# 服务器物理硬件的优化

在挑选硬件服务器时，我们应该从下面几个方面着重对MySQL服务器的硬件配置进行优化，也就是说将项目中的资金着重投入到如下几处：

1、磁盘寻道能力（磁盘I/O），我们现在用的都是SAS15000转的硬盘，用6快这样的硬盘作RAID1+0。MySQL每一秒钟都在进行大量、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是约制MySQL性能的最大因素之一。对于日均访问量在100万PV以上的论坛（Discuz）、博客（Wordpress），如果性能不好，造成的直接后果就是MySQL的性能会非常低下！解决这一制约因素可以考虑解决访问是：使用RAID1+0磁盘阵列，注意不要尝试RAID5，MySQL在PAID5磁盘阵列上的效率不会像你期待的那样快，如果资金允许，可以选择固态硬盘SSD来替代SAS硬盘做RAID1+0。

2、CPU对于MySQL的影响也是不容忽视的，建议选择运算能力强悍的CPU。推荐使用DELL R710（双四核），商家的卖点也是其强大的虚拟化和数据库能力。

3、对于一台使用MySQL的Database Server来说，建议服务器的内存不要小于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，如果是高端服务器，基本上内存都超过了32GB，我们的数据流服务器都是32GB内存。

我在工作中用的比较多的数据库服务器是DELL R710/R720，其稳定性和性能都不错，同时我也发现许多同行都是采用它作为数据库的服务器，所以在这里也向大家推荐一下。

# MySQL应采用编译方式安装

关于MySQL数据库的线上环境安装，我建议采取编译安装的方式，这样性能会有较大的提升。服务器系统则建议CentOS6.7 X86\_64，源码包的编译参数会默认以Debug模式生成二进制代码，而Debug模式给MySQL带来的性能损失是比较大的，所以当我们编译准备安装的产品代码时，一定不要忘记使用--without-debug参数禁止Debug模式。如果把--with-mysqld-ldflags和--with-client-ld-flags两个编译参数设置为--all-static的话，可以告诉编译器以静态的方式编译，编译结果将得到最高的性能。使用静态编译和使用动态编译的代码相比，性能差距可能会达到5%至10%之多。在后面我会跟大家分享我们线上MySQL数据库的编译参数，大家可以参考下，然后根据自己的线上环境自行修改内容。

# MySQL配置文件优化

[client]

port = 3306 #客户端端口号为3306

socket =/data/3306/mysql.sock #

default-character-set = utf8 #客户端字符集

(控制character\_set\_client、character\_set\_connection、character\_set\_results)

[mysql]

no-auto-rehash #仅仅允许使用键值的updates和deletes

[mysqld] #组包括了mysqld服务启动的参数，它涉及的方面很多，其中有MySQL的目录和文件，通信、网络、信息安全，内存管理、优化、查询缓存区，还有MySQL日志设置等。

user = mysql #mysql\_safe脚本使用MySQL运行用户(编译时--user=mysql指定),推荐使用mysql用户。

port = 3306 #MySQL服务运行时的端口号。建议更改默认端口,默认容易遭受攻击。

socket =/data/3306/mysql.sock #socket文件是在Linux/Unix环境下特有的，用户在Linux/Unix环境下客户端连接可以不通过TCP/IP网络而直接使用unix socket连接MySQL。

basedir = /application/mysql #mysql程序所存放路径,常用于存放mysql启动、配置文件、日志等

datadir = /data/3306/data #MySQL数据存放文件(极其重要)

character-set-server = utf8 #数据库和数据库表的默认字符集。(推荐utf8,以免导致乱码)

log-error=/data/3306/mysql\_xuliangwei.err #mysql错误日志存放路径及名称(启动出现错误一定要看错误日志,百分之百都能通过错误日志排插解决。)

pid-file=/data/3306/mysql\_xuliangwei.pid #MySQL\_pid文件记录的是当前mysqld进程的pid，pid亦即ProcessID。

skip-locking #避免MySQL的外部锁定，减少出错几率，增强稳定性。

skip-name-resolv #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解析的时候。但是需要注意的是，如果开启该选项，则所有远程主机连接授权都要使用IP地址方式了，否则MySQL将无法正常处理连接请求！

