下面先说我的服务器的硬件以及论坛情况，  
CPU: 2颗四核Intel Xeon 2.00GHz  
内存: 4GB DDR  
硬盘: SCSI 146GB  
论坛：在线会员 一般在 5000 人左右 � 最高记录是 13264.  
下面，我们根据以上硬件配置结合一份已经做过一次优化的my.cnf进行分析说明：有些参数可能还得根据论坛的变化情况以及程序员的程序进行再调整。

[mysqld]

# 一般配置选项

port            = 3306

 # mysql服务端默认监听(listen on)的TCP/IP端口

socket = @MYSQL\_UNIX\_ADDR@

skip-external-locking

#不使用系统锁定，要使用myisamchk,必须关闭服务器

connect\_timeout = 20

#连接超时之前的最大秒数,在Linux平台上，该超时也用作等待服务器首次回应的时间

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。

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。

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

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! 建议增加该参数

back\_log = 300

# back\_log 是操作系统在监听队列中所能保持的连接数,队列保存了在 MySQL 连接管理器线程处理之前的连接.

# 如果你有非常高的连接率并且出现 “connection refused” 报错,你就应该增加此处的值.

# 检查你的操作系统文档来获取这个变量的最大值.如果将back\_log设定到比你操作系统限制更高的值,将会没有效果

back\_log = 600

# 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。  
# 如果期望在一个短时间内有很多连接，你需要增加它。也就是说，如果MySQL的连接数据达到max\_connections时，新来的请求将会被存在堆栈中，  
# 以等待某一连接释放资源，该堆栈的数量即back\_log，如果等待连接的数量超过back\_log，将不被授予连接资源。  
# 另外，这值（back\_log）限于您的操作系统对到来的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（可以检查你的OS文档找出这个变量的最大值），试图设定back\_log高于你的操作系统的限制将是无效的。

back\_log = 500

#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。默认值为50。对于Linux系统推荐设置为小于512的整数。

back\_log = 500

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。

#skip-networking

# 不在 TCP/IP 端口上进行监听.如果所有的进程都是在同一台服务器连接到本地的 mysqld,这样设置将是增强安全的方法，

# 所有 mysqld 的连接都是通过 Unix Sockets 或者命名管道进行的.注意在 Windows下如果没有打开命名管道选项而只是用此项

# (通过 “enable-named-pipe” 选项) 将会导致 MySQL 服务没有任何作用!

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!

max\_connections = 3000

# MySQL 服务所允许的同时会话数的上限，其中一个连接将被 SUPER 权限保留作为管理员登录.即便已经达到了连接数的上限.

max\_connections = 1000

# MySQL的最大连接数，如果服务器的并发连接请求量比较大，建议调高此值，以增加并行连接数量，当然这建立在机器能支撑的情况下，因为如果连接数越多，介于MySQL会为每个连接提供连接缓冲区，就会开销越多的内存，所以要适当调整该值，不能盲目提高设值。可以过'conn%'通配符查看当前状态的连接数量，以定夺该值的大小。

max\_connection=1024

 #mysql服务器支持的最大并发连接数据（用户数）;

一般情况下根据同时在线人数设置一个比较综合的数字，我们设置的是1000.

max\_connections 还取决于操作系统对单进程允许打开最大文件数的限制; MySQL 的 open\_files\_limit 参数值是在MySQL启动时记录的操作系统对单进程打开最大文件数限制的值可以使用 show variables like 'open\_files\_limit'; 查看 open\_files\_limit 值或者直接在 Linux 下通过ulimit -n命令查看操作系统对单进程打开最大文件数限制 ( 默认为1024 )

