以下是 my.cnf 配置文件参数解释：

```
1. [client]
2. port = 3309
3. socket = /home/mysql/mysql/tmp/mysql.sock
4. [mysqld]
5. !include /home/mysql/mysql/etc/mysqld.cnf #包含的配置文件 ，把用户名，密码文件单独存放
6. port = 3309
7. socket = /home/mysql/mysql/tmp/mysql.sock
8. pid-file = /longxibendi/mysql/mysql/var/mysql.pid
9. basedir = /home/mysql/mysql/
10. datadir = /longxibendi/mysql/mysql/var/
11. # tmp dir settings
12. # 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.
13. tmpdir = /home/mysql/mysql/tmp/
14. slave-load-tmpdir = /home/mysql/mysql/tmp/
15. #当 slave 执行 load data infile 时用
16. #language = /home/mysql/mysql/share/mysql/english/
17. character-sets-dir = /home/mysql/mysql/share/mysql/charsets/
18. # skip options
19. #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！
20. skip-name-resolve #grant 时，必须使用 ip 不能使用主机名
21. skip-symbolic-links #不能使用连接文件
22. #多个客户可能会访问同一个数据库，因此这防止外部客户锁定 MySQL 服务器。该选项默认开启
23. skip-external-locking #不使用系统锁定，要使用 myisamchk,必须关闭服务器 ,避免 MySQL 的外部锁定，减少出错几率增强稳定性。
24. skip-slave-start #启动 mysql,不启动复制
25. #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！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
26. skip-networking
27. #sysdate-is-now
```

```
28. # res settings
29. #指定 MySQL 可能的连接数量。当 MySQL 主线程在很短的时间内接收到非常多的连接请求，该参数生效，主线程花费很短的时间检查连接并且启动一个新线程。 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.
    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。
30. back_log = 50 #接受队列,对于没建立 tcp 连接的请求队列放入缓存中,队列大小为 back_log,受限制与 OS 参数
31. #指定 MySQL 允许的最大连接进程数。如果在访问数据库时经常出现 Too Many Connections 的错误提 示，则需要增大该参数值。
32. max_connections = 1000 #最大并发连接数 ，增大该值需要相应增加允许打开的文件描述符数
33. 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 ； 命令或者服务重启，防止黑客 ， 非法的密码以及其他在链接时的错误会增加此值
34. #open_files_limit = 10240
35. connect-timeout = 10 #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
36. #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。
37. wait-timeout = 28800 #等待关闭连接的时间
38. interactive-timeout = 28800 #关闭连接之前，允许 interactive_timeout（取代了 wait_timeout）秒的不活动时间。客户端的会话 wait_timeout 变量被设为会话 interactive_timeout 变量的值。
39. slave-net-timeout = 600 #从服务器也能够处理网络连接中断。但是，只有从服务器超过 slave_net_timeout 秒没有从主服务器收到数据才通知网络中断
40. net_read_timeout = 30 #从服务器读取信息的超时
41. net_write_timeout = 60 #从服务器写入信息的超时
42. net_retry_count = 10 #如果某个通信端口的读操作中断了，在放弃前重试多次
43. net_buffer_length = 16384 #包消息缓冲区初始化为 net_buffer_length 字节,但需要时可以增长到 max_allowed_packet 字节
44. # 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。 设置最大包,限制 server 接受的数据包大小,避免超长 SQL 的执行有问题 默认值为 16M,当 MySQL 客户端或 mysqld 服务器收到大于 max_allowed_packet 字节的信息包时，将发出“信息包过大”错误，并关闭连接。对于某些客户端，如果通信信息包过大，在执行查询期间，可能会遇到“丢失与 MySQL 服务器的连接”错误。默认值 16M。
45. max_allowed_packet = 64M #
46. # 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 4096
```

```
#table_cache = 512 #所有线程打开的表的数目。增大该值可以增加 mysqld 需要的文件描述符的数量
# 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.默认设置足以满足大多数应用
thread_stack = 192K #每个线程的堆栈大小
# 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.)
# 服务器线程缓存这个值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
thread_cache_size = 20 #线程缓存
#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.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
