链滴

linux（总结）MySQL my.cnf 参数配置优化详解

作者：elijah

原文链接：https://ld246.com/article/1469023657986

来源网站：链滴

PS：本配置文件针对Dell R710，双至强E5620、16G内存的件配置。CentOS 5.6 64位系统，MySQL 5.5.x 稳定版适用于日IP 50-100w，PV 100-300w的站点，主要使用InnoDB存储引擎。其他应用环请根据实际情况来设置优化。
以下选项会被MySQL客户端应用读取。
注意只有MySQL附带的客户端应用程序保证以读取这段内容。
如果你想你自己的MySQL应用程序获取这些值。
需要在MySQL户端库初始化的时候指定这些选项。
#
[client]
#password = [your_password]
port = @MYSQL_TCP_PORT@
socket = @MYSQL_UNIX_ADDR@
# *** 应用定制选项 ***
#
MySQL 服务端
#
[mysqld]
# 一般配置选项
port = @MYSQL_TCP_PORT@
socket = @MYSQL_UNIX_ADD@
# 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&rduo; 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.<r />
# 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 = 300
# 不在TCP/IP端口上进行听.
# 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下如果没打开命名管道选项而只是用此项
# (通过 “enable-named-pipe” 选项) 将会导mysql服务没有任何作用!
#skip-networking
# MySQL 服务所允许的同时会话数的上限
# 其中一个连接将被SUPER权限保留作为管理登录.
# 即便已经达到了连接数的上限.
max_connections = 3000
# 每个客户端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&rquo;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_erros = 30

# 所有线程所打开表的数量.
# 增加此值就增加了mysqld所需要的文件描符的数量
# 这样你需要确认在[mysqld_safe]中 “open-files-limit” 变量设置开文件数量允许至少4096
table_cache = 4096
# 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一些软件依赖来锁定<a title="MyISAM" href="http://www.ha97.com/tag/myisam">MyISAM</>表
#external-locking
# 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 = 32M</>
# 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 = 4M
# 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所有的内存资源.
max_heap_table_size = 128M
# 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 “Sot_merge_passes” 状态变量.
# 在排序发生时由每个线程分配
sort_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_buf

er_size = 16M
# 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升.)
thread_cache_size = 16
# 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).
# 你可可以尝试用 [CPU数量]*(2..4) 来作为thread_concurrency的值
thread_concurrency = 8
# 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.
# 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,
# 查询缓冲也引起性能下降而不是性能提升.
query_cache_size = 128M
# 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将其他所有的查询结果都覆盖.
query_cache_limit = 4M
# 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
ft_min_word_len = 8

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
# 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,将会使用值
default_table_type = MYISAM
# 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需要更多的堆,
# 你也许需要将其设置的更高一点.
thread_stack = 512K
# 设定默认的事务隔离级别.可用的级别如下:
# READ-UNCOMMITTED, READ-COMMITED, REPEATABLE-READ, SERIALIZABLE
transaction_isolation = REPEATABLE-READ
# 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的表.
# 此限制是针对单个表的,而不是总和.
tmp_table_size = 128M
# 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=ysql-bin
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_slave_updates
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
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并仔细审查错误日志,查出可能的原因.
#log_warnings
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,
log_slow_queries
# 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”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 目前时间的确度只能达到秒的级别).
long_query_time = 6
# 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
log_long_format
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”;”来放置多个路径
他们会按照roud-robin方法被轮询使用.
#tmpdir = /tmp

