MYSQL服务器my.cnf配置文档详解

硬件：内存16G

[client]

port = 3306

socket = /data/3306/mysql.sock

[mysql]

no-auto-rehash

[mysqld]

user = mysql

port = 3306

socket = /data/3306/mysql.sock

basedir = /usr/local/mysql

datadir = /data/3306/data

open\_files\_limit    = 10240

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，则需要增大此值。

max\_connect\_errors = 6000

#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的连接请求，直到mysql服务器重启或通过flush hosts命令清空此host的相关信息。

table\_cache = 614

#指示表调整缓冲区大小。# 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。

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的倍数。

sort\_buffer\_size = 2M

# 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

join\_buffer\_size = 2M

#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连接独享。

thread\_cache\_size = 300

# 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。设置规则如下：1GB 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

# 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

query\_cache\_size = 64M

## 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，建议把该功能禁掉。

#重点优化参数（主库 增删改-MyISAM）

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%

default-storage-engine = MyISAM

#default\_table\_type = InnoDB

thread\_stack = 192K

#设置MYSQL每个线程的堆栈大小，默认值足够大，可满足普通操作。可设置范围为128K至4GB，默认为192KB。

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 -可重复读4.SERIALIZABLE -串行

tmp\_table\_size = 256M

# 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。如果超过该值，则会将临时表写入磁盘。

max\_heap\_table\_size = 256M

long\_query\_time = 2

log\_long\_format

log-slow-queries=/data/3306/slow-log.log

#log-bin = /data/3306/mysql-bin

log-bin

binlog\_cache\_size = 4M

max\_binlog\_cache\_size = 8M

max\_binlog\_size = 512M

expire\_logs\_days = 7

key\_buffer\_size = 2048M

#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。

read\_buffer\_size = 1M

# 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。和sort\_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

# 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。

bulk\_insert\_buffer\_size = 64M

#批量插入数据缓存大小，可以有效提高插入效率，默认为8M

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 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\_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

skip-name-resolve

lower\_case\_table\_names = 1

server-id = 1

innodb\_additional\_mem\_pool\_size = 16M

#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\_buffer\_pool\_size = 2048M

# 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\_data\_file\_path = ibdata1:1024M:autoextend

#表空间文件 重要数据

innodb\_file\_io\_threads = 4

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

innodb\_thread\_concurrency = 8

#服务器有几个CPU就设置为几，建议用默认设置，一般为8.

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

# 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\_log\_buffer\_size = 16M

#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 = 128M

#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\_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3M

innodb\_max\_dirty\_pages\_pct = 90

#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\_lock\_wait\_timeout = 120

# 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\_file\_per\_table = 0

#独享表空间（关闭）

[mysqldump]

quick

max\_allowed\_packet = 32M

[mysqld\_safe]

log-error=/data/3306/mysql\_oldboy.err

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

#补充

#wait\_timeout = 10

#指定一个请求的最大连接时间，对于4GB左右的内存服务器来说，可以将其设置为5-10。

#skip\_networking

#开启该选可以彻底关闭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数据库服务器的，则不要开启该选项，否则将无法正常连接。

#log-queries-not-using-indexes

将没有使用索引的查询也记录下来