# 设置 thread_concurrency 的值的正确与否,对 mysql 的性能影响很大,在多个 cpu(或多核)的情况下,错误设置了 thread_concurrency 的值,会导致 mysql 不能充分利用多 cpu(或多核),出现同一时刻只能一个 cpu(或核)在工作的情况。thread_concurrency 应设为 CPU 核数的 2 倍.比如有一个双核的 CPU,那么 thread_concurrency 的应该为 4; 2 个双核的 cpu, thread_concurrency 的值应为 8
#属重点优化参数
thread_concurrency = 8 #同时运行的线程的数据 此处最好为 CPU 个数两倍。本机配置为 CPU 的个数
# qcache settings
# 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
# 对于使用 MySQL 的用户,对于这个变量大家一定不会陌生。前几年的 MyISAM 引擎优化中,这个参数也是一个重要的优化参数。但随着发展,这个参数也爆露出来一些问题。机器的内存越来越大,人们也都习惯性的把以前有用的参数分配的值越来越大。这个参数加大后也引发了一系列问题。我们首先分析一下 query_cache_size 的工作原理:一个 SELECT 查询在 DB 中工作后,DB 会把该语句缓存下来,当同样的一个 SQL 再次来到 DB 里调用时,DB 在该表没发生变化的情况下把结果从缓存中返回给 Client。这里有一个关建点,就是 DB 在利用 Query_cache 工作时,要求该语句涉及的表在这段时间内没有发生变更。那如果该表在发生变更时,Query_cache 里的数据又怎么处理呢?首先要把 Query_cache 和该表相关的语句全部置为失效,然后在写入更新。那么如果 Query_cache 非常大,该表的查询结构又比较多,查询语句失效也慢,一个更新或是 Insert 就会很慢,这样看到的就是
```

```
Update 或是 Insert 怎么这么慢了。所以在数据库写入量或是更新量也比较大的系统，该参数不适合分配过大。而且在高并发，写入量大的系统，建系把该功能禁掉。
#重点优化参数（主库 增删改-MyISAM）
query_cache_size = 256M #查询缓存大小
# 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 #不缓存查询大于该值的结果
#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%
query_cache_min_res_unit = 2K #查询缓存分配的最小块大小
# 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
#memlock
# default settings
# 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
default_table_type = MYISAM
# time zone
default-time-zone = system #服务器时区
character-set-server = utf8 #server 级别字符集
default-storage-engine = InnoDB #默认存储
# tmp & heap
# 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 = 512M #临时表大小，如果超过该值，则结果放到磁盘中
# 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 = 512M #该变量设置 MEMORY (HEAP 堆)表可以增长到的最大空间大小
# 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-bin = mysql-bin #这些路径相对于 datadir
```

```
80.# 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.
81.log_slave_updates
82.log-bin-index = mysql-bin.index
83.relay-log = relay-log
84.relay_log_index = relay-log.index
85.# 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.
86.log
87.# warning & error log
88.# 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.
89.log-warnings = 1
90.log-error = /home/mysql/mysql/log/mysql.err
91.log_output = FILE #参数 log_output 指定了慢查询输出的格式，默认为 FILE，你可以将它设
   为 TABLE，然后就可以查询 mysql 架构下的 slow_log 表了
92.# 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,
93.log_slow_queries
94.# slow query log
95.slow_query_log = 1
96.# 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).
97.long-query-time = 1 #慢查询时间 超过 1 秒则为慢查询
98.# 在慢速日志中记录更多的信息.
   # 一般此项最好打开.
   # 打开此项会记录使得那些没有使用索引的查询也被作为到慢速查询附加到慢速日志里
99.log_long_format
100.slow_query_log_file = /home/mysql/mysql/log/slow.log
101.#log-queries-not-using-indexes
102.#log-slow-slave-statements
103.general_log = 1
104.general_log_file = /home/mysql/mysql/log/mysql.log
105.max_binlog_size = 1G
106.max_relay_log_size = 1G
107.# if use auto-ex, set to 0
108.relay-log-purge = 1 #当不用中继日志时，删除他们。这个操作有 SQL 线程完成
109.# max binlog keeps days
110.expire_logs_days = 30 #超过 30 天的 binlog 删除
```

```
111.# 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
112.binlog_cache_size = 1M #session 级别
113.# replication
114.replicate-wild-ignore-table = mysql.% #复制时忽略数据库及表
115.replicate-wild-ignore-table = test.% #复制时忽略数据库及表
116.# slave_skip_errors=all
117.#*** MyISAM 相关选项
118.# 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！
