Mysql性能优化参数设置

# 全局缓冲

Mysql启动时就需要分配并且总是存在的全局缓冲

**key\_buffer\_size（**参数只对MyISAM有影响**）：**

但是内部的临时磁盘表是MyISAM表，也要使用该值

1、 设置key\_buffer\_size variable。MyISAN最主要的cache设置，用于缓存MyISAM表格的index数据，该参数只对MyISAM有影响。通常在只使用MyISAM的Server中设置25-33%的内存大小。

2、 可以使用几个不同的Key Caches（对一些hot data）。

a) SET GLOBAL test.key\_buffer\_size=512\*1024;

b) CACHE INDEX t1.i1, t2.i1, t3 IN test;

3、 Preload index到Cache中可以提高查询速度。因为preloading index是顺序的，所以非常快。

a) LOAD INDEX INTO CACHE t1, t2 IGNORE LEAVES；

步骤：

1、show variables like 'key\_buffer\_size'; （/1024/1024=MB）

2、show global status like 'key\_read%';（key\_cache\_miss\_rate在0.1%以下都很好）

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

| Variable\_name | Value |

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

| Key\_read\_requests | 27813678764 |

| Key\_reads | 6798830 |

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

一共有27813678764个索引读取请求，有6798830个请求在内存中没有找到直接从硬盘读取索引，计算索引未命中缓存的概率：

key\_cache\_miss\_rate = Key\_reads / Key\_read\_requests \* 100%=0.024%

3、show global status like 'key\_blocks\_u%';

+------------------------+-------------+  
| Variable\_name | Value |  
+------------------------+-------------+  
| Key\_blocks\_unused | 0 |  
| Key\_blocks\_used | 413543 |  
+------------------------+-------------+

Key\_blocks\_unused表示未使用的缓存簇(blocks)数，

Key\_blocks\_used表示曾经用到的最大的blocks数，

要么增加key\_buffer\_size，要么就是过渡索引了，把缓存占满了。

比较理想的设置：  
Key\_blocks\_used / (Key\_blocks\_unused + Key\_blocks\_used) \* 100% ≈ 80%

**innodb\_buffer\_pool\_size（**每 MB 增加约 3000 的句柄数占用**）：**

这是InnoDB最重要的设置，对InnoDB性能有决定性的影响。默认的设置只有8M，所以默认的数据库设置下面InnoDB性能很差。在只有InnoDB存储引擎的数据库服务器上面，可以设置60-80%的内存。更精确一点，在内存容量允许的情况下面设置比InnoDB tablespaces大10%的内存大小

步骤：

1、show variables like 'innodb\_buffer\_pool\_size';

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

| Variable\_name           | Value     |

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

| innodb\_buffer\_pool\_size | 134217728 |

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

2、show status like 'Innodb\_buffer\_pool\_read%';

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

| Variable\_name                         | Value |

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

| Innodb\_buffer\_pool\_read\_ahead\_rnd     | 0     |

| Innodb\_buffer\_pool\_read\_ahead         | 0     |

| Innodb\_buffer\_pool\_read\_ahead\_evicted | 0     |

| Innodb\_buffer\_pool\_read\_requests      | 95811 |

| Innodb\_buffer\_pool\_reads              | 1742  |

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

计算缓存命中率： (Innodb\_buffer\_pool\_read\_requests - Innodb\_buffer\_pool\_reads) / Innodb\_buffer\_pool\_read\_requests \* 100%

**innodb\_additional\_mem\_pool\_size（**调整的可能不是太多**）：**

设置了InnoDB存储引擎用来存放数据字典信息以及一些内部数据结构的内存空间大小，所以当我们一个MySQL Instance中的数据库对象非常多的时候，是需要适当调整该参数的大小以确保所有数据都能存放在内存中提高访问效率的。

当过小的时候，MySQL 会记录 Warning 信息到数据库的 error log 中，这时候你就知道该调整这个参数大小了

步骤：

1、 show variables like 'innodb\_additional\_mem\_pool\_size';