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数据库服务器的，则不要开启该选项，否则无法正常连接！

open\_files\_limit = 1024 #MySQLd能打开文件的最大个数,如果出现too mant openfiles之类的就需要调整该值了。

back\_log = 384 #back\_log参数是值指出在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，则需要增加该参数的值，该参数值指定到来的TCP/IP连接的监听队列的大小。不同的操作系统在这个队列的大小上有自己的限制。如果试图将back\_log设置得高于操作系统的限制将是无效的，其默认值为50.对于Linux系统而言，推荐设置为小于512的整数。

max\_connections = 800 #指定MySQL允许的最大连接进程数。如果在访问博客时经常出现 Too Many Connections的错误提示，则需要增大该参数值。

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。

wait\_timeout = 120 #指定一个请求的最大连接时间，对于4GB左右内存的服务器来说，可以将其设置为5~10。

table\_cache = 614K #table\_cache指示表高速缓冲区的大小。当MySQL访问一个表时，如果在MySQL缓冲区还有空间，那么这个表就被打开并放入表缓冲区，这样做的好处是可以更快速地访问表中的内容。一般来说，可以查看数据库运行峰值时间的状态值Open\_tables和Open\_tables，用以判断是否需要增加table\_cache的值，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。

external-locking = FALSE #MySQL选项可以避免外部锁定。True为开启。

max\_allowed\_packet =16M #服务器一次能处理最大的查询包的值，也是服务器程序能够处理的最大查询

sort\_buffer\_size = 1M #设置查询排序时所能使用的缓冲区大小，系统默认大小为2MB。

注意：该参数对应的分配内存是每个连接独占的，如果有100个连接，那么实际分配的总排序缓冲区大小为100 x6=600MB。所以，对于内存在4GB左右的服务器来说，推荐将其设置为6MB~8MB

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个连接独享。

thread\_cache\_size = 64 #设置Thread Cache池中可以缓存的连接线程最大数量，可设置为0~16384，默认为0.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。我们可以根据物理内存设置规则如下:1GB内存我们配置为8,2GB内存我们配置为16,3GB我们配置为32,4GB或4GB以上我们给此值为64或更大的值。

thread\_concurrency = 8 #该参数取值为服务器逻辑CPU数量x 2，在本例中，服务器有两个物理CPU，而每个物理CPU又支持H.T超线程，所以实际取值为4 x 2 = 8。这也是双四核主流服务器的配置。

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得非常频繁。另外如果改值较小反而会影响效率，那么可以考虑不用查询缓冲。对于Qcache\_free\_blocks，如果该值非常大，则表明缓冲区中碎片很多。

query\_cache\_limit = 2M #只有小于此设置值的结果才会被缓存

query\_cache\_min\_res\_unit = 2k #设置查询缓存分配内存的最小单位，要适当第设置此参数，可以做到为减少内存快的申请和分配次数，但是设置过大可能导致内存碎片数值上升。默认值为4K，建议设置为1K~16K。

default\_table\_type = InnoDB #默认表的类型为InnoDB

thread\_stack = 256K #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128KB至4GB，默认为192KB

#transaction\_isolation = Level #数据库隔离级别 (READ UNCOMMITTED(读取未提交内容) READ COMMITTED(读取提交内容) REPEATABLE READ(可重读) SERIALIZABLE(可串行化))

tmp\_table\_size = 64M #设置内存临时表最大值。如果超过该值，则会将临时表写入磁盘，其范围1KB到4GB。

max\_heap\_table\_size = 64M #独立的内存表所允许的最大容量。

table\_cache = 614 #给经常访问的表分配的内存，物理内存越大，设置就越大。调大这个值，一般情况下可以降低磁盘IO，但相应的会占用更多的内存,这里设置为614。

table\_open\_cache = 512 #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。

long\_query\_time = 1 #慢查询的执行用时上限,默认设置是10s,推荐(1s~2s)

log\_long\_format #没有使用索引的查询也会被记录。(推荐,根据业务来调整)

log-slow-queries = /data/3306/slow.log #慢查询日志文件路径(如果开启慢查询,建议打开此日志)

