链滴

mysql5.6—配置详解

作者：wuhw

原文链接：https://ld246.com/article/1535348649337

来源网站：链滴

说明

1，红色代表重点参数

2，“全局缓存”、“线程缓存”，全局缓存是所有线程共享，线程缓存是每个线程连接上数据时创一个线程（如果没有设置线程池），假如有200连接、那就是200个线程，如果参数设定值是10M，么参数值就是10*200=2000M=2G，很有可能吃垮内存，所以线程缓存并不是像全局缓存一样设定少就占用多少，不可设置过大，一定注意

character-set-server=utf8

#服务器端字符集

collation-server=utf8_bin

#字符集的校队规则，这里是个坑，新手得小心

#默认***_ci，ci是 case insensitive, 即 "大小写不敏感", a 和 A 会在字符判断中会被当做一样的;

***_bin是二进制数据编译存储，大小写敏感，

lower_case_table_names = 0

#默认为0，数据库、表名大小写敏感

back_log = 600

#在MYSQL暂时停止响应新请求之前，短时间内的多少个请求可以被存在堆栈中。如果系统在短时间有很多连接，则需要增大该参数的值，该参数值指定到来的TCP/IP连接的监听队列的大小。默认值50。

max_connections = 3000

#MySQL允许最大的进程连接数，如果经常出现Too Many Connections的错误提示，则需要增大此，但是该值越大，占用内存越大。

max_connect_errors = 50

#设置每个主机的连接请求异常中断的最大次数，当超过该次数，MYSQL服务器将禁止host的连接请，直到mysql服务器重启或通过flush hosts命令清空此host的相关信息。

wait_timeout=864000

#wait_timeout的初始值是28800，当应用程序持续8小时没有连接，会启动切断与应用程序连接池的联，再连接时，会提示wait_timeout错误。

external-locking = FALSE

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

max_allowed_packet = 32M

#设置在网络传输中一次消息传输量的最大值。系统默认值 为1MB，最大值是1GB，必须设置1024倍数。

skip-name-resolve

#禁用DNS反向解析，唯一的局限是之后GRANT语句中只能使用IP地址了，因此在添加这项设置到一已有系统中必须注意

slow_query_log=on

#打开慢查询记录

slow_query_log_file=mysql-slow

#慢查询记录日志

long_query_time = 1

#记录执行时间超过N（秒）的查询

server-id = 1

#主从复制时必须设置的参数，并且不能和其他机器重复

log-bin=mysql-bin

#二进制日志

binlog_cache_size = 4M

#【**线程缓存**】 为每个session 分配的内存，在事务过程中用来存储二进制日志的缓存。

#show global status like 'bin%';

#上述语句我们可以得到当前 数据库binlog_cache_size的使用情况

#+-----------------------+-------+

#| Variable_name | Value |

#+-----------------------+-------+

#| Binlog_cache_disk_use | ???

#| Binlog_cache_use | ?????

#Binlog_cache_disk_use表示因为我们binlog_cache_size设计的内存不足导致缓存二进制日志用到临时文件的次数

#Binlog_cache_use 表示 用binlog_cache_size缓存的次数

#当对应的Binlog_cache_disk_use 值比较大的时候 我们可以考虑适当的调高 binlog_cache_size 对的值

max_binlog_cache_size =2M

#表示的是binlog 能够使用的最大cache 内存大小，这个默认就可以

max_binlog_size = 512M

#单binlog文件最大容量，超过则建立新binlog文件

expire_logs_days = 7

#二进制文件自动删除天数，默认为0，表示不删除

#mysql使用flush logs的操作来清除日志，下面几种情况会触发flush logs

#1. 重启

#2. BINLOG文件大小达到参数max_binlog_size限制

#3. 手工执行命令。

#双主复制、级联复制时，备主或者级联节点将所有的操作写入到binlog

log-slave-updates

#中继日志路径

relay-log = relay-bin

#当slave从库宕机后，假如relay-log损坏了，导致一部分中继日志没有处理，则自动放弃所有未执行

relay-log，并且重新从master上获取日志，这样就保证了relay-log的完整性。默认情况下该功能是闭的，将relay_log_recovery的值设置为 1时，可在slave从库上开启该功能，建议开启。