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

| Variable\_name                   | Value   |

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

| innodb\_additional\_mem\_pool\_size | 8388608 |

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

**innodb\_log\_file\_size：**

该参数决定了recovery speed。太大的话recovery就会比较慢，太小了影响查询性能，一般取256M可以兼顾性能和recovery的速度

**innodb\_log\_buffer\_size（**一般8M**）：**

InnoDB 存储引擎的事务日志所使用的缓冲区

InnoDB 在写事务日志的时候，为了提高性能，也是先将信息写入 Innodb Log Buffer 中，当满足 innodb\_flush\_log\_trx\_commit 参数所设置的相应条件（或者日志缓冲区写满）之后，才会将日志写到文件（或者同步到磁盘）中

innodb\_flush\_logs\_at\_trx\_commit=2： 该参数设定了事务提交时内存中log信息的处理。

1、=0时， 日志缓冲每秒一次地被写到日志文件，并且对日志文件做到磁盘操作的刷新。任何mysqld进程的崩溃会删除崩溃前最后一秒的事务

2、=1时，在每个事务提交时，日志缓冲被写到日志文件，对日志文件做到磁盘操作的刷新。Truly ACID。速度慢。

3、=2时，在每个事务提交时，日志缓冲被写到文件，但不对日志文件做到磁盘操作的刷新。只有操作系统崩溃或掉电才会删除最后一秒的事务，不然不会丢失事务。此外，每秒会有一次文件系统到磁盘同步操作

步骤：

1、show variables like 'innodb\_log\_buffer\_size'

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

| Variable\_name          | Value   |

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

| innodb\_log\_buffer\_size | 8388608 |

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

**query\_cache\_size：**

用来缓存 MySQL 中的 ResultSet，仅仅只能针对select语句。

如果该语句满足Query Cache的要求(未显式说明不允许使用Query Cache，或者已经显式申明需要使用Query Cache)，MySQL 会直接根据预先设定好的HASH算法将接受到的select语句以字符串方式进行hash，然后到Query Cache 中直接查找是否已经缓存。

Query Cache 也有一个致命的缺陷，那就是当某个表的数据有任何任何变化，都会导致所有引用了该表的select语句在Query Cache 中的缓存数据失效。所以，当我们的数据变化非常频繁的情况下，使用Query Cache 可能会得不偿失。

最为关键的是 query\_cache\_size 和 query\_cache\_type ，前者设置用于缓存 ResultSet 的内存大小，后者设置在何场景下使用 Query Cache。

步骤：

1、show variables like 'query\_cache%'

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

| Variable\_name                | Value   |

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

| query\_cache\_limit            | 1048576 |

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

| query\_cache\_size             | 1048576 |

| query\_cache\_type             | OFF     |

| query\_cache\_wlock\_invalidate | OFF     |

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

query\_cache\_type：

1、=0（OFF），完全不使用query cache

2、=1（ON），除显式要求不使用query cache(使用sql\_no\_cache)之外的所有的select都使用query cache

3、=2（DEMAND），只有显示要求才使用query cache(使用sql\_cache)

2、show status like 'Qcache%';

计算Query Cache的命中率：Qcache\_hits/(Qcache\_hits+Qcache\_inserts)\*100

**binlog\_cache\_size（通常2～4M）：**

用于在打开了二进制日志(binlog)记录功能的环境，是 MySQL 用来提高binlog的记录效率而设计的一个用于短时间内临时缓存binlog数据的内存区域

如果我们的数据库中没有什么大事务，写入也不是特别频繁，2MB～4MB是一个合适的选择。但是如果我们的数据库大事务较多，写入量比较大，可与适当调高binlog\_cache\_size

步骤：

1、show variables like 'binlog%';