119.# 关键词缓冲的大小, 一般用来缓冲 MyISAM 表的索引块.
    # 不要将其设置大于你可用内存的 30%,
    # 因为一部分内存同样被 OS 用来缓冲行数据
    # 甚至在你并不使用 MyISAM 表的情况下, 你也需要仍旧设置起 8-64M 内存由于它同样会被内部临时磁盘表使用.
120.key_buffer_size = 256M #myisam 索引 buffer,只有 key 没有 data
121.#查询排序时所能使用的缓冲区大小。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。# 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  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。#
122.sort_buffer_size = 2M #排序 buffer 大小; 线程级别
123.#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！用来做 MyISAM 表全表扫描的缓冲大小.当全表扫描需要时,在对应线程中分配.
124.read_buffer_size = 2M #以全表扫描(Sequential Scan)方式扫描数据的 buffer 大小 ; 线程级别
125.#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！ # 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。
126.join_buffer_size = 8M # join buffer 大小;线程级别
127.#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：
    代码：
    # > 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。
128.query_cache_size = 64M
```

```
129.# 当在排序之后,从一个已经排序好的序列中读取行时,行数据将从这个缓冲中读取来防止磁盘寻道.
    # 如果你增高此值,可以提高很多 ORDER BY 的性能.当需要时由每个线程分配
130.read_rnd_buffer_size = 8M #MyISAM 以索引扫描(Random Scan)方式扫描数据的 buffer 大小 ; 线程级别
131.# MyISAM 使用特殊的类似树的 cache 来使得突发插入
    # (这些插入是,INSERT … SELECT, INSERT … VALUES (…), (…), …, 以及 LOAD DATA
    # INFILE) 更快. 此变量限制每个进程中缓冲树的字节数.
    # 设置为 0 会关闭此优化.
    # 为了最优化不要将此值设置大于 "key_buffer_size".
    # 当突发插入被检测到时此缓冲将被分配
132.bulk_insert_buffer_size = 64M #MyISAM 用在块插入优化中的树缓冲区的大小。注释：这是一个 per thread 的限制 （bulk 大量）
133.# 此缓冲当 MySQL 需要在 REPAIR, OPTIMIZE, ALTER 以及 LOAD DATA INFILE 到一个空表中引起重建索引时被分配.这在每个线程中被分配.所以在设置大值时需要小心.
134.myisam_sort_buffer_size = 64M #MyISAM 设置恢复表之时使用的缓冲区的尺寸,当在 REPAIR TABLE 或用 CREATE INDEX 创建索引或 ALTER TABLE 过程中排序 MyISAM 索引分配的缓冲区
135.# MySQL 重建索引时所允许的最大临时文件的大小 (当 REPAIR, ALTER TABLE 或者 LOAD DATA INFILE). 如果文件大小比此值更大,索引会通过键值缓冲创建(更慢)
136.myisam_max_sort_file_size = 10G #MyISAM 如果临时文件会变得超过索引，不要使用快速排序索引方法来创建一个索引。注释：这个参数以字节的形式给出.重建 MyISAM 索引(在 REPAIR TABLE、ALTER TABLE 或 LOAD DATA INFILE 过程中)时，允许 MySQL 使用的临时文件的最大空间大小。如果文件的大小超过该值，则使用键值缓存创建索引，要慢得多。该值的单位为字节
137.# 如果被用来更快的索引创建索引所使用临时文件大于制定的值,那就使用键值缓冲方法.
    # 这主要用来强制在大表中长字串键去使用慢速的键值缓冲方法来创建索引.
138.myisam_max_extra_sort_file_size = 10G
139.# 如果一个表拥有超过一个索引, MyISAM 可以通过并行排序使用超过一个线程去修复他们.这对于拥有多个 CPU 以及大量内存情况的用户,是一个很好的选择.
140.myisam_repair_threads = 1 #如果该值大于 1，在 Repair by sorting 过程中并行创建 MyISAM 表索引(每个索引在自己的线程内)
141.myisam_recover = 64K#允许的 GROUP_CONCAT()函数结果的最大长度
142.# 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
    # 1.READ UNCOMMITTED-读未提交 2.READ COMMITTE-读已提交 3.REPEATABLE READ -可重复读 4.SERIALIZABLE -串行
    transaction_isolation = REPEATABLE-READ
143.# *** INNODB 相关选项 ***
144.# 如果你的 MySQL 服务包含 InnoDB 支持但是并不打算使用的话,
    # 使用此选项会节省内存以及磁盘空间,并且加速某些部分
    skip-innodb
145.innodb_file_per_table
146.#innodb_status_file = 1
```

```
147.#innodb_open_files = 2048
148.# 附加的内存池被 InnoDB 用来保存 metadata 信息
    # 如果 InnoDB 为此目的需要更多的内存,它会开始从 OS 这里申请内存.
    # 由于这个操作在大多数现代操作系统上已经足够快, 你一般不需要修改此值.
    # SHOW INNODB STATUS 命令会显示当先使用的数量.
149.innodb_additional_mem_pool_size = 100M #帧缓存的控制对象需要从此处申请缓存，所以该值与 innodb_buffer_pool 对应
150.# InnoDB 使用一个缓冲池来保存索引和原始数据, 不像 MyISAM.
    # 这里你设置越大,你在存取表里面数据时所需要的磁盘 I/O 越少.
    # 在一个独立使用的数据库服务器上,你可以设置这个变量到服务器物理内存大小的 80%
    # 不要设置过大,否则,由于物理内存的竞争可能导致操作系统的换页颠簸.
