**MySQL优化说明**

### 1、删除多余数据库

删除多余的数据库，可将其备份成文件的形式。

### 2、MySQL优化：

一、慢日志查询，将long\_query\_time=2大于2秒的查询记录到.log文件中。

在my.ini的[mysqld]添加如下语句：

slow\_query\_log=ON  
 log-slow-queries = E:\web\mysql\log\mysqlslowquery.log  
 long\_query\_time = 2

二、配置my.ini文件优化

[mysqld] ：

port = 3306 serverid = 1 socket = /tmp/mysql.sock skip-locking #避免MySQL的外部锁定，减少出错几率增强稳定性。

skip-name-resolve #禁止MySQL对外部连接进行DNS解析，使用这一选项可以消除MySQL进行DNS解析的时间。但需要注意，如果开启该选项，则所有远程主机连接授权都要使用IP地址方式，否则MySQL将无法正常处理连接请求!

back\_log = 384 #back\_log 参数的值指出在MySQL暂时停止响应新请求之前的短时间内多少个请求可以被存在堆栈中。 如果系统在一个短时间内有很多连接，则需要增大该参数的值，该参数值指定到来的TCP/IP连接的侦听队列的大小。不同的操作系统在这个队列大小上有它自 己的限制。 试图设定back\_log高于你的操作系统的限制将是无效的。默认值为50。对于Linux系统推荐设置为小于512的整数。

key\_buffer\_size = 256M #key\_buffer\_size指定用于索引的缓冲区大小，增加它可得到更好的索引处理性能。对于内存在4GB左右的服务器该参数可设置为256M或384M。注意：该参数值设置的过大反而会是服务器整体效率降低!

max\_allowed\_packet = 4M

thread\_stack = 256K

table\_cache = 128K

sort\_buffer\_size = 6M #查询排序时所能使用的缓冲区大小。注意：该参数对应的分配内存是每连接独占，如果有100个连接，那么实际分配的总共排序缓冲区大小为100 × 6 = 600MB。所以，对于内存在4GB左右的服务器推荐设置为6-8M。

read\_buffer\_size = 4M #读查询操作所能使用的缓冲区大小。和sort\_buffer\_size一样，该参数对应的分配内存也是每连接独享。

join\_buffer\_size = 8M #联合查询操作所能使用的缓冲区大小，和sort\_buffer\_size一样，该参数对应的分配内存也是每连接独享。

myisam\_sort\_buffer\_size = 64M

table\_cache = 512

thread\_cache\_size = 64

query\_cache\_size = 64M # 指定MySQL查询缓冲区的大小。可以通过在MySQL控制台观察，如果Qcache\_lowmem\_prunes的值非常大，则表明经常出现缓冲不够的 情况;如果Qcache\_hits的值非常大，则表明查询缓冲使用非常频繁，如果该值较小反而会影响效率，那么可以考虑不用查询缓 冲;Qcache\_free\_blocks，如果该值非常大，则表明缓冲区中碎片很多。

tmp\_table\_size = 256M max\_connections = 768 #指定MySQL允许的最大连接进程数。如果在访问论坛时经常出现Too Many Connections的错误提 示，则需要增大该参数值。

max\_connect\_errors = 10000000 wait\_timeout = 10 #指定一个请求的最大连接时间，对于4GB左右内存的服务器可以设置为5-10。

hread\_concurrency = 8 #该参数取值为服务器逻辑CPU数量\*2，在本例中，服务器有2颗物理CPU，而每颗物理CPU又支持H.T超线程，所以实际取值为4\*2=8 skip-networking #开启该选项可以彻底关闭MySQL的TCP/IP连接方式，如果WEB服务器是以远程连接的方式访问MySQL数据库服务器则不要开启该选项!否则将无法正常连接!