每个连接都需要内存。这个数字受你的内存限制。与操作系统无关。内存足够的话，你可以设置1-100000之间的数字。

max\_connections = 1000

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 = 50

# 每个客户端连接最大的错误允许数量,如果达到了此限制.这个客户端将会被 MySQL 服务阻止直到执行了 “FLUSH HOSTS” 或者服务重启

# 非法的密码以及其他在链接时的错误会增加此值.查看 “Aborted\_connects” 状态来获取全局计数器.

max\_connect\_errors = 6000

# 对于同一主机，如果有超出该参数值个数的中断错误连接，则该主机将被禁止连接。如需对该主机进行解禁，执行：FLUSH HOST。

max\_connect\_errors=10000

#如果某个用户发起的连接error超过该数值，则该用户的下次连接将被阻塞，直到管理员执行flush hosts ; 命令；防止黑客

max\_connect\_errors = 10000000

对于同一主机，如果有超出该参数值个数的中断错误连接，则该主机将被禁止连接。如需对该主机进行解禁，执行：FLUSH HOST;。

thread\_concurrency = 8

# 此允许应用程序给予线程系统一个提示在同一时间给予渴望被运行的线程的数量.此值只对于支持 thread\_concurrency() 函数的系统有意义( 例如Sun Solaris).

# 你可可以尝试使用 [CPU数量]\*(2..4) 来作为 thread\_concurrency 的值

thread\_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

open\_ files\_limit = 65535  
# MySQL打开的文件描述符限制，默认最小1024;当open\_files\_limit没有被配置的时候，比较max\_connections\*5和ulimit -n的值，哪个大用哪个，  
# 当open\_file\_limit被配置的时候，比较open\_files\_limit和max\_connections\*5的值，哪个大用哪个。

#external-locking

# 允许外部文件级别的锁. 打开文件锁会对性能造成负面影响，所以只有在你在同样的文件上运行多个数据库实例时才使用此选项(注意仍会有其他约束!)

# 或者你在文件层面上使用了其他一些软件依赖来锁定 MyISAM 表

max\_allowed\_packet = 32M

# 服务所能处理的请求包的最大大小以及服务所能处理的最大的请求大小(当与大的 BLOB 字段一起工作时相当必要)，每个连接独立的大小，大小动态增加

max\_allowed\_packet = 4M

# 接受的数据包大小；增加该变量的值十分安全，这是因为仅当需要时才会分配额外内存。例如，仅当你发出长查询或MySQLd必须返回大的结果行时MySQLd才会分配更多内存。  
# 该变量之所以取较小默认值是一种预防措施，以捕获客户端和服务器之间的错误信息包，并确保不会因偶然使用大的信息包而导致内存溢出。

max\_allowed\_packet = 8M

增加该变量的值十分安全，这是因为仅当需要时才会分配额外内存。例如，仅当你发出长查询或mysqld必须返回大的结果行时mysqld才会分配更多内存。该变量之所以取较小默认值是一种预防措施，以捕获客户端和服务器之间的错误信息包，并确保不会因偶然使用大的信息包而导致内存溢出。

max\_heap\_table\_size = 128M

# 独立的内存表所允许的最大容量.，此选项为了防止意外创建一个超大的内存表导致永尽所有的内存资源.

max\_heap\_table\_size = 8M

# MySQL的heap（堆积）表缓冲大小。所有联合在一个DML指令内完成，并且大多数联合甚至可以不用临时表即可以完成。  
# 大多数临时表是基于内存的(HEAP)表。具有大的记录长度的临时表 (所有列的长度的和)或包含BLOB列的表存储在硬盘上。  
# 如果某个内部heap（堆积）表大小超过tmp\_table\_size，MySQL可以根据需要自动将内存中的heap表改为基于硬盘的MyISAM表。还可以通过设置tmp\_table\_size选项来增加临时表的大小。也就是说，如果调高该值，MySQL同时将增加heap表的大小，可达到提高联接查询速度的效果

max\_heap\_table\_size=64M #该变量设置MEMORY (HEAP)表可以增长到的最大空间大小

tmp\_table\_size = 128M

# 内部(内存中)临时表的最大大小， 如果一个表增长到比此值更大,将会自动转换为基于磁盘的表.此限制是针对单个表的,而不是总和.

tmp\_table\_size = 16M  
# MySQL的heap（堆积）表缓冲大小。所有联合在一个DML指令内完成，并且大多数联合甚至可以不用临时表即可以完成。  
# 大多数临时表是基于内存的(HEAP)表。具有大的记录长度的临时表 (所有列的长度的和)或包含BLOB列的表存储在硬盘上。  
# 如果某个内部heap（堆积）表大小超过tmp\_table\_size，MySQL可以根据需要自动将内存中的heap表改为基于硬盘的MyISAM表。还可以通过设置tmp\_table\_size选项来增加临时表的大小。也就是说，如果调高该值，MySQL同时将增加heap表的大小，可达到提高联接查询速度的效果

tmp\_table\_size=64M

# 内存中的每个临时表允许的最大大小。如果临时表大小超过该值，临时表将自动转为基于磁盘的表

tmp\_table\_size = 256M

read\_buffer\_size = 2M  
# MySQL读入缓冲区大小。对表进行顺序扫描的请求将分配一个读入缓冲区，MySQL会为它分配一段内存缓冲区。read\_buffer\_size变量控制这一缓冲区的大小。  
# 如果对表的顺序扫描请求非常频繁，并且你认为频繁扫描进行得太慢，可以通过增加该变量值以及内存缓冲区大小提高其性能

read\_buffer\_size = 512k

 #读查询操作所能使用的缓冲区大小。

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!