#relay_log_recovery =1

#这个参数和sync_binlog是一样的，当设置为1时，slave的I/O线程每次接收到master发送过来的binog日志都要写入系统缓冲区，然后刷入relay log中继日志里，这样是最安全的，因为在崩溃的时候你最多会丢失一个事务，但会造成磁盘的大量I/O。当设置为0时，并不是马上就刷入中继日志里，而由操作系统决定何时来写入，虽然安全性降低了，但减少了大量的磁盘I/O操作。这个值默认是0，可态修改，建议采用默认值。

#sync_relay_log=0

#这个参数和sync_relay_log参数一样，当设置为1时，slave的I/O线程每次接收到master发送过来的bnlog日志都要写入系统缓冲区，然后刷入relay-log.info里，这样是最安全的，因为在崩溃的时候，最多会丢失一个事务，但会造成磁盘的大量I/O。当设置为0时，并不是马上就刷入relay-log.info里而是由操作系统决定何时来写入，虽然安全性降低了，但减少了大量的磁盘I/O操作。这个值默认是0可动态修改，建议采用默认值。

#sync_relay_log_info=0

default-storage-engine = InnoDB

#默认引擎

transaction_isolation = READ-COMMITTED

#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4.SERIALIZABLE -串行

table_cache = 2000

#设置最大缓存表的个数。默认2000，全局性参数， 缓存表的信息，包括字段、索引等。 使用表缓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show status like 'open%tables%'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，也需要加table_cache。

tmp_table_size = 20M

#**【线程缓存】** 它规定了内部内存临时表的最大值，每个线程都要分配。（实际起限制作用的是tmp_table_size和max_heap_table_size的最小值。）如果临时表超出了限制，MySQL就会自动地把它转化基于磁盘的MyISAM表，而内存表则不会。

#如果使用临时表情况少，可以默认

max_heap_table_size = 256M

#如果不使用内存表，可以默认

thread_cache_size = 64

#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.过比较 Connections 和 Threads_created 状态的变量，可以看到这个变量的作用。设置规则如下：1B 内存配置为8，2GB配置为16，3GB配置为32，4GB或更高内存，可配置更大。

thread_concurrency = 8

#hread_concurrency变量是针对于Solaris 8及低版本的系统，设置了这个变量mysqld会调用thr_setoncurrency()函数。这个函数允许应用程序给同一时间运行的线程系统提示所需数量的线程。当前的Slaris 版本中这个参数已经没有作用了。这个参数在mysql 5.6.1中已经被标记为过时，在5.7.2版本的msql中被移除。

query_cache_size = 64M

#【全局缓存】 前面的文章专门讲过这个参数，该参数不适合分配过大。而且在高并发，写入量大的统，建议把该功能禁掉。

query_cache_limit = 4M

#指定单个查询能够使用的缓冲区大小，缺省为1M

query_cache_min_res_unit = 2k

#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_ache_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8%以上而且Qcache_lowmem_prunes > 50的话说明query_cache_size可能有点小，要不就是碎片多。

#查询缓存命中率 = (Qcache_hits – Qcache_inserts) / Qcache_hits * 100%

read_buffer_size = 1M

#**【线程缓存】** 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。和srt_buffer_size一样，该参数对应的分配内存也是每个连接独享。

read_rnd_buffer_size = 16M

#**【线程缓存】**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，以应尽量适当设置该值，以避免内存开销过大。

sort_buffer_size = 2M

#**【线程缓存】** Sort_Buffer_Size 是一个connection级参数，在每个connection（session）第一次要使用这个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+高并发可能会耗尽统内存资源。例如：500个连接将会消耗 500sort_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。

#explain selectfrom table where order limit；出现filesort

max_length_for_sort_data= 1024

#mysql排序使用sort_buffer_size和max_length_for_sort_data两种方式，如果查询列和orderby列长度和值会大于1024，将使用max_length_for_sort_data，

#可以默认

join_buffer_size = 2M

#【**线程缓存**】 用于表间关联缓存的大小，和sort_buffer_size一样，该参数对应的分配内存也是每个接独享。

bulk_insert_buffer_size = 64M

#多值的 INSERT 或者 LOAD DATA 是往一个非空的数据表里增加记录，通过调整该参数可以有效提插入效率，默认为8M

key_buffer_size = 2048M

#全局缓存】 针对MyISAM引擎，批定用于索引的缓冲区大小，增加它可以得到更好的索引处理性能对于内存在4GB左右的服务器来说，该参数可设置为256MB或384MB。

myisam_sort_buffer_size = 128M

#MyISAM表发生变化时重新排序所需的缓冲

myisam_max_sort_file_size = 10G

#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 INFLE).