table\_cache=1024 #物理内存越大,设置就越大.默认为2402,调到512-1024最佳

innodb\_additional\_mem\_pool\_size=4M #默认为2M

innodb\_flush\_log\_at\_trx\_commit=1 #设置为0就是等到innodb\_log\_buffer\_size列队满后再统一储存

innodb\_log\_buffer\_size=2M #默认为1M

innodb\_thread\_concurrency=8 #你的服务器CPU有几个就设置为几,建议用默认一般为8

key\_buffer\_size=256M #默认为218，调到128最佳

tmp\_table\_size=64M #默认为16M，调到64-256最挂

read\_buffer\_size=4M #默认为64K

read\_rnd\_buffer\_size=16M #默认为256K

sort\_buffer\_size=32M #默认为256K

thread\_cache\_size=120 #默认为60

query\_cache\_size=32M

三、优化mysql数据库性能的十个参数

(1)、max\_connections：

允许的同时客户的数量。增加该值增加 mysqld 要求的文件描述符的数量。这个数字应该增加，否则，你将经常看到 too many connections 错误。 默认数值是100，我把它改为1024 。

(2)、record\_buffer：

每个进行一个顺序扫描的线程为其扫描的每张表分配这个大小的一个缓冲区。如果你做很多顺序扫描，你可能想要增加该值。默认数值是131072(128k)，我把它改为16773120 (16m)

(3)、key\_buffer\_size：

索引块是缓冲的并且被所有的线程共享。key\_buffer\_size是用于索引块的缓冲区大小，增加它可得到更好处理的索引(对所有读和多重写)，到你能负担得起那样多。如果你使它太大，系统将开始换页并且真的变慢了。默认数值是8388600(8m)，我的mysql主机有2gb内存，所以我把它改为 402649088(400mb)。

4)、back\_log：

要求 mysql 能有的连接数量。当主要mysql线程在一个很短时间内得到非常多的连接请求，这就起作用，然后主线程花些时间(尽管很短)检查连接并且启动一个新线程。

back\_log 值指出在mysql暂时停止回答新请求之前的短时间内多少个请求可以被存在堆栈中。只有如果期望在一个短时间内有很多连接，你需要增加它，换句话说，这值对到来的tcp/ip连接的侦听队列的大小。你的操作系统在这个队列大小上有它自己的限制。试图设定back\_log高于你的操作系统的限制将是无效的。

当你观察你的主机进程列表，发现大量 264084 | unauthenticated user | xxx.xxx.xxx.xxx | null | connect | null | login | null 的待连接进程时，就要加大 back\_log 的值了。默认数值是50，我把它改为500。

(5)、interactive\_timeout：

服务器在关闭它前在一个交互连接上等待行动的秒数。一个交互的客户被定义为对 mysql\_real\_connect()使用 client\_interactive 选项的客户。 默认数值是28800，我把它改为7200。

(6)、sort\_buffer：

每个需要进行排序的线程分配该大小的一个缓冲区。增加这值加速order by或group by操作。默认数值是2097144(2m)，我把它改为 16777208 (16m)。

(7)、table\_cache：

为所有线程打开表的数量。增加该值能增加mysqld要求的文件描述符的数量。mysql对每个唯一打开的表需要2个文件描述符。默认数值是64，我把它改为512。

(8)、thread\_cache\_size：

可以复用的保存在中的线程的数量。如果有，新的线程从缓存中取得，当断开连接的时候如果有空间，客户的线置在缓存中。如果有很多新的线程，为了提高性能可以这个变量值。通过比较 connections 和 threads\_created 状态的变量，可以看到这个变量的作用。我把它设置为 80。