log-bin = /data/3306/mysql-bin #logbin数据库的操作日志,例如update、delete、create等都会存储到binlog日志,通过logbin可以实现增量恢复

relay-log = /data/3306/relay-bin #relay-log日志记录的是从服务器I/O线程将主服务器的二进制日志读取过来记录到从服务器本地文件,然后SQL线程会读取relay-log日志的内容并应用到从服务器

relay-log-info-file = /data/3306/relay-log.info #从服务器用于记录中继日志相关信息的文件,默认名为数据目录中的relay-log.info。

binlog\_cache\_size = 4M #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，可以增加此值来获取更大的性能，所有从事务来的状态都被缓冲在binlog缓冲中，然后再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，此缓冲在每个链接的事务第一次更新状态时被创建。

max\_binlog\_cache\_size = 8M #最大的二进制Cache日志缓冲尺寸。

max\_binlog\_size = 1G #二进制日志文件的最大长度(默认设置1GB)一个二进制文件信息超过了这个最大长度之前,MySQL服务器会自动提供一个新的二进制日志文件接续上。

expire\_logs\_days = 7 #超过7天的binlog,mysql程序自动删除(如果数据重要,建议不要开启该选项)

key\_buffer\_size = 256M #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左右的服务器来说，该参数可设置为256MB或384MB。

注意：如果该参数值设置得过大反而会使服务器的整体效率降低！

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个连接独享。

read\_rnd\_buffer\_size = 16M #设置进行随机读的时候所使用的缓冲区。此参数和read\_buffer\_size所设置的Buffer相反，一个是顺序读的时候使用，一个是随机读的时候使用。但是两者都是针对与线程的设置，每个线程都可以产生两种Buffer中的任何一个。默认值256KB，最大值4GB。

bulk\_insert\_buffer\_size = 8M #如果经常性的需要使用批量插入的特殊语句来插入数据,可以适当调整参数至16MB~32MB,建议8MB。

#myisam\_sort\_buffer\_size = 8M #设置在REPAIR Table或用Create index创建索引或 Alter table的过程中排序索引所分配的缓冲区大小，可设置范围4Bytes至4GB，默认为8MB

lower\_case\_table\_names = 1 #实现MySQL不区分大小。(发开需求-建议开启)

slave-skip-errors = 1032,1062 #从库可以跳过的错误数字值(mysql错误以数字代码反馈,全的mysql错误代码大全,以后会发布至博客)。

replicate-ignore-db=mysql #在做主从的情况下,设置不需要同步的库。

server-id = 1 #表示本机的序列号为1,如果做主从，或者多实例,serverid一定不能相同。

myisam\_sort\_buffer\_size = 128M #当需要对于执行REPAIR, OPTIMIZE, ALTER 语句重建索引时，MySQL会分配这个缓存，以及LOAD DATA INFILE会加载到一个新表，它会根据最大的配置认真的分配的每个线程。

myisam\_max\_sort\_file\_size = 10G #当重新建索引（REPAIR，ALTER，TABLE，或者LOAD，DATA，TNFILE）时，MySQL被允许使用临时文件的最大值。

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.

myisam\_recover #自动检查和修复没有适当关闭的 MyISAM 表.

innodb\_additional\_mem\_pool\_size = 4M #用来设置InnoDB存储的数据目录信息和其他内部数据结构的内存池大小。应用程序里的表越多，你需要在这里面分配越多的内存。对于一个相对稳定的应用，这个参数的大小也是相对稳定的，也没有必要预留非常大的值。如果InnoDB用广了这个池内的内存，InnoDB开始从操作系统分配内存，并且往MySQL错误日志写警告信息。默认为1MB，当发现错误日志中已经有相关的警告信息时，就应该适当的增加该参数的大小。

innodb\_buffer\_pool\_size = 64M #InnoDB使用一个缓冲池来保存索引和原始数据，设置越大，在存取表里面数据时所需要的磁盘I/O越少。强烈建议不要武断地将InnoDB的Buffer Pool值配置为物理内存的50%~80%，应根据具体环境而定。

innodb\_data\_file\_path = ibdata1:128M:autoextend #设置配置一个可扩展大小的尺寸为128MB的单独文件，名为ibdata1.没有给出文件的位置，所以默认的是在MySQL的数据目录内。