    # 注意在 32 位系统上你每个进程可能被限制在 2-3.5G 用户层面内存限制,
    # 所以不要设置的太高.
151.innodb_buffer_pool_size = 2G #包括数据页、索引页、插入缓存、锁信息、自适应哈希所以、数据字典信息
152.# 设置此选项如果你希望 InnoDB 表空间文件被保存在其他分区.
    # 默认保存在 MySQL 的 datadir 中.
153.innodb_data_home_dir = /longxibendi/mysql/mysql/var/
154.# InnoDB 将数据保存在一个或者多个数据文件中成为表空间.
    # 如果你只有单个逻辑驱动保存你的数据,一个单个的自增文件就足够好了.
    # 其他情况下.每个设备一个文件一般都是个好的选择.
    # 你也可以配置 InnoDB 来使用裸盘分区 - 请参考手册来获取更多相关内容
155.#innodb_data_file_path = ibdata1:1G:autoextend
156.innodb_data_file_path = ibdata1:500M;ibdata2:2210M:autoextend #表空间
157.# 用来同步 IO 操作的 IO 线程的数量. This value is
    # 此值在 Unix 下被硬编码为 4,但是在 Windows 磁盘 I/O 可能在一个大数值下表现的更好.
158.innodb_file_io_threads = 4 #io 线程数
159.# 在 InnoDb 核心内的允许线程数量.
    # 最优值依赖于应用程序,硬件以及操作系统的调度方式.
    # 过高的值可能导致线程的互斥颠簸.
160.innodb_thread_concurrency = 16 #InnoDB 试着在 InnoDB 内保持操作系统线程的数量少于或等于这个参数给出的限制
161.# 如果设置为 1 ,InnoDB 会在每次提交后刷新(fsync)事务日志到磁盘上,
    # 这提供了完整的 ACID 行为.
    # 如果你愿意对事务安全折衷, 并且你正在运行一个小的食物, 你可以设置此值到 0 或者 2 来减少由事务日志引起的磁盘 I/O
    # 0 代表日志只大约每秒写入日志文件并且日志文件刷新到磁盘.
    # 2 代表日志写入日志文件在每次提交后,但是日志文件只有大约每秒才会刷新到磁盘上.
162.innodb_flush_log_at_trx_commit = 1 #每次 commit 日志缓存中的数据刷到磁盘中
163.# 用来缓冲日志数据的缓冲区的大小.
    # 当此值快满时, InnoDB 将必须刷新数据到磁盘上.
    # 由于基本上每秒都会刷新一次,所以没有必要将此值设置的太大(甚至对于长事务而言)
164.innodb_log_buffer_size = 8M #事物日志缓存
```

```
165. # 在日志组中每个日志文件的大小，你应该设置日志文件总合大小到你缓冲池大小的
     25%~100%，来避免在日志文件覆写上不必要的缓冲池刷新行为.
     # 不论如何，请注意一个大的日志文件大小会增加恢复进程所需要的时间.
166. innodb_log_file_size = 500M #事物日志大小
167. #innodb_log_file_size =100M
168. 在日志组中的文件总数.
     # 通常来说 2~3 是比较好的.
169. innodb_log_files_in_group = 2 #两组事物日志
170. # InnoDB 的日志文件所在位置. 默认是 MySQL 的 datadir.
     # 你可以将其指定到一个独立的硬盘上或者一个 RAID1 卷上来提高其性能
171. innodb_log_group_home_dir = /longxibendi/mysql/mysql/var/#日志组
172. # 在 InnoDB 缓冲池中最大允许的脏页面的比例.
     # 如果达到限额, InnoDB 会开始刷新他们防止他们妨碍到干净数据页面.
     # 这是一个软限制,不被保证绝对执行.
173. innodb_max_dirty_pages_pct = 90 #innodb 主线程刷新缓存池中的数据，使脏数据比例小
     于 90%
174. # 在被回滚前,一个 InnoDB 的事务应该等待一个锁被批准多久.
     # InnoDB 在其拥有的锁表中自动检测事务死锁并且回滚事务.
     # 如果你使用 LOCK TABLES 指令，或者在同样事务中使用除了 InnoDB 以外的其他事务安全的存储
     引擎
     # 那么一个死锁可能发生而 InnoDB 无法注意到.
     # 这种情况下这个 timeout 值对于解决这种问题就非常有帮助.
175. innodb_lock_wait_timeout = 50 #InnoDB 事务在被回滚之前可以等待一个锁定的超时秒数。
     InnoDB 在它自己的 锁定表中自动检测事务死锁并且回滚事务。InnoDB 用 LOCK TABLES 语句注意
     到锁定设置。默认值是 50 秒
176. # InnoDB 用来刷新日志的方法.
     # 表空间总是使用双重写入刷新方法
     # 默认值是 “fdatasync”，另一个是 “O_DSYNC”.
177. #innodb_flush_method = O_DSYNC
178. # 如果你发现 InnoDB 表空间损坏，设置此值为一个非零值可能帮助你导出你的表.
     # 从 1 开始并且增加此值知道你能够成功的导出表.
179. innodb_force_recovery=1
180. # 加速 InnoDB 的关闭. 这会阻止 InnoDB 在关闭时做全清除以及插入缓冲合并.
     # 这可能极大增加关机时间，但是取而代之的是 InnoDB 可能在下次启动时做这些操作.
      innodb_fast_shutdown
181. [mysqldump]
182. # 不要在将内存中的整个结果写入磁盘之前缓存. 在导出非常巨大的表时需要此项
183. quick
184. 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。 如果你正是用大的 BLOB 值，而且未为 mysqld 授予
     为处理查询而访问足够内存的权限，也会遇到与大信息包有关的奇怪问题。如果怀疑出
     现了该情况，请尝试在 mysqld_safe 脚本开始增加 ulimit -d 256000，并重启 mysqld。
```

```
max_allowed_packet = 64M
[mysql]
disable-auto-rehash #允许通过 TAB 键提示
default-character-set = utf8
connect-timeout = 3
[mysqld_safe]
# 增加每个进程的可打开文件数量.
# 警告: 确认你已经将全系统限制设定的足够高!
# 打开大量表需要将此值设 b
open-files-limit = 8192
```

```
##############################################
#####      MySQL 怎样打开和关闭数据库表 #####
##############################################
```

