------+-------+

| Variable\_name | Value |

+---------------+-------+

| table\_cache | 1024 |

+---------------+-------+

Open\_tables / Opened\_tables \* 100% =69% 理想值 （>= 85%） Open\_tables / table\_cache \* 100% = 100% 理想值 (<= 95%)

# 9, 查询缓存(query cache)

mysql> show global status like 'qcache%';

+-------------------------+----------+

| Variable\_name | Value |

+-------------------------+----------+

| Qcache\_free\_blocks | 2226 |

| Qcache\_free\_memory | 10794944 |

| Qcache\_hits | 5385458 |

| Qcache\_inserts | 1806301 |

| Qcache\_lowmem\_prunes | 433101 |

| Qcache\_not\_cached | 4429464 |

| Qcache\_queries\_in\_cache | 7168 |

| Qcache\_total\_blocks | 16820 |

+-------------------------+----------+

Qcache\_free\_blocks：缓存中相邻内存块的个数。数目大说明可能有碎片。FLUSH QUERY CACHE会对缓存中的碎片进行整理，从而得到一个空闲块。 Qcache\_free\_memory：缓存中的空闲内存。 Qcache\_hits：每次查询在缓存中命中时就增大 Qcache\_inserts：每次插入一个查询时就增大。命中次数除以插入次数就是不中比率。 Qcache\_lowmem\_prunes：缓存出现内存不足并且必须要进行清理以便为更多查询提供空间的次数。这个数字最好长时间来看；如果这个数字在不断增长，就表示可能碎片非常严重，或者内存很少。（上面的          free\_blocks和free\_memory可以告诉您属于哪种情况） Qcache\_not\_cached：不适合进行缓存的查询的数量，通常是由于这些查询不是 SELECT 语句或者用了now()之类的函数。 Qcache\_queries\_in\_cache：当前缓存的查询（和响应）的数量。 Qcache\_total\_blocks：缓存中块的数量。  
我们再查询一下服务器关于query\_cache的配置：

mysql> show variables like 'query\_cache%';

+------------------------------+----------+

| Variable\_name | Value |

+------------------------------+----------+

| query\_cache\_limit | 33554432 |

| query\_cache\_min\_res\_unit | 4096 |

| query\_cache\_size | 33554432 |

| query\_cache\_type | ON |

| query\_cache\_wlock\_invalidate | OFF |

+------------------------------+----------+

各字段的解释：  
query\_cache\_limit：超过此大小的查询将不缓存 query\_cache\_min\_res\_unit：缓存块的最小大小 query\_cache\_size：查询缓存大小 query\_cache\_type：缓存类型，决定缓存什么样的查询，示例中表示不缓存 select sql\_no\_cache 查询 query\_cache\_wlock\_invalidate：当有其他客户端正在对MyISAM表进行写操作时，如果查询在query cache中，是否返回cache结果还是等写操作完成再读表获取结果。  
query\_cache\_min\_res\_unit的配置是一柄”双刃剑”，默认是4KB，设置值大对大数据查询有好处，但如果你的查询都是小数据查询，就容易造成内存碎片和浪费。  
查询缓存碎片率 = Qcache\_free\_blocks / Qcache\_total\_blocks \* 100%  
如果查询缓存碎片率超过20%，可以用FLUSH QUERY CACHE整理缓存碎片，或者试试减小query\_cache\_min\_res\_unit，如果你的查询都是小数据量的话。  
查询缓存利用率 = (query\_cache\_size – Qcache\_free\_memory) / query\_cache\_size \* 100%  
查询缓存利用率在25%以下的话说明query\_cache\_size设置的过大，可适当减小；查询缓存利用率在80％以上而且Qcache\_lowmem\_prunes > 50的话说明query\_cache\_size可能有点小，要不就是碎片太多。  
查询缓存命中率 = (Qcache\_hits – Qcache\_inserts) / Qcache\_hits \* 100%  
示例服务器 查询缓存碎片率 ＝ 20.46％，查询缓存利用率 ＝ 62.26％，查询缓存命中率 ＝ 1.94％，命中率很差，可能写操作比较频繁吧，而且可能有些碎片。

## 8, 进程使用情况

mysql> show global status like 'Thread%';

+-------------------+-------+

| Variable\_name | Value |

+-------------------+-------+

| Threads\_cached | 31 |

| Threads\_connected | 239 |

| Threads\_created | 2914 |

| Threads\_running | 4 |

+-------------------+-------+

如果我们在MySQL服务器配置文件中设置了thread\_cache\_size，当客户端断开之后，服务器处理此客户的线程将会缓存起来以响应下一个客户而不是销毁（前提是缓存数未达上限）。Threads\_created表示创建过的线程数，如果发现Threads\_created值过大的话，表明 MySQL服务器一直在创建线程，这也是比较耗资源，可以适当增加配置文件中thread\_cache\_size值，查询服务器 thread\_cache\_size配置：

mysql> show variables like 'thread\_cache\_size';