*** 主从复制相关的设置
唯一的服务辨识号,数值位于 1 到 2^32-1之间.
#此值在master和slave上都需要设置.<b
/># 如果 “master-host” 没有被设置,则默认为1, 但是如果忽略此选项,MySQL不会
为master生效.
server-id
= 1
复制的Slave (去掉master段的注释来使其生效)
#
为了配置此主机作为复制的sl
ve服务器,你可以选择两种方法:
#
1) 使用 CHANGE MASTER TO 命令 (在我们的手册
有完整描述) -
语法如下:
#
# CHANGE MASTER TO MASTER_HOST=, MAST
R_PORT=,
# MASTER_USER=, MASTER_PASSWORD= ;
#
你需要替换掉 , ,
被尖括号包围的字段以及使用master的端口号替换 (默认3306).
#
例子:
#<br /
CHANGE MASTER TO MASTER_HOST=’125.564.12.1′, MASTER_PORT=3306,
br /># MASTER_USER=’joe’, MASTER_PASSWORD=’secret’;
#
或者
#
2) 设置以下的变量. 不论如何, 在你选择这种方法的情况下, 然后
一次启动复制(甚至不成功的情况下,
例如如果你输入错密码在master-password字段并且slav
无法连接),
slave会创建一个 master.info 文件,并且之后任何对于包含在此文件内的参数的变
都会被忽略
并且由 master.info 文件内的内容覆盖, 除非你关闭slave服务, 删除 master.info
并且重启slave 服务.
由于这个原因,你也许不想碰一下的配置(注释掉的) 并且使用 CHANGE
ASTER TO (查看上面) 来代替
#
所需要的唯一id号位于 2 和 2^32 – 1之间
(并且和master不同)
如果master-host被设置了.则默认值是2
但是如果省略,则
会生效
#server-id = 2
#
复制结构中的master – 必须
#master-
ost =
#
当连接到master上时slave所用来认证的用户名 – 必须
#master
user =
#
当连接到master上时slave所用来认证的密码 – 必须
#master
password =
#
master监听的端口.
可选 – 默认是3306
#maste
-port =
使得slave只读.只有用户拥有SUPER权限和在上面的slave线程能够修改数据.
你可以
用此项去保证没有应用程序会意外的修改slave而不是master上的数据
#read_only
#*** MyISAM 相关选项
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
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
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.
key_buffer_size =
28M
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.
r
ad_buffer_size = 8M
# 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
r
ad_rnd_buffer_size = 64M
# MyISAM 使用特殊的类似树的cache来使得突发插入
# (这些插入是,INSERT … SE
ECT, INSERT … VALUES (…), (…), …, 以及 LOAD DATA
INFILE)
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
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.
br />bulk_insert_buffer_size = 256M
此缓冲当MySQL需要在 REPAIR, OPTIMIZE, ALTER 以及 LOAD DATA INFILE 到一个空表中
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.
myisam_sor
_buffer_size = 256M
# 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_f
le_size = 10G
# 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.
myisam_max_extra_
ort_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

默认关闭 Federated
skip-federated
*** BDB 相关选项 ***
如果你运行的MySQL服务有BDB支持但是你不准备使用的时候使用此选项. 这会节省内存并且能加速一些事.
skip-bdb
*** INNODB 相关选项 ***
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以及磁盘空间,并且加速某些部分
#skip-innodb
# 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, 你一般需要修改此值.
# SHOW INNODB STATUS 命令会显示当先使用的数量.
innodb_additional_mem_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,
# 所不要设置的太高.
innodb_buffer_pool_size = 6G
# 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
innodb_data_file_path = ibdata1:10M:autoextend
# 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.
#innodb_data_home_dir =
# 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可能在一个数值下表现的更好.
innodb_file_io_threads = 4
# 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开并且增加此值知道你能够成功的导出表.
#innodb_force_recovery=1
# 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.
# 过高的值可能导致线程的互斥颠簸.
innodb_thread_concurrency = 16
# 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CID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.
# 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.
nnodb_flush_log_at_trx_commit = 2
（说明：如果是游戏服务器，建议此值设置2；如果是对数据安全要求极高的应用，建议设置为1；设置为0性能最高，但如果发生故障，数据可会有丢失的危险！默认值1的意思是每一次事务提交或事务外的指令都需要把日志写入（flush）硬盘这是很费时的。特别是使用电池供电缓存（Battery backed up cache）时。设成2对于很多运用，特是从MyISAM表转过来的是可以的，它的意思是不写入硬盘而是写入系统缓存。日志仍然会每秒flush硬盘，所以你一般不会丢失超过1-2秒的更新。设成0会更快一点，但安全方面比较差，即使MySQL了也可能会丢失事务的数据。而值2只会在整个操作系统挂了时才可能丢数据。）
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_shutdon
# 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.<r /># 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)
innodb_log_buffer_size = 16M
# 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, 请注意一个大日志文件大小会增加恢复进程所需要的时间.
innodb_log_file_size = 512M
# 在日志组中的文件总数.
# 通常来说2~3是比较好的.
innodb_log_files_in_group 3
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盘上或者一个RAID1卷上来提高其性能
#innodb_log_group_home_dir
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

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.
innodb_max_dirty_pages_pct = 90
# 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”.
#innodb_flush_method=ODSYNC
# 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nnoDB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.
innodb_lock_wait_timeout = 12
[mysqldump]
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此项
quick
max_allowed_packet = 32M
[mysql]
no-auto-rehash
仅仅允许使用键值的 UPDATEs 和 DELETEs .
#safe-updates
[isamchk]
key_buffer = 2048M
sort_buffer_size = 2048M
read_buffer = 32M
write_buffer = 32M
[myisamchk]
key_buffer = 2048M
sort_buffer_size = 2048M
read_buffer = 32M
write_buffer = 32M
[mysqlhotcopy]
interactive-timeout
[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设定的足够高!
打开大量表需要将此值设大
open-files-limit = 8192