table_cache, max_connections 和 max_tmp_tables 影响服务器保持打开的文件的最大数量。如果你增加这些值的一个或两个，你可以遇到你的操作系统每个进程打开文件描述符的数量上强加的限制。然而，你可以能在许多系统上增加该限制。请教你的 OS 文档找出如何做这些，因为改变限制的方法各系统有很大的不同。

table_cache 与 max_connections 有关。例如，对于 200 个打开的连接，你应该让一张表的缓冲至少有 200 * n，这里 n 是一个联结(join)中表的最大数量。

打开表的缓存可以增加到一个 table_cache 的最大值（缺省为 64；这可以用 mysqld 的-O table_cache=#选项来改变）。一个表绝对不被关闭，除非当缓存满了并且另外一个线程试图打开一个表时或如果你使用 mysqladmin refresh 或 mysqladmin flush-tables。

当表缓存满时，服务器使用下列过程找到一个缓存入口来使用：

不是当前使用的表被释放，以最近最少使用（LRU）顺序。

如果缓存满了并且没有表可以释放，但是一个新表需要打开，缓存必须临时被扩大。

如果缓存处于一个临时扩大状态并且一个表从在用变为不在用状态，它被关闭并从缓存中释放。

对每个并发存取打开一个表。这意味着，如果你让 2 个线程存取同一个表或在同一个查询中存取表两次(用 AS)，表需要被打开两次。任何表的第一次打开占 2 个文件描述符；表的每一次额外使用仅占一个文件描述符。对于第一次打开的额外描述符用于索引文件；这个描述符在所有线程之间共享

MySQL 5.5.13

参数说明：

```
[client]
character-set-server = utf8
port    = 3306
socket  = /data/mysql/3306/mysql.sock
[mysqld]
character-set-server = utf8
user    = mysql
port    = 3306
socket  = /data/mysql/3306/mysql.sock
basedir = /usr/local/webserver/mysql
datadir = /data/mysql/3306/data
log-error = /data/mysql/3306/mysql_error.log
pid-file = /data/mysql/3306/mysql.pid
# table_cache 参数设置表高速缓存的数目。每个连接进来，都会至少打开一个表缓存。#因此，table_cache 的大小应与 max_connections 的设置有关。例如，对于 200 个#并行运行的连接，应该让表的缓存至少有 200 × N ，这里 N 是应用可以执行的查询#的一个联接中表的最大数量。此外，还需要为临时表和文件保留一些额外的文件描述符。
# 当 Mysql 访问一个表时，如果该表在缓存中已经被打开，则可以直接访问缓存；如果#还没有被缓存，但是在 Mysql 表缓冲区中还有空间，那么这个表就被打开并放入表缓#冲区；如果表
```