+-------------------+-------+

| Variable\_name | Value |

+-------------------+-------+

| thread\_cache\_size | 32 |

# 10,排序使用情况

mysql> show global status like 'sort%';

+-------------------+----------+

| Variable\_name | Value |

+-------------------+----------+

| Sort\_merge\_passes | 2136 |

| Sort\_range | 81888 |

| Sort\_rows | 35918141 |

| Sort\_scan | 55269 |

+-------------------+----------+

Sort\_merge\_passes 包括两步。MySQL 首先会尝试在内存中做排序，使用的内存大小由系统变量 Sort\_buffer\_size 决定，如果它的大小不够把所有的记录都读到内存中，MySQL 就会把每次在内存中排序的结果存到临时文件中，等 MySQL 找到所有记录之后，再把临时文件中的记录做一次排序。这再次排序就会增加 Sort\_merge\_passes。实际上，MySQL 会用另一个临时文件来存再次排序的结果，所以通常会看到 Sort\_merge\_passes 增加的数值是建临时文件数的两倍。因为用到了临时文件，所以速度可能会比较慢，增加 Sort\_buffer\_size 会减少 Sort\_merge\_passes 和 创建临时文件的次数。但盲目的增加 Sort\_buffer\_size 并不一定能提高速度，见 How fast can you sort data with MySQL?（引自http://qroom.blogspot.com/2007/09/mysql-select-sort.html）  
另外，增加read\_rnd\_buffer\_size(3.2.3是record\_rnd\_buffer\_size)的值对排序的操作也有一点的好处，参见：http://www.mysqlperformanceblog.com/2007/07/24/what-exactly-is- read\_rnd\_buffer\_size/

# 11.文件打开数(open\_files)

mysql> show global status like 'open\_files';

+---------------+-------+

| Variable\_name | Value |

+---------------+-------+

| Open\_files | 821 |

+---------------+-------+

mysql> show variables like 'open\_files\_limit';

+------------------+-------+

| Variable\_name | Value |

+------------------+-------+

| open\_files\_limit | 65535 |

+------------------+-------+

比较合适的设置：Open\_files / open\_files\_limit \* 100% <= 75％  
正常

# 12。 表锁情况

mysql> show global status like 'table\_locks%';

+-----------------------+---------+

| Variable\_name | Value |

+-----------------------+---------+

| Table\_locks\_immediate | 4257944 |

| Table\_locks\_waited | 25182 |

+-----------------------+---------+

Table\_locks\_immediate 表示立即释放表锁数，Table\_locks\_waited表示需要等待的表锁数，如果 Table\_locks\_immediate / Table\_locks\_waited > 5000，最好采用InnoDB引擎，因为InnoDB是行锁而MyISAM是表锁，对于高并发写入的应用InnoDB效果会好些.

查询表锁争用情况

检查table\_locks\_waited和table\_locks\_immediate状态变量分析

* table\_locks\_immediate : 可以立即获取锁的次数
* table\_locks\_waited : 不能立即获取锁，需要等待锁的次数

13. 表扫描情况

mysql> show global status like 'handler\_read%';

+-----------------------+-----------+

| Variable\_name | Value |

+-----------------------+-----------+

| Handler\_read\_first | 108763 |

| Handler\_read\_key | 92813521 |

| Handler\_read\_next | 486650793 |

| Handler\_read\_prev | 688726 |

| Handler\_read\_rnd | 9321362 |

| Handler\_read\_rnd\_next | 153086384 |

+-----------------------+-----------+

各字段解释参见http://hi.baidu.com/thinkinginlamp/blog/item/31690cd7c4bc5cdaa144df9c.html，

调出服务器完成的查询请求次数：

mysql> show global status like 'com\_select';

+---------------+---------+

| Variable\_name | Value |

+---------------+---------+

| Com\_select | 2693147 |

+---------------+---------+

计算表扫描率：  
表扫描率 ＝ Handler\_read\_rnd\_next / Com\_select  
如果表扫描率超过4000，说明进行了太多表扫描，很有可能索引没有建好，增加read\_buffer\_size值会有一些好处，但最好不要超过8MB。

# 慢查询

show global status like '%slow%';

2，long\_query\_time 当SQL语句执行时间超过此数值时，就会被记录到日志中，建议设置为1或者更短(单位秒)。

3，slow\_query\_log\_file 记录日志的文件名。

4，log\_queries\_not\_using\_indexes 这个参数设置为ON，可以捕获到所有未使用索引的SQL语句，尽管这个SQL语句有可能执行得挺快。

# 4, 连接数 判断够不够用

# show global status like '%conn%';

mysql> show variables like 'max\_connections';

+-----------------+-------+

| Variable\_name | Value |

+-----------------+-------+

| max\_connections | 500 |

+-----------------+-------+

mysql> show global status like 'max\_used\_connections';

+----------------------+-------+

| Variable\_name | Value |

+----------------------+-------+

| Max\_used\_connections | 498 |

+----------------------+-------+

设置的最大连接数是500，而响应的连接数是498  
max\_used\_connections / max\_connections \* 100% = 99.6% （理想值 ≈ 85%）

