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，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-networking

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。

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\_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。

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。

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

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。

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

max\_heap\_table\_size = 8M

# 定义了用户可以创建的内存表(memory table)的大小。这个值用来计算内存表的最大行数值。这个变量支持动态改变

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

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\_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

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

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 = 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

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\_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

ft\_min\_word\_len = 4

# 分词词汇最小长度，默认4

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

binlog\_format = mixed

expire\_logs\_days = 30 #超过30天的binlog删除

log\_error = /data/mysql/mysql-error.log #错误日志路径

slow\_query\_log = 1

long\_query\_time = 1 #慢查询时间 超过1秒则为慢查询

slow\_query\_log\_file = /data/mysql/mysql-slow.log

performance\_schema = 0

explicit\_defaults\_for\_timestamp

#lower\_case\_table\_names = 1 #不区分大小写

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

default-storage-engine = InnoDB #默认存储引擎

innodb\_file\_per\_table = 1

# InnoDB为独立表空间模式，每个数据库的每个表都会生成一个数据空间

# 独立表空间优点：

# 1．每个表都有自已独立的表空间。

# 2．每个表的数据和索引都会存在自已的表空间中。

# 3．可以实现单表在不同的数据库中移动。

# 4．空间可以回收（除drop table操作处，表空不能自已回收）

# 缺点：

# 单表增加过大，如超过100G

# 结论：

# 共享表空间在Insert操作上少有优势。其它都没独立表空间表现好。当启用独立表空间时，请合理调整：innodb\_open\_files

innodb\_open\_files = 500

# 限制Innodb能打开的表的数据，如果库里的表特别多的情况，请增加这个。这个值默认是300

innodb\_buffer\_pool\_size = 64M

# 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.

innodb\_write\_io\_threads = 4

innodb\_read\_io\_threads = 4

# innodb使用后台线程处理数据页上的读写 I/O(输入输出)请求,根据你的 CPU 核数来更改,默认是4

# 注:这两个参数不支持动态改变,需要把该参数加入到my.cnf里，修改完后重启MySQL服务,允许值的范围从 1-64

innodb\_thread\_concurrency = 0

# 默认设置为 0,表示不限制并发数，这里推荐设置为0，更好去发挥CPU多核处理能力，提高并发量

innodb\_purge\_threads = 1

# InnoDB中的清除操作是一类定期回收无用数据的操作。在之前的几个版本中，清除操作是主线程的一部分，这意味着运行时它可能会堵塞其它的数据库操作。

# 从MySQL5.5.X版本开始，该操作运行于独立的线程中,并支持更多的并发数。用户可通过设置innodb\_purge\_threads配置参数来选择清除操作是否使用单

# 独线程,默认情况下参数设置为0(不使用单独线程),设置为 1 时表示使用单独的清除线程。建议为1

innodb\_flush\_log\_at\_trx\_commit = 2

# 0：如果innodb\_flush\_log\_at\_trx\_commit的值为0,log buffer每秒就会被刷写日志文件到磁盘，提交事务的时候不做任何操作（执行是由mysql的master thread线程来执行的。

# 主线程中每秒会将重做日志缓冲写入磁盘的重做日志文件(REDO LOG)中。不论事务是否已经提交）默认的日志文件是ib\_logfile0,ib\_logfile1

# 1：当设为默认值1的时候，每次提交事务的时候，都会将log buffer刷写到日志。

# 2：如果设为2,每次提交事务都会写日志，但并不会执行刷的操作。每秒定时会刷到日志文件。要注意的是，并不能保证100%每秒一定都会刷到磁盘，这要取决于进程的调度。

# 每次事务提交的时候将数据写入事务日志，而这里的写入仅是调用了文件系统的写入操作，而文件系统是有 缓存的，所以这个写入并不能保证数据已经写入到物理磁盘

# 默认值1是为了保证完整的ACID。当然，你可以将这个配置项设为1以外的值来换取更高的性能，但是在系统崩溃的时候，你将会丢失1秒的数据。

# 设为0的话，mysqld进程崩溃的时候，就会丢失最后1秒的事务。设为2,只有在操作系统崩溃或者断电的时候才会丢失最后1秒的数据。InnoDB在做恢复的时候会忽略这个值。

# 总结

# 设为1当然是最安全的，但性能页是最差的（相对其他两个参数而言，但不是不能接受）。如果对数据一致性和完整性要求不高，完全可以设为2，如果只最求性能，例如高并发写的日志服务器，设为0来获得更高性能

innodb\_log\_buffer\_size = 2M

# 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\_file\_size = 32M

# 此参数确定数据日志文件的大小，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\_files\_in\_group = 3

# 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3

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%

innodb\_lock\_wait\_timeout = 120

# 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

bulk\_insert\_buffer\_size = 8M

# 批量插入缓存大小， 这个参数是针对MyISAM存储引擎来说的。适用于在一次性插入100-1000+条记录时， 提高效率。默认值是8M。可以针对数据量的大小，翻倍增加。

myisam\_sort\_buffer\_size = 8M

# 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

myisam\_max\_sort\_file\_size = 10G

# 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

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)

interactive\_timeout = 28800

# 服务器关闭交互式连接前等待活动的秒数。交互式客户端定义为在mysql\_real\_connect()中使用CLIENT\_INTERACTIVE选项的客户端。默认值：28800秒（8小时）

wait\_timeout = 28800

# 服务器关闭非交互连接之前等待活动的秒数。在线程启动时，根据全局wait\_timeout值或全局interactive\_timeout值初始化会话wait\_timeout值，

# 取决于客户端类型(由mysql\_real\_connect()的连接选项CLIENT\_INTERACTIVE定义)。参数默认值：28800秒（8小时）

# MySQL服务器所支持的最大连接数是有上限的，因为每个连接的建立都会消耗内存，因此我们希望客户端在连接到MySQL Server处理完相应的操作后，

# 应该断开连接并释放占用的内存。如果你的MySQL Server有大量的闲置连接，他们不仅会白白消耗内存，而且如果连接一直在累加而不断开，

# 最终肯定会达到MySQL Server的连接上限数，这会报'too many connections'的错误。对于wait\_timeout的值设定，应该根据系统的运行情况来判断。

# 在系统运行一段时间后，可以通过show processlist命令查看当前系统的连接状态，如果发现有大量的sleep状态的连接进程，则说明该参数设置的过大，

# 可以进行适当的调整小些。要同时设置interactive\_timeout和wait\_timeout才会生效。

[mysqldump]

quick

max\_allowed\_packet = 16M #服务器发送和接受的最大包长度

[myisamchk]

key\_buffer\_size = 8M

sort\_buffer\_size = 8M

read\_buffer = 4M

write\_buffer = 4M