缓存满了，则会按照一定的规则将当前未用的表释放，或者临时扩大表缓存来存放，使用表缓存的好处是可以更快速地访问表中的内容。执行 flush tables 会#清空缓存的内容。一般来说，可以通过查看数据库运行峰值时间的状态值 Open_tables #和 Opened_tables ,判断是否需要增加 table_cache 的值（其中 open_tables 是当#前打开的表的数量， Opened_tables 则是已经打开的表的数量）。即如果 open_tables 接近 table_cache 的时候，并且 Opened_tables 这个值在逐步增加，那就要考虑增加这个#值的大小了。还有就是 Table_locks_waited 比较高的时候，也需要增加 table_cache。

open_files_limit = 10240

table_cache = 512

#非动态变量，需要重启服务

指定 MySQL 可能的连接数量。当 MySQL 主线程在很短的时间内接收到非常多的连接请求，该参数生效，主线程花费很短的时间检查连接并且启动一个新线程。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。

back_log = 600

#MySQL 允许最大连接数

max_connections = 5000

#可以允许多少个错误连接

max_connect_errors = 6000

#使用–skip-external-locking MySQL 选项以避免外部锁定。该选项默认开启

external-locking = FALSE

设置最大包,限制 server 接受的数据包大小，避免超长 SQL 的执行有问题 默认值为 16M，当 MySQL 客户端或 mysqld 服务器收到大于 max_allowed_packet 字节的信息包时，将发出“信息包过大”错误，并关闭连接。对于某些客户端，如果通信信息包过大，在执行查询期间，可能会遇到“丢失与 MySQL 服务器的连接”错误。默认值 16M。

#dev-doc: http://dev.mysql.com/doc/refman/5.5/en/packet-too-large.html

max_allowed_packet = 32M

Sort_Buffer_Size 是一个 connection 级参数，在每个 connection（session）第一次需要使用这个 buffer 的时候，一次性分配设置的内存。

#Sort_Buffer_Size 并不是越大越好，由于是 connection 级的参数，过大的设置+高并发可能会耗尽系统内存资源。例如：500 个连接将会消耗 500*sort_buffer_size(8M)=4G 内存

#Sort_Buffer_Size 超过 2KB 的时候，就会使用 mmap() 而不是 malloc() 来进行内存分配，导致效率降低。

#技术导读 http://blog.webshuo.com/2011/02/16/mysql-sort_buffer_size/

#dev-doc: http://dev.mysql.com/doc/refman/5.5/en/server-parameters.html

#explain select*from table where order limit；出现 filesort

#属重点优化参数

sort_buffer_size = 8M

#用于表间关联缓存的大小

join_buffer_size = 1M

服务器线程缓存这个值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

thread_cache_size = 300

设置 thread_concurrency 的值的正确与否，对 mysql 的性能影响很大，在多个 cpu(或多核)的情况下，错误设置了 thread_concurrency 的值，会导致 mysql 不能充分利用多 cpu(或多核)，出现同一时刻只能一个 cpu(或核)在工作的情况。thread_concurrency 应设为 CPU 核数的 2 倍.比如有一个双核的 CPU，那么 thread_concurrency 的应该为 4; 2 个双核的 cpu，thread_concurrency 的值应为 8

#属重点优化参数

thread_concurrency = 8

对于使用 MySQL 的用户，对于这个变量大家一定不会陌生。前几年的 MyISAM 引擎优化中，这个参数也是一个重要的优化参数。但随着发展，这个参数也爆露出来一些问题。机器的内存越来越大，人们也都习惯性的把以前有用的参数分配的值越来越大。这个参数加大后也引发了一系列问题。我们首先分析一下 query_cache_size 的工作原理：一个 SELECT 查询在 DB 中工作后，DB 会把该语句缓存下来，当同样的一个 SQL 再次来到 DB 里调用时，DB 在该表没发生变化的情况下把结果从缓存中返回给 Client。这里有一个关建点，就是 DB 在利用 Query_cache 工作时，要求该语句涉及的表在这段时间内没有发生变更。那如果该表在发生变更时，Query_cache 里的数据又怎么处理呢？首先要把 Query_cache 和该表相关的语句全部置为失效，然后在写入更新。那么如果 Query_cache 非常大，该表的查询结构又比较多，查询语句失效也慢，一个更新或是 Insert 就会很慢，这样看到的就是 Update 或是 Insert 怎么这么慢了。所以在数据库写入量或是更新量也比较大的系统，该参数不适合分配过大。而且在高并发，写入量大的系统，建系把该功

```
能禁掉。

#重点优化参数（主库 增删改-MyISAM）

query_cache_size = 512M

#指定单个查询能够使用的缓冲区大小，缺省为 1M

query_cache_limit = 2M

#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%

query_cache_min_res_unit = 2k

default-storage-engine = MyISAM

#限定用于每个数据库线程的栈大小。默认设置足以满足大多数应用

thread_stack = 192K

# 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

# 1.READ UNCOMMITTED-读未提交 2.READ COMMITTE-读已提交 3.REPEATABLE READ -
```