innodb\_file\_io\_threads = 4 #InnoDB中的文件I/O线程。通常设置为4，如果是windows可以设置更大的值以提高磁盘I/O

innodb\_thread\_concurrency = 8 #你的服务器有几个CPU就设置为几，建议用默认设置，一般设为8。

innodb\_flush\_log\_at\_trx\_commit = 1 #设置为0就等于innodb\_log\_buffer\_size队列满后在统一存储，默认为1，也是最安全的设置。

innodb\_log\_buffer\_size = 2M #默认为1MB，通常设置为8~16MB就足够了。

innodb\_log\_file\_size = 32M #确定日志文件的大小，更大的设置可以提高性能，但也会增加恢复数据库的时间。

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。

innodb\_lock\_wait\_timeout = 120 #InnoDB事务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locak tables 语句注意到锁定设置。默认值是50秒。

innodb\_file\_per\_table = 0 #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。0关闭，1开启。

独立表空间优点：

1、每个表都有自己独立的表空间。

2、每个表的数据和索引都会存在自己的表空间中。

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，表空不能自己回收。）

[mysqldump]

quick

max\_allowed\_packet = 2M #设定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，必须设置为1024的倍数。单位为字节。

[mysqld\_safe]

值得注意：

强烈建议不要武断地将InnoDB的Buffer Pool值配置为物理内存的50%~80%，应根据具体环境而定。

如果key\_reads太大，则应该把my.cnf中的key\_buffer\_size变大，保持key\_reads/key\_read\_re-quests至少在1/100以上，越小越好。

如果qcache\_lowmem\_prunes很大，就要增加query\_cache\_size的值。

不过很多时候需要具体情况具体分析，其他参数的变更我们可以等MySQL上线稳定一段时间后在根据status值进行调整。

# 电商MySQL数据库配置文件

这是一份电子商务网站MySQL数据库调整后所运行的配置文件/etc/my.cnf(服务器为DELL R710、16GB内存、RAID10)，大家可以根据实际的MySQL数据库硬件情况进行调整配置文件如下：

[client]

port = 3306

socket =/data/3306/mysql.sock

default-character-set = utf8

[mysqld]

user = mysql

port = 3306

character-set-server = utf8

socket =/data/3306/mysql.sock

basedir = /application/mysql

datadir = /data/3306/data

log-error=/data/3306/mysql\_err.log

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

log\_slave\_updates = 1

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

binlog\_format = mixed

binlog\_cache\_size = 4M

max\_binlog\_cache\_size = 8M

max\_binlog\_size = 1G

expire\_logs\_days = 90

binlog-ignore - db = mysql

binlog-ignore - db = information\_schema

key\_buffer\_size = 384M

sort\_buffer\_size = 2M

read\_buffer\_size = 2M

read\_rnd\_buffer\_size = 16M

join\_buffer\_size = 2M

thread\_cache\_size = 8

query\_cache\_size = 32M

query\_cache\_limit = 2M

query\_cache\_min\_res\_unit = 2k

thread\_concurrency = 32

table\_cache = 614

table\_open\_cache = 512

open\_files\_limit = 10240

back\_log = 600

max\_connections = 5000

max\_connect\_errors = 6000

external-locking = FALSE

max\_allowed\_packet =16M

thread\_stack = 192K

transaction\_isolation = READ-COMMITTED

tmp\_table\_size = 256M

max\_heap\_table\_size = 512M

bulk\_insert\_buffer\_size = 64M

myisam\_sort\_buffer\_size = 64M

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

myisam\_repair\_threads = 1

myisam\_recover

long\_query\_time = 2

slow\_query\_log

slow\_query\_log\_file = /data/3306/slow.log

skip-name-resolv

skip-locking

skip-networking

server-id = 1

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

innodb\_buffer\_pool\_size = 512M

innodb\_data\_file\_path = ibdata1:256M:autoextend

innodb\_file\_io\_threads = 4

innodb\_thread\_concurrency = 8

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

innodb\_log\_buffer\_size = 16M

innodb\_log\_file\_size = 128M

innodb\_log\_files\_in\_group = 3

innodb\_max\_dirty\_pages\_pct = 90

innodb\_lock\_wait\_timeout = 120

innodb\_file\_per\_table = 0

[mysqldump]

quick

max\_allowed\_packet = 64M

[mysql]

no – auto - rehash

# MySQL上线后根据status状态进行优化

MySQL数据库上线后，可以等其稳定运行一段时间后再根据服务器的status状态进行适当优化，我们可以用如下命令列出MySQL服务器运行的各种状态值：

mysql > show global status;