read\_rnd\_buffer\_size = 16M

# 随机读取数据缓冲区使用内存(read\_rnd\_buffer\_size)：和顺序读取相对应，当 MySQL 进行非顺序读取（随机读取）数据块的时候，会利用>这个缓冲区暂存读取的数据

# 如根据索引信息读取表数据，根据排序后的结果集与表进行 Join 等等，总的来说，就是当数据块的读取需要满足>一定的顺序的情况下，MySQL 就需要产生随机读取，进而使用到 read\_rnd\_buffer\_size 参数所设置的内存缓冲区

read\_rnd\_buffer\_size = 8M

# MySQL的随机读缓冲区大小。当按任意顺序读取行时(例如，按照排序顺序)，将分配一个随机读缓存区。进行排序查询时，  
# MySQL会首先扫描一遍该缓冲，以避免磁盘搜索，提高查询速度，如果需要排序大量数据，可适当调高该值。但MySQL会为每个客户连接发放该缓冲空间，所以应尽量适当设置该值，以避免内存开销过大

 read\_rnd\_buffer\_size = 512K

#MyISAM以索引扫描(Random Scan)方式扫描数据的buffer大小；线程级别

sort\_buffer\_size = 16M

# 排序缓冲被用来处理类似 ORDER BY 以及 GROUP BY 队列所引起的排序，如果排序后的数据无法放入排序缓冲,一个用来替代的基于磁盘的合并分类会被使用

# 查看 “Sort\_merge\_passes” 状态变量.在排序发生时由每个线程分配

sort\_buffer\_size = 8M

# MySQL执行排序使用的缓冲大小。如果想要增加ORDER BY的速度，首先看是否可以让MySQL使用索引而不是额外的排序阶段。  
# 如果不能，可以尝试增加sort\_buffer\_size变量的大小

sort\_buffer\_size = 512k

#排序buffer大小；线程级别

sort\_buffer\_size = 4M

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 × 4 = 400MB。所以，对于内存在4GB左右的服务器推荐设置为4-8M。

join\_buffer\_size = 16M

# 此缓冲被使用来优化全联合(FULL JOINS 不带索引的联合).类似的联合在极大多数情况下有非常糟糕的性能表现,但是将此值设大能够减轻性能影响.

# 通过 “Select\_full\_join” 状态变量查看全联合的数量，当全联合发生时,在每个线程中分配

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

join\_buffer\_size=512K

 #联合查询操作所能使用的缓冲区大小。

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!

thread\_cache\_size = 16

# 我们在 cache 中保留多少线程用于重用，当一个客户端断开连接后,如果 cache 中的线程还少于 thread\_cache\_size,则客户端线程被放入cache 中.

# 这可以在你需要大量新连接的时候极大的减少线程创建的开销， (一般来说如果你有好的线程模型的话,这不会有明显的性能提升.)

thread\_cache\_size = 8

# 这个值（默认8）表示可以重新利用保存在缓存中线程的数量，当断开连接时如果缓存中还有空间，那么客户端的线程将被放到缓存中，  
# 如果线程重新被请求，那么请求将从缓存中读取,如果缓存中是空的或者是新的请求，那么这个线程将被重新创建,如果有很多新的线程，  
# 增加这个值可以改善系统性能.通过比较Connections和Threads\_created状态的变量，可以看到这个变量的作用。(–>表示要调整的值)  
# 根据物理内存设置规则如下：  
# 1G  —> 8  
# 2G  —> 16  
# 3G  —> 32  
# 大于3G  —> 64

thread\_cache\_size = 64

#缓存的最大线程数。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）。。

thread\_cache\_size = 64

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

table\_open\_cache = 4096

# 所有线程所打开表的数量.增加此值就增加了 mysqld 所需要的文件描述符的数量

# 这样你需要确认在 [mysqld\_safe] 中 “open-files-limit” 变量设置打开文件数量允许至少等于 table\_cache 的值

table\_open\_cache = 128

# MySQL每打开一个表，都会读入一些数据到table\_open\_cache缓存中，当MySQL在这个缓存中找不到相应信息时，才会去磁盘上读取。默认值64  
# 假定系统有200个并发连接，则需将此参数设置为200\*N(N为每个连接所需的文件描述符数目)；  
# 当把table\_open\_cache设置为很大时，如果系统处理不了那么多文件描述符，那么就会出现客户端失效，连接不上

table\_open\_cache = 6144

#表描述符缓存大小，可减少文件打开/关闭次数

table\_cache=3096   table\_cache(5.1.3之后这个值叫做table\_open\_cache)