可重复读 4.SERIALIZABLE -串行

transaction_isolation = READ-COMMITTED

tmp_table_size 的默认大小是 32M。如果一张临时表超出该大小，MySQL 产生一个 The table tbl_name is full 形式的错误 ,如果你做很多高级 GROUP BY 查询 ,增加 tmp_table_size 值。

tmp_table_size = 246M

max_heap_table_size = 246M

#索引缓存大小: 它决定了数据库索引处理的速度，尤其是索引读的速度

key_buffer_size = 512M

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 = 4M

MySql 的随机读（查询操作）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 = 16M

#批量插入数据缓存大小，可以有效提高插入效率，默认为 8M

bulk_insert_buffer_size = 64M

MyISAM 表发生变化时重新排序所需的缓冲

myisam_sort_buffer_size = 128M

```
# MySQL 重建索引时所允许的最大临时文件的大小 (当 REPAIR, ALTER TABLE 或者 LOAD DATA INFILE).
# 如果文件大小比此值更大,索引会通过键值缓冲创建(更慢)
myisam_max_sort_file_size = 10G
# 如果一个表拥有超过一个索引, MyISAM 可以通过并行排序使用超过一个线程去修复他们.
# 这对于拥有多个 CPU 以及大量内存情况的用户,是一个很好的选择.
myisam_repair_threads = 1
#自动检查和修复没有适当关闭的 MyISAM 表
myisam_recover
interactive_timeout = 120
wait_timeout = 120
innodb_data_home_dir = /data/mysql/3306/data
#表空间文件 重要数据
innodb_data_file_path = ibdata1:2000M;ibdata2:10M:autoextend
#这个参数用来设置 InnoDB 存储的数据目录信息和其它内部数据结构的内存池大小，类似于 Oracle 的 library cache。这不是一个强制参数，可以被突破。
innodb_additional_mem_pool_size = 16M
# 这对 Innodb 表来说非常重要。Innodb 相比 MyISAM 表对缓冲更为敏感。MyISAM 可以在默认的 key_buffer_size 设置下运行的可以，然而 Innodb 在默认的 innodb_buffer_pool_size 设置下却跟蜗牛似的。由于 Innodb 把数据和索引都缓存起来，无需留给操作系统太多的内存，因此如果只需要用 Innodb 的话则可以设置它高达 70-80% 的可用内存。一些应用于 key_buffer 的规则有 — 如果你的数据量不大，并且不会暴增，那么无需把
```

innodb_buffer_pool_size 设置的太大了

innodb_buffer_pool_size = 512M

#文件 IO 的线程数，一般为 4，但是在 Windows 下，可以设置得较大。

innodb_file_io_threads = 4

在 InnoDb 核心内的允许线程数量.

最优值依赖于应用程序,硬件以及操作系统的调度方式.

过高的值可能导致线程的互斥颠簸.

innodb_thread_concurrency = 8

如果将此参数设置为 1，将在每次提交事务后将日志写入磁盘。为提供性能，可以设置为 0 或 2，但要承担在发生故障时丢失数据的风险。设置为 0 表示事务日志写入日志文件，而日志文件每秒刷新到磁盘一次。设置为 2 表示事务日志将在提交时写入日志，但日志文件每次刷新到磁盘一次。

innodb_flush_log_at_trx_commit = 2

#此参数确定些日志文件所用的内存大小，以 M 为单位。缓冲区更大能提高性能，但意外的故障将会丢失数据.MySQL 开发人员建议设置为 1 - 8M 之间

innodb_log_buffer_size = 16M

#此参数确定数据日志文件的大小，以 M 为单位，更大的设置可以提高性能，但也会增加恢复故障数据库所需的时间

innodb_log_file_size = 128M

#为提高性能，MySQL 可以以循环方式将日志文件写到多个文件。推荐设置为 3M

innodb_log_files_in_group = 3

#推荐阅读

http://www.taobaodba.com/html/221_innodb_max_dirty_pages_pct_checkpoint.html

Buffer_Pool 中 Dirty_Page 所占的数量，直接影响 InnoDB 的关闭时间。参数 innodb_max_dirty_pages_pct 可以直接控制了 Dirty_Page 在 Buffer_Pool 中所占的比率，而且幸运的是 innodb_max_dirty_pages_pct 是可以动态改变的。所以，在关闭 InnoDB 之前先将 innodb_max_dirty_pages_pct 调小，强制数据块 Flush 一段时间，则能够大大缩短 MySQL 关闭的时间。