我个人比较喜欢的用法是 show status like '查询%';

## 1.慢查询

有时我们为了定位系统中效率比较低下的Query语法，需要打开慢查询日志，也就是Slow Que-ry log。打开慢查询日志的相关命令如下：

mysql> show variableslike '%slow%';

+---------------------+-----------------------------------------+

| Variable\_name |Value |

+---------------------+-----------------------------------------+

| log\_slow\_queries | ON |

| slow\_launch\_time | 2 |

+---------------------+-----------------------------------------+

mysql> show globalstatus like '%slow%';

+---------------------+-------+

| Variable\_name | Value|

+---------------------+-------+

| Slow\_launch\_threads | 0 |

| Slow\_queries | 2128 |

+---------------------+-------+

打开慢查询日志可能会对系统性能有一点点影响，如果你的MySQL是主从结构，可以考虑打开其中一台从服务器的慢查询日志，这样既可以监控慢查询，对系统性能影响也会很小。另外，可以用MySQL自带的命令mysqldumpslow进行查询。比如：下面的命令可以查出访问次数最多的20个SQL语句：

mysqldumpslow-s c -t 20 host-slow.log

## 2.连接数

我们如果经常遇见MySQL：ERROR1040：Too manyconnections的情况，一种情况是访问量确实很高，MySQL服务器扛不住了，这个时候就要考虑增加从服务器分散读压力。另外一种情况是MySQL配置文件中max\_connections的值过小。来看一个例子。

mysql>show variables like'max\_connections';

+-----------------+-------+

| Variable\_name | Value |

+-----------------+-------+

| max\_connections | 800 |

+-----------------+-------+

这台服务器最大连接数是256，然后查询一下该服务器响应的最大连接数；

mysql> show global status like 'Max\_used\_connections';

+----------------------+-------+

| Variable\_name | Value|

+----------------------+-------+

| Max\_used\_connections | 245 |

+----------------------+-------+

MySQL服务器过去的最大连接数是245，没有达到服务器连接数的上线800，不会出现1040错误。

Max\_used\_connections/max\_connections \* 100% = 85%

最大连接数占上限连接数的85%左右,如果发现比例在10%以下，则说明MySQL服务器连接数的上限设置得过高了。

## 3.key\_buffer\_size

key\_buffer\_size是设置MyISAM表索引缓存空间的大小，此参数对MyISAM表性能影响最大。下面是一台MyISAM为主要存储引擎服务器的配置：

mysql> show variables like 'key\_buffer\_size';

+-----------------+-----------+

| Variable\_name | Value |

+-----------------+-----------+

| key\_buffer\_size | 536870912 |

+-----------------+-----------+

从上面可以看出，分配了512MB内存给key\_buffer\_size。再来看key\_buffer\_size的使用情况：

mysql> show global status like 'key\_read%';

+-------------------+--------------+

| Variable\_name | Value |

+-------------------+-------+

| Key\_read\_requests | 27813678766 |

| Key\_reads | 6798830 |

+-------------------+--------------+

一共有27813678766个索引读取请求，有6798830个请求在内存中没有找到，直接从硬盘读取索引。

key\_cache\_miss\_rate = key\_reads /key\_read\_requests \* 100%

比如上面的数据，key\_cache\_miss\_rate为0.0244%，4000%个索引读取请求才有一个直接读硬盘，效果已经很好了，key\_cache\_miss\_rate在0.1%以下都很好，如果key\_cache\_miss\_rate在0.01%以下的话，则说明key\_buffer\_size分配得过多，可以适当减少。

## 4.临时表

当执行语句时，关于已经被创建了隐含临时表的数量，我们可以用如下命令查询其具体情况：

mysql> show global status like 'created\_tmp%';