#用于设置table高速缓存的数量。

table\_cache = 512

table\_cache指定表高速缓存的大小。每当MySQL访问一个表时，如果在表缓冲区中还有空间，该表就被打开并放入其中，这样可以更快地访问表内容。通过检查峰值时间的状态值Open\_tables和Opened\_tables，可以决定是否需要增加table\_cache的值。如果你发现 open\_tables等于table\_cache，并且opened\_tables在不断增长，那么你就需要增加table\_cache的值了(上述状态值可以使用SHOW STATUS LIKE ‘Open%tables’获得)。注意，不能盲目地把table\_cache设置成很大的值。如果设置得太高，可能会造成文件描述符不足，从而造成性能不稳定或者连接失败。

binlog\_cache\_size = 4M

# 在一个事务中 binlog 为了记录 SQL 状态所持有的 cache 大小，如果你经常使用大的,多声明的事务,你可以增加此值来获取更大的性能.

# 所有从事务来的状态都将被缓冲在 binlog 缓冲中然后在提交后一次性写入到 binlog 中，如果事务比此值大, 会使用磁盘上的临时文件来替代.

# 此缓冲在每个连接的事务第一次更新状态时被创建

binlog\_cache\_size = 1M

# 一个事务，在没有提交的时候，产生的日志，记录到Cache中；等到事务提交需要提交的时候，则把日志持久化到磁盘。默认binlog\_cache\_size大小32K

query\_cache\_size = 128M

# 查询缓冲常被用来缓冲 SELECT 的结果并且在下一次同样查询的时候不再执行直接返回结果.打开查询缓冲可以极大的提高服务器速度, 如果你有大量的相同的查询并且很少修改表.

# 查看 “Qcache\_lowmem\_prunes” 状态变量来检查是否当前值对于你的负载来说是否足够高.注意: 在你表经常变化的情况下或者如果你的查询原文每次都不同,

# 查询缓冲也许引起性能下降而不是性能提升.

query\_cache\_size = 8M

#MySQL的查询缓冲大小（从4.0.1开始，MySQL提供了查询缓冲机制）使用查询缓冲，MySQL将SELECT语句和查询结果存放在缓冲区中，  
# 今后对于同样的SELECT语句（区分大小写），将直接从缓冲区中读取结果。根据MySQL用户手册，使用查询缓冲最多可以达到238%的效率。  
# 通过检查状态值'Qcache\_%'，可以知道query\_cache\_size设置是否合理：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的值也非常大，则表明查询缓冲使用非常频繁，此时需要增加缓冲大小；如果Qcache\_hits的值不大，则表明你的查询重复率很低，  
# 这种情况下使用查询缓冲反而会影响效率，那么可以考虑不用查询缓冲。此外，在SELECT语句中加入SQL\_NO\_CACHE可以明确表示不使用查询缓冲

query\_cache\_size= 0

# 查询缓存大小，用于缓存SELECT查询结果。

query\_cache\_size = 64M

指定MySQL查询缓冲区的大小。可以通过在MySQL控制台执行以下命令观察：  
# > SHOW VARIABLES LIKE ‘%query\_cache%’; # > SHOW STATUS LIKE ‘Qcache%’; # 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。

bulk\_insert\_buffer\_size=32M

#默认8M参数来提高数据插入的效率，这个参数设置的是 bulk insert 的缓存大小

query\_cache\_limit = 4M

# 只有小于此设定值的结果才会被缓冲，此设置用来保护查询缓冲,防止一个极大的结果集将其他所有的查询结果都覆盖.

query\_cache\_limit = 2M

#指定单个查询能够使用的缓冲区大小，默认1M

key\_buffer\_size = 4M  
#指定用于索引的缓冲区大小，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。对于内存在4GB左右的服务器该参数可设置为384M或512M。通过检查状态值Key\_read\_requests和Key\_reads，  
# 可以知道key\_buffer\_size设置是否合理。比例key\_reads/key\_read\_requests应该尽可能的低，  
# 至少是1:100，1:1000更好(上述状态值可以使用SHOW STATUS LIKE 'key\_read%'获得)。注意：该参数值设置的过大反而会是服务器整体效率降低

key\_buffer\_size = 64M

#myisam索引buffer,只有key没有data，key\_buffer\_size是对**MyISAM**表性能影响最大的一个参数，如果没有设置**myisam** 那么将不生效；

key\_buffer\_size = 384M

 # key\_buffer\_size指定用于索引的缓冲区大小，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。对于内存在4GB左右的服务器该参数可设置为384M或512M。通过检查状态值Key\_read\_requests和 Key\_reads,可以知道key\_buffer\_size设置是否合理。比例key\_reads / key\_read\_requests应该尽可能的低，至少是1:100，1:1000更好(上述状态值可以使用SHOW STATUS LIKE ‘key\_read%’获得)。注意：该参数值设置的过大反而会是服务器整体效率降低!