(9)mysql的搜索功能

用mysql进行搜索，目的是能不分大小写，又能用中文进行搜索

只需起动mysqld时指定 --default-character-set=gb2312

(10)、wait\_timeout：

服务器在关闭它之前在一个连接上等待行动的秒数。 默认数值是28800，我把它改为7200。

### 3、mysqlcheck使用

如命令行：mysqlcheck -A -r -u root -p

输入密码

返回databases ok 修复成功

返回The storage engine for the table doesn't support repair 不需要处理的意思

### 4、例子

2G内存,针对站多,抗压型的设置,最佳:

table\_cache=1024 物理内存越大,设置就越大.默认为2402,调到512-1024最佳

innodb\_additional\_mem\_pool\_size=4M 默认为2M

innodb\_flush\_log\_at\_trx\_commit=1

(设置为0就是等到innodb\_log\_buffer\_size列队满后再统一储存,默认为1)

innodb\_log\_buffer\_size=2M 默认为1M

innodb\_thread\_concurrency=8 你的服务器CPU有几个就设置为几,建议用默认一般为8

key\_buffer\_size=256M 默认为218 调到128最佳

tmp\_table\_size=64M 默认为16M 调到64-256最挂

read\_buffer\_size=4M 默认为64K

read\_rnd\_buffer\_size=16M 默认为256K

sort\_buffer\_size=32M 默认为256K

max\_connections=1024 默认为1210

thread\_cache\_size=120 默认为60

query\_cache\_size=64M

一般:

table\_cache=512

innodb\_additional\_mem\_pool\_size=8M

innodb\_flush\_log\_at\_trx\_commit=0

innodb\_log\_buffer\_size=4M

innodb\_thread\_concurrency=8

key\_buffer\_size=128M

tmp\_table\_size=128M

read\_buffer\_size=4M

read\_rnd\_buffer\_size=16M

sort\_buffer\_size=32M

max\_connections=1024

### 5、优化Mysql数据库的8个方法

1、创建索引

对于查询占主要的应用来说，索引显得尤为重要。很多时候性能问题很简单的就是因为我们忘了添加索引而造成的，或者说没有添加更为有效的索引导致。如果不加索引的话，那么查找任何哪怕只是一条特定的数据都会进行一次全表扫描，如果一张表的数据量很大而符合条件的结果又很少，那么不加索引会引起致命的性能下降。但是也不是什么情况都非得建索引不可，比如性别可能就只有两个值，建索引不仅没什么优势，还会影响到更新速度，这被称为过度索引。

2、复合索引

比如有一条语句是这样的：select \* from users where area='beijing' and age=22;

如果我们是在area和age上分别创建单个索引的话，由于mysql查询每次只能使用一个索引，所以虽然这样已经相对不做索引时全表扫描提高了很多效率，但是如果在area、age两列上创建复合索引的话将带来更高的效率。如果我们创建了(area, age, salary)的复合索引，那么其实相当于创建了(area,age,salary)、(area,age)、(area)三个索引，这被称为最佳左前缀特性。因此我们在创建复合索引时应该将最常用作限制条件的列放在最左边，依次递减。

3、索引不会包含有NULL值的列

只要列中包含有NULL值都将不会被包含在索引中，复合索引中只要有一列含有NULL值，那么这一列对于此复合索引就是无效的。所以我们在数据库设计时不要让字段的默认值为NULL。

4、使用短索引

对串列进行索引，如果可能应该指定一个前缀长度。例如，如果有一个CHAR(255)的 列，如果在前10 个或20 个字符内，多数值是惟一的，那么就不要对整个列进行索引。短索引不仅可以提高查询速度而且可以节省磁盘空间和I/O操作。

5、排序的索引问题

mysql查询只使用一个索引，因此如果where子句中已经使用了索引的话，那么order by中的列是不会使用索引的。因此数据库默认排序可以符合要求的情况下不要使用排序操作；尽量不要包含多个列的排序，如果需要最好给这些列创建复合索引。

6、like语句操作

一般情况下不鼓励使用like操作，如果非使用不可，如何使用也是一个问题。like “%aaa%” 不会使用索引而like “aaa%”可以使用索引。

7、不要在列上进行运算

select \* from users where YEAR(adddate)<2007;

将在每个行上进行运算，这将导致索引失效而进行全表扫描，因此我们可以改成

select \* from users where adddate<‘2007-01-01';

8、不使用NOT IN和<>操作

NOT IN和<>操作都不会使用索引将进行全表扫描。NOT IN可以NOT EXISTS代替，id<>3则可使用id>3 or id<3来代替。