+-------------------------+----------+

| Variable\_name |Value |

+-------------------------+----------+

| Created\_tmp\_disk\_tables | 21119 |

| Created\_tmp\_files |6 |

| Created\_tmp\_tables |17715532 |

+-------------------------+----------+

每次创建临时表时，Created\_tmp\_table都会增加，如果磁盘上创建临时表，Created\_tmp\_disk\_tables也会增加。Created\_tmp\_files表示MySQL服务创建的临时文件数，比较理想的配置是：

Created\_tmp\_disk\_tables/ Created\_tmp\_files \* 100% <= 25%

比如上面的服务器Created\_tmp\_disk\_tables/ Created\_tmp\_files \* 100% =1.20%，就相当不错。我们在看一下MySQL服务器对临时表的配置：

mysql> show variables where Variable\_name in('tmp\_table\_size','max\_heap\_table\_size');

+---------------------+---------+

| Variable\_name |Value |

+---------------------+---------+

| max\_heap\_table\_size | 2097152 |

| tmp\_table\_size |2097152 |

+---------------------+---------+

## 5.打开表的情况

Open\_tables表示打开表的数量，Opened\_tables表示打开过的表数量，我们可以用如下命令查看其具体情况：

mysql> show global status like 'open%tables%';

+---------------+-------+

| Variable\_name | Value |

+---------------+-------+

| Open\_tables | 351 |

| Opened\_tables | 1455 |

如果Opened\_tables数量过大，说明配置中table\_open\_cache的值可能太小。我们查询下服务器table\_open\_cache;

mysql> show variables like 'table\_open\_cache';

+------------------+-------+

| Variable\_name | Value |

+------------------+-------+

| table\_open\_cache | 2048 |

+------------------+-------+

比较合适的值为：

open\_tables/ opened\_tables\* 100% > = 85%

open\_tables/ table\_open\_cache\* 100% < = 95%

## 6.进程使用情况

如果我们在MySQL服务器的配置文件中设置了thread\_cache\_size，当客户端断开时，服务器处理此客户请求的线程将会缓存起来以响应一下客户而不是销毁(前提是缓存数未达上线)Thread\_created表示创建过的线程数，我们可以用如下命令查看：

mysql> show global status like 'thread%';

+-------------------+-------+

| Variable\_name | Value |

+-------------------+-------+

| Threads\_cached | 40 |

| Threads\_connected | 1 |

| Threads\_created | 330 |

| Threads\_running | 1 |

+-------------------+-------+

如果发现Threads\_created的值过大的话，表明MySQL服务器一直在创建线程，这也是比较耗费资源的，可以适当增大配置文件中thread\_cache\_size的值。查询服务器thread\_cache\_size配置如下：

mysql> show variables like 'thread\_cache\_size';

+-------------------+-------+

| Variable\_name | Value |

+-------------------+-------+

| thread\_cache\_size | 100 |

+-------------------+-------+

示例中的MySQL服务器还是挺健康的。

## 7.查询缓存(querycache)

它主要涉及两个参数，query\_cache\_size是设置MySQL的Query Cache大小，query\_cache\_type是设置使用查询缓存的类型，我们可以用如下命令查看其具体情况：

mysql> show global status like 'qcache%';

+-------------------------+-----------+

| Variable\_name |Value |

+-------------------------+-----------+

| Qcache\_free\_blocks |22756 |

| Qcache\_free\_memory |76764704 |

| Qcache\_hits | 213028692 |

| Qcache\_inserts |208894227 |

| Qcache\_lowmem\_prunes |4010916 |

| Qcache\_not\_cached |13385031 |

| Qcache\_queries\_in\_cache | 43560 |

| Qcache\_total\_blocks |111212 |

+-------------------------+-----------+

MySQL查询缓存变量的相关解释如下：

Qcache\_free\_blocks： 缓存中相领内存快的个数。数目大说明可能有碎片。flush query cache会对缓存中的碎片进行整理，从而得到一个空间块。