#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_extra_sort_file_size = 10G

myisam_repair_threads = 1

#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_recover

#自动检查和修复没有适当关闭的 MyISAM 表

innodb_buffer_pool_size = 2048M

#【全局缓存】 这对Innodb表来说非常重要。Innodb相比MyISAM表对缓冲更为敏感。MyISAM可在默认的 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，因此如果只要用Innodb的话则可以设置它高达 70-80% 的可用内存（注意，这里是**可用**，不是内存总量）。

innodb_additional_mem_pool_size = 16M

#全局缓存】 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，类于Oracle的library cache。

#可以默认

innodb_data_file_path = ibdata1:1024M:autoextend

#表空间文件 重要数据

#可以默认

innodb_read_io_threads

#读线程数，默认为4

innodb_write_io_threads

#写线程数，默认为4

innodb_thread_concurrency =0

#并发线程数量，默认是0即不限制，取值范围0-1000，不可动态修改

#可以默认

innodb_flush_log_at_trx_commit = 1

#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，但承担在发生故障时丢失数据的风险。

#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，这是很费时的设成2对于很多运用，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到硬 盘，所以一般不会丢失超过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挂了也可能会丢失事务的数据；设置成2只会在个操作系统 挂了时才可能丢数据；设置成1是最安全的设置，性能也是相对最弱的。

innodb_log_buffer_size = 1M

#【全局缓存】 这项配置决定了为尚未执行的事务分配的缓存。其默认值（1MB）一般来说已经够用，但是如果你的事务中包含有二进制大对象或者大文本字段的话，这点缓存很快就会被填满并触发额的I/O操作。看看Innodb_log_waits状态变量，如果它不是0，增加innodb_log_buffer_size。MySQ开发人员建议设置为1 - 8M之间

innodb_log_file_size = 500M

#redo日志的大小，redo日志被用于确保写操作快速而可靠并且在崩溃时恢复。一直到MySQL 5.5，rdo日志的总尺寸被限定在4GB(默认可以有2个log文件)。这在MySQL 5.6里被提高。如果把innodb_lg_file_size设置成512M(这样有1GB的redo日志)会使你有充裕的写操作空间。一般设置为256~512M

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。默认为2，推荐设置为3M

innodb_max_dirty_pages_pct = 75

#Buffer_Pool中Dirty_Page（脏页）所占的数量，直接影响InnoDB的关闭时间。参数innodb_max_drty_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小，强制数据块Flush一段时间，则能够大大缩短 MySQL关闭的时间。

#可以默认

innodb_lock_wait_timeout = 50

#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MyISA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_lck_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前要等待多长时间(秒数)

#当事务等待一个锁的时间 ，如果超过时间则回滚

#根据需要设置等待时间

innodb_print_all_deadlocks = 1

#将死锁信息打印到错误日志里。5.6版本开始的才有的参数

innodb_file_per_table = 0

#独立表空间的设置，默认为0，则为共享表空间；如果为1，则为独立表空间

#建议打开, 尤其是在数据量比较大的，ibdata1文件只会增大，不会减小，大到一定程度会影响insert update 速度，另外如果删表频繁的话，共享表空间产生的碎片会比较多，并且无法向OS回收空间。

innodb_buffer_pool_load_at_startup = 1

#默认为关闭OFF。如果开启该参数，启动MySQL服务时，MySQL将本地热数据加载到InnoDB缓冲中。

#在MySQL启动时会延长启动时间

innodb_buffer_pool_dump_at_shutdown = 1

#默认为关闭OFF。如果开启该参数，停止MySQL服务时，InnoDB将InnoDB缓冲池中的热数据保存本地硬盘。

#在MySQL关闭时会延长关闭时间

[mysqldump]

quick

max_allowed_packet = 32M

[mysqld_safe]

log-error=/data/3306/mysql_oldboy.err

pid-file=/data/3306/mysqld.pid