ft\_min\_word\_len = 8

# 被全文检索索引的最小的字长.你也许希望减少它,如果你需要搜索更短字的时候.注意在你修改此值之后,你需要重建你的 FULLTEXT 索引

#memlock

# 如果你的系统支持 memlock() 函数,你也许希望打开此选项用以让运行中的 mysql 在在内存高度紧张的时候,数据在内存中保持锁定并且防止可能被 swapping out

# 此选项对于性能有益

default\_table\_type = InnoDB

# 当创建新表时作为默认使用的表类型,如果在创建表示没有特别执行表类型,将会使用此值

thread\_stack = 512K

# 线程使用的堆大小. 此容量的内存在每次连接时被预留.MySQL 本身常不会需要超过 64K 的内存

# 如果你使用你自己的需要大量堆的 UDF 函数或者你的操作系统对于某些操作需要更多的堆,你也许需要将其设置的更高一点.

transaction\_isolation = REPEATABLE-READ

# 设定默认的事务隔离级别.可用的级别如下:READ-UNCOMMITTED, READ-COMMITTED, REPEATABLE-READ, SERIALIZABLE

transaction\_isolation = REPEATABLE-READ

# MySQL支持4种事务隔离级别，他们分别是：  
# READ-UNCOMMITTED, READ-COMMITTED, REPEATABLE-READ, SERIALIZABLE.  
# 如没有指定，MySQL默认采用的是REPEATABLE-READ，ORACLE默认的是READ-COMMITTED

log-bin=mysql-bin

# 打开二进制日志功能.在复制(replication)配置中,作为 MASTER 主服务器必须打开此项，如果你需要从你最后的备份中做基于时间点的恢复,你也同样需要二进制日志.

#log\_slave\_updates

# 如果你在使用链式从服务器结构的复制模式 (A->B->C),你需要在服务器B上打开此项.，此选项打开在从线程上重做过的更新的日志, 并将其写入从服务器的二进制日志.

#log

# 打开全查询日志. 所有的由服务器接收到的查询 (甚至对于一个错误语法的查询)，都会被记录下来. 这对于调试非常有用, 在生产环境中常常关闭此项.

log\_error = /data/mysql/mysql-error.log

#错误日志路径

#log\_warnings

# 将警告打印输出到错误 log 文件. 如果你对于 MySQL 有任何问题，你应该打开警告 log 并且仔细审查错误日志,查出可能的原因.

log\_slow\_queries

# 记录慢速查询. 慢速查询是指消耗了比 “long\_query\_time” 定义的更多时间的查询.如果 log\_long\_format 被打开,那些没有使用索引的查询也会被记录.

# 如果你经常增加新查询到已有的系统内的话. 一般来说这是一个好主意,

#############slow log 慢查询###########

slow\_query\_log =1

#1为开启慢查询 on

slow\_query\_log\_file=/opt/mysql/logs/mysql.slow

#慢查询的日志目录

long\_query\_time=2

#慢查询时间  超过2秒则为慢查询

long\_query\_time = 6

# 所有的使用了比这个时间(以秒为单位)更多的查询会被认为是慢速查询.

# 不要在这里使用“1″, 否则会导致所有的查询,甚至非常快的查询页被记录下来(由于 MySQL 目前时间的精确度只能达到秒的级别).

long\_query\_time = 1

指定记录慢查询的时间阀值，查询时间超过1s就记录

log\_long\_format

简单的说log-long-format选项是用来设置日志的格式，它是以扩展方式记录有关事件。扩展方式可记录谁发出查询和什么时候发出查询的信息。可使我们更好地掌握客户端的操作情况。

准确的说，它是记录激活的更新日志、二进制更新日志、和慢查询日志的大量信息。例如，所有查询的用户名和时间戳将记录下来。不赞成选用该选项，因为它现在代表 默认记录行为

#tmpdir = /tmp

# 此目录被MySQL用来保存临时文件.例如,它被用来处理基于磁盘的大型排序,和内部排序一样. 以及简单的临时表.

# 如果你不创建非常大的临时文件,将其放置到 swapfs/tmpfs 文件系统上也许比较好，另一种选择是你也可以将其放置在独立的磁盘上.

# 你可以使用”;”来放置多个路径， 他们会按照 roud-robin 方法被轮询使用.