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

| Variable\_name                           | Value     |

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

| binlog\_cache\_size                       | 32768     |

| binlog\_checksum                         | CRC32     |

| binlog\_direct\_non\_transactional\_updates | OFF       |

| binlog\_format                           | STATEMENT |

| binlog\_max\_flush\_queue\_time             | 0         |

| binlog\_order\_commits                    | ON        |

| binlog\_row\_image                        | FULL      |

| binlog\_rows\_query\_log\_events            | OFF       |

| binlog\_stmt\_cache\_size                  | 32768     |

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

2、show status like 'binlog%'；

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

| Variable\_name              | Value |

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

| Binlog\_cache\_disk\_use      | 0     |

| Binlog\_cache\_use           | 37    |

| Binlog\_stmt\_cache\_disk\_use | 0     |

| Binlog\_stmt\_cache\_use      | 0     |

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

通过binlog\_cache\_use 以及 binlog\_cache\_disk\_use来分析设置的binlog\_cache\_size是否足够

# 链接使用的缓冲

对数据表做复杂的操作例如扫描、排序或者需要临时表，则需分配大约

**read\_buffer\_size,**

**sort\_buffer\_size,**

**read\_rnd\_buffer\_size,**

**tmp\_table\_size**

        不过它们只是在需要的时候才分配，并且在那些操作做完之后就释放了。

# Innodb优化

**１.内存利用方面：**首先介绍一个Innodb最重要的参数：  
**innodb\_buffer\_pool\_size**  
　　这个参数和MyISAM的key\_buffer\_size有相似之处，但也是有差别的。这个参数主要缓存innodb表的索引，数据，插入数据时的缓冲。为Innodb加速优化首要参数。  
　　该参数分配内存的原则：这个参数默认分配只有8M，可以说是非常小的一个值。如果是一个专用ＤＢ服务器，那么他可以占到内存的70%-80%。这个参数不能动态更改，所以分配需多考虑。分配过大，会使Swap占用过多，致使Mysql的查询特慢。如果你的数据比较小，那么可分配是你的数据大小＋１０％左右做为这个参数的值。例如：数据大小为５０Ｍ,那么给这个值分配innodb\_buffer\_pool\_size＝６４Ｍ  
设置方法：  
**innodb\_buffer\_pool\_size=4G**  
这个参数分配值的使用情况可以根据**show innodb status\G;**中的  
**----------------------  
BUFFER POOL AND MEMORY  
----------------------  
Total memory allocated 4668764894;**   
去确认使用情况。

第二个：  
**innodb\_additional\_mem\_pool：**  
作用：用来存放Innodb的内部目录  
这个值不用分配太大，系统可以自动调。不用设置太高。通常比较大数据设置16Ｍ够用了，如果表比较多，可以适当的增大。如果这个值自动增加，会在error log有中显示的。  
分配原则：  
用**show innodb status\G;**去查看运行中的ＤＢ是什么状态（参考BUFFER POOL AND MEMORY段中），然后可以调整到适当的值。  
----------------------  
BUFFER POOL AND MEMORY  
----------------------  
Total memory allocated 4668764894; in additional pool allocated 16777216  
参考：in additional pool allocated 16777216  
根据你的参数情况，可以适当的调整。  
设置方法：  
**innodb\_additional\_mem\_pool=16M**

２.关于日值方面：  
**innodb\_log\_file\_size**  
作用：指定日值的大小  
分配原则：几个日值成员大小加起来差不多和你的innodb\_buffer\_pool\_size相等。上限为每个日值上限大小为4G.一般控制在几个ＬＯＧ文件相加大小在２Ｇ以内为佳。具体情况还需要看你的事务大小，数据大小为依据。  
说明：这个值分配的大小和数据库的写入速度，事务大小，异常重启后的恢复有很大的关系。  
设置方法：  
**innodb\_log\_file\_size=256M**

**innodb\_log\_files\_in\_group**作用：指定你有几个日值组。  
分配原则：　一般我们可以用２－３个日值组。默认为两个。  
设置方法：  
**innodb\_log\_files\_in\_group=3**