innodb_max_dirty_pages_pct = 90

InnoDB 有其内置的死锁检测机制，能导致未完成的事务回滚。但是，如果结合 InnoDB 使用 MyISAM 的 lock tables 语句或第三方事务引擎,则 InnoDB 无法识别死锁。为消除这种可能性，可以将 innodb_lock_wait_timeout 设置为一个整数值，指示 MySQL 在允许其他事务修改那些最终受事务回滚的数据之前要等待多长时间(秒数)

innodb_lock_wait_timeout = 120

#独享表空间（关闭）

innodb_file_per_table = 0

#start mysqld with –slow-query-log-file=/data/mysql/3306/slow.log

slow_query_log

long_query_time = 1

replicate-ignore-db = mysql

replicate-ignore-db = test

replicate-ignore-db = information_schema

#配置从库上的更新操作是否写二进制文件，如果这台从库，还要做其他从库的主库，那么就需要打这个参数，以便从库的从库能够进行日志同步这个参数要和—logs-bin 一起使用

```
log-slave-updates

log-bin = /data/mysql/3306/binlog/binlog

binlog_cache_size = 4M

#STATEMENT,ROW,MIXED

# 基于 SQL 语句的复制(statement-based replication, SBR)，基于行的复制(row-based

replication, RBR)，混合模式复制(mixed-based replication, MBR)。相应地，binlog 的格式也

有三种：STATEMENT，ROW，MIXED。

binlog_format = MIXED

max_binlog_cache_size = 64M

max_binlog_size = 1G

relay-log-index = /data/mysql/3306/relaylog/relaylog

relay-log-info-file = /data/mysql/3306/relaylog/relaylog

relay-log = /data/mysql/3306/relaylog/relaylog

expire_logs_days = 30

skip-name-resolve

#master-connect-retry = 10

slave-skip-errors = 1032,1062,126,1114,1146,1048,1396

server-id = 1

[mysqldump]

quick

max_allowed_packet = 32M

[myisamchk]
```

```
key_buffer_size = 256M

sort_buffer_size = 256M

read_buffer = 2M

write_buffer = 2M

[mysqlhotcopy]

interactive-timeout
```

一、服务器硬件对 MySQL 性能的影响

①磁盘寻道能力（磁盘 I/O）,以目前高转速 SCSI 硬盘(7200 转/秒)为例，这种硬盘理论上每秒寻道 7200 次，这是物理特性决定的，没有办法改变。MySQL 每秒钟都在进行大量、复杂的查询操作，对磁盘的读写量可想而知。所以，通常认为磁盘 I/O 是制约 MySQL 性能的最大因素之一，对于日均访问量在 100 万 PV 以上的 Discuz!论坛，由于磁盘 I/O 的制约，MySQL 的性能会非常低下！解决这一制约因素可以考虑以下几种解决方案： 使用 RAID-0+1 磁盘阵列，注意不要尝试使用 RAID-5，MySQL 在 RAID-5 磁盘阵列上的效率不会像你期待的那样快。

②CPU 对于 MySQL 应用 ,推荐使用 S.M.P.架构的多路对称 CPU ,例如 :可以使用两颗 Intel Xeon 3.6GHz 的 CPU ,现在我较推荐用 4U 的服务器来专门做数据库服务器 ,不仅仅是针对于 mysql。

③物理内存对于一台使用 MySQL 的 Database Server 来说，服务器内存建议不要小于 2GB，推

荐使用 4GB 以上的物理内存，不过内存对于现在的服务器而言可以说是一个可以忽略的问题，工作中遇到了高端服务器基本上内存都超过了 16G。

二、MySQL 自身因素当解决了上述服务器硬件制约因素后，让我们看看 MySQL 自身的优化是如何操作的。对 MySQL 自身的优化主要是对其配置文件 my.cnf 中的各项参数进行优化调整。下面我们介绍一些对性能影响较大的参数。 由于 my.cnf 文件的优化设置是与服务器硬件配置息息相关的，因而我们指定一个假想的服务器硬件环境：CPU: 2 颗 Intel Xeon 2.4GHz 内存: 4GB DDR 硬盘: SCSI 73GB(很常见的 2U 服务器)。

下面，我们根据以上硬件配置结合一份已经优化好的 my.cnf 进行说明：

#vim /etc/my.cnf 以下只列出 my.cnf 文件中[mysqld]段落中的内容，其他段落内容对 MySQL 运行性能影响甚微，因而姑且忽略。

[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_connecti** = 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 Connecti**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。

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

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,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

thread_cache_size=120

#默认为 60

query_cache_size=32M