# explain分析查询

使用 EXPLAIN 关键字可以模拟优化器执行SQL查询语句，从而知道MySQL是如何处理你的SQL语句的。这可以帮你分析你的查询语句或是表结构的性能瓶颈。通过explain命令可以得到:

– 表的读取顺序

– 数据读取操作的操作类型

– 哪些索引可以使用

– 哪些索引被实际使用

– 表之间的引用

– 每张表有多少行被优化器查询

mysql&gt; explain select \* from user;

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+| id | select\_type | table | partitions | type | possible\_keys | key | key\_len | ref | rows | filtered | Extra |

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+

| 1 | SIMPLE | user | NULL | ALL | NULL | NULL | NULL | NULL | 85 | 100.00 | NULL |

+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+-------+1 row in set, 1 warning (0.01 sec)

#### **EXPLAIN字段：**

id: 查询序号即为sql语句执行的顺序

select\_type

select类型，它有以下几种值

2.1 simple 它表示简单的select,没有union和子查询

2.2 primary 最外面的select,在有子查询的语句中，最外面的select查询就是primary,上图中就是这样

2.3 union union语句的第二个或者说是后面那一个.现执行一条语句，explain

Table：显示这一行的数据是关于哪张表的

possible\_keys：显示可能应用在这张表中的索引。如果为空，没有可能的索引。可以为相关的域从WHERE语句中选择一个合适的语句

key：实际使用的索引。如果为NULL，则没有使用索引。MYSQL很少会选择优化不足的索引，此时可以在SELECT语句中使用USE INDEX（index）来强制使用一个索引或者用IGNORE INDEX（index）来强制忽略索引

key\_len：使用的索引的长度。在不损失精确性的情况下，长度越短越好

ref：显示索引的哪一列被使用了，如果可能的话，是一个常数

rows：MySQL认为必须检索的用来返回请求数据的行数

type：这是最重要的字段之一，显示查询使用了何种类型。从最好到最差的连接类型为system、const、eq\_reg、ref、range、index和ALL

Extra：关于MYSQL如何解析查询的额外信息，主要有以下几种

Extra:参数说明

using index：只用到索引,可以避免访问表.

using where：使用到where来过虑数据. 不是所有的where clause都要显示using where. 如以=方式访问索引.

using tmporary：用到临时表

using filesort：用到额外的排序. (当使用order by v1,而没用到索引时,就会使用额外的排序)

range checked for eache record(index map:N)：没有好的索引.

type：参数说明

system、const：可以将查询的变量转为常量. 如id=1; id为 主键或唯一键.

eq\_ref：访问索引,返回某单一行的数据.(通常在联接时出现，查询使用的索引为主键或惟一键)

ref：访问索引,返回某个值的数据.(可以返回多行) 通常使用=时发生

range：这个连接类型使用索引返回一个范围中的行，比如使用>或<查找东西，并且该字段上建有索引时发生的情况(注:不一定好于index)

index：以索引的顺序进行全表扫描，优点是不用排序,缺点是还要全表扫描

ALL：全表扫描，应该尽量避免

#### **profiling分析查询 le**

通过慢日志查询可以知道哪些SQL语句执行效率低下，通过explain我们可以得知SQL语句的具体执行情况，索引使用等，还可以结合show命令查看执行状态。

如果觉得explain的信息不够详细，可以同通过profiling命令得到更准确的SQL执行消耗系统资源的信息。

profiling默认是关闭的。可以通过以下语句查看 select @@profiling;

打开功能： mysql>set profiling=1; 执行需要测试的sql 语句：

开启后执行了sql语句 就可以通过show profiles;命令查看执行时间

mysql&gt; show profiles;

+----------+------------+---------------------------+

| Query\_ID | Duration | Query |

+----------+------------+---------------------------+

| 1 | 0.00417000 | select \* from user |

| 2 | 0.00214700 | select count(\*) from user |

+----------+------------+---------------------------+2 rows in set, 1 warning (0.00 sec)

mysql> show profiles\G; 可以得到被执行的SQL语句的时间和ID

mysql>show profile for query 1; 得到对应SQL语句执行的详细信息

mysql&gt; show profile for query 1;

+----------------------+----------+| Status | Duration |

+----------------------+----------+

| starting | 0.001587 || checking permissions | 0.000351 |

| Opening tables | 0.000015 || init | 0.000021 |

| System lock | 0.000264 || optimizing | 0.000014 |

| statistics | 0.000014 || preparing | 0.000012 |

| executing | 0.000003 || Sending data | 0.001485 |

| end | 0.000018 || query end | 0.000120 |

| closing tables | 0.000026 || freeing items | 0.000157 |

| cleaning up | 0.000083 |

+----------------------+----------+15 rows in set, 1 warning (0.00 sec)

# ref

Mysql运行状态查询命令及调优详解\_Leeon的博客-CSDN博客\_查看mysql状态