**innodb\_log\_buffer\_size：**作用：事务在内存中的缓冲。  
分配原则：控制在2-8M.这个值不用太多的。他里面的内存一般一秒钟写到磁盘一次。具体写入方式和你的事务提交方式有关。在Ｏｒａｃｌｅ等数据库了解这个，一般最大指定为３Ｍ比较合适。  
参考：Innodb\_os\_log\_written(show global status 可以拿到)  
如果这个值增长过快，可以适当的增加innodb\_log\_buffer\_size  
另外如果你需要处理大理的ＴＥＸＴ，或是ＢＬＯＢ字段，可以考虑增加这个参数的值。  
设置方法：  
**innodb\_log\_buffer\_size=3M**

**innodb\_flush\_logs\_at\_trx\_commit**  
作用：控制事务的提交方式  
分配原则：这个参数只有３个值，０，１，２请确认一下自已能接受的级别。默认为１，主库请不要更改了。  
性能更高的可以设置为０或是２，但会丢失一秒钟的事务。  
**说明：**  
这个参数的设置对Ｉｎｎｏｄｂ的性能有很大的影响，所以在这里给多说明一下。  
当这个值为1时：innodb 的事务LOG在每次提交后写入日值文件，并对日值做刷新到磁盘。这个可以做到不丢任何一个事务。  
当这个值为2时：在每个提交，日志缓冲被写到文件，但不对日志文件做到磁盘操作的刷新,在对日志文件的刷新在值为2的情况也每秒发生一次。但需要注意的是，由于进程调用方面的问题，并不能保证每秒１００％的发生。从而在性能上是最快的。但操作系统崩溃或掉电才会删除最后一秒的事务。  
当这个值为0时：日志缓冲每秒一次地被写到日志文件，并且对日志文件做到磁盘操作的刷新，但是在一个事务提交不做任何操作。mysqld进程的崩溃会删除崩溃前最后一秒的事务。

从以上分析，当这个值不为１时，可以取得较好的性能，但遇到异常会有损失，所以需要根据自已的情况去衡量。

设置方法：  
**innodb\_flush\_logs\_at\_trx\_commit=1**

3.　文件IO分配，空间占用方面  
**innodb\_file\_per\_table**  
作用：使每个Innodb的表，有自已独立的表空间。如删除文件后可以回收那部分空间。  
分配原则：只有使用不使用。但ＤＢ还需要有一个公共的表空间。  
设置方法：  
**innodb\_file\_per\_table=1**

**innodb\_file\_io\_threads**  
作用：文件读写ＩＯ数，这个参数只在Windows上起作用。在LINUX上只会等于４  
设置方法：  
**innodb\_file\_io\_threads=4**

**innodb\_open\_files**  
作用：限制Innodb能打开的表的数据。  
分配原则：如果库里的表特别多的情况，请增加这个。这个值默认是３００。  
设置方法：  
**innodb\_open\_files=800**请适当的增加table\_cache

**4. 其它相关参数**  
这里说明一个比较重要的参数：  
**innodb\_flush\_method**  
作用：Ｉｎｎｏｄｂ和系统打交道的一个ＩＯ模型  
分配原则：Windows不用设置。  
Ｕｎｉｘ可以设置：fsync() or O\_SYNC/O\_DSYNC  
如果系统可以禁止系统的Ｃａｃｈｅ那就把他禁了。  
Ｌｉｎｕｘ可以选择：O\_DIRECT   
直接写入磁盘，禁止系统Ｃａｃｈｅ了  
设置方法：  
**innodb\_flush\_method=O\_DIRECT**

**innodb\_max\_dirty\_pages\_pct**  
作用：控制Innodb的脏页在缓冲中在那个百分比之下，值在范围1-100,默认为90.  
这个参数的另一个用处：当Ｉｎｎｏｄｂ的内存分配过大，致使Ｓｗａｐ占用严重时，可以适当的减小调整这个值，使达到Ｓｗａｐ空间释放出来。建义：这个值最大在９０％，最小在１５％。太大，缓存中每次更新需要致换数据页太多，太小，放的数据页太小，更新操作太慢。  
设置方法：  
innodb\_max\_dirty\_pages\_pct＝90  
动态更改需要有Ｓｕｐｅｒ权限：  
set global innodb\_max\_dirty\_pages\_pct=50;