Qcache\_free\_memory：缓存中的空闲空间。

Qcache\_hits：多少次命中。通过这个参数可以查看到Query Cache的基本效果。

Qcache\_inserts：插入次数，没插入一次查询时就增加1。命中次数除以插入次数就是命中比率。

Qcache\_lowmem\_prunes：多少条Query因为内存不足而被清楚出Query Cache。通过Qcache\_lowmem\_prunes和Query\_free\_memory相互结合，能够更清楚地了解到系统中Query Cache的内存大小是否真的足够，是否非常频繁地出现因为内存不足而有Query被换出的情况。

Qcache\_not\_cached：不适合进行缓存的查询数量，通常是由于这些查询不是select语句或用了now()之类的函数。

Qcache\_queries\_in\_cache：当前缓存的查询和响应数量。

Qcache\_total\_blocks：缓存中块的数量。

我们在查询一下服务器上关于query\_cache的配置命令：

mysql> show variables like 'query\_cache%';

+------------------------------+---------+

| Variable\_name | Value |

+------------------------------+---------+

| query\_cache\_limit | 1048576 |

| query\_cache\_min\_res\_unit | 2048 |

| query\_cache\_size | 2097152 |

| query\_cache\_type | ON |

| query\_cache\_wlock\_invalidate | OFF |

+------------------------------+---------+

字段解释如下：

query\_cache\_limit：超过此大小的查询将不缓存。

query\_cache\_min\_res\_unit：缓存块的最小值。

query\_cache\_size：查询缓存大小。

query\_cache\_type：缓存类型，决定缓存什么样的查询，示例中表示不缓存select sql\_no\_cache查询。

query\_cache\_wlock\_invalidat：表示当有其他客户端正在对MyISAM表进行写操作，读请求是要等WRITELOCK释放资源后再查询还是允许直接从Query Cache中读取结果，默认为OFF（可以直接从Query Cache中取得结果。）

query\_cache\_min\_res\_unit的配置是一柄双刃剑，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%

如果查询碎片率超过20%，可以用 flushquery cache 整理缓存碎片，或者试试减少query\_cache\_min\_res\_unit，如果你查询都是小数据库的话。

查询缓存利用率 =(Qcache\_free\_size – Qcache\_free\_memory)/query\_cache\_size \* 100%

查询缓存利用率在25%一下的话说明query\_cache\_size设置得过大，可适当减少;查询缓存利用率在80%以上而且Qcache\_lowmem\_prunes> 50的话则说明query\_cache\_size可能有点小，不然就是碎片太多。

查询命中率 = (Qcache\_hits- Qcache\_insert)/Qcache)hits \* 100%

示例服务器中的查询缓存碎片率等于20%左右，查询缓存利用率在50%，查询命中率在2%，说明命中率很差，可能写操作比较频繁，而且可能有些碎片。

## 8.排序使用情况

它表示系统中对数据进行排序时所用的Buffer，我们可以用如下命令查看：

mysql> show global status like 'sort%';

+-------------------+----------+

| Variable\_name | Value |

+-------------------+----------+

| Sort\_merge\_passes | 10 |

| Sort\_range | 37431240 |

| Sort\_rows | 6738691532|

| Sort\_scan | 1823485 |

+-------------------+----------+

Sort\_merge\_passes包括如下步骤：MySQL首先会尝试在内存中做排序，使用的内存大小由系统变量sort\_buffer\_size来决定，如果它不够大则把所有的记录都读在内存中，而MySQL则会把每次在内存中排序的结果存到临时文件中，等MySQL找到所有记录之后，再把临时文件中的记录做一次排序。这次再排序就会增加sort\_merge\_passes。实际上，MySQL会用另外一个临时文件来存储再次排序的结果，所以我们通常会看到sort\_merge\_passes增加的数值是建临时文件数的两倍。因为用到了临时文件，所以速度可能会比较慢，增大sort\_buffer\_size会减少sort\_merge\_passes和创建临时文件的次数，但盲目地增大sort\_buffer\_size并不一定能提高速度。

## 9.文件打开数(open\_files)

我们现在处理MySQL故障时，发现当Open\_files大于open\_files\_limit值时，MySQL数据库就会发生卡住的现象，导致Nginx服务器打不开相应页面。这个问题大家在工作中应注意，我们可以用如下命令查看其具体情况：

show global status like 'open\_files';

+---------------+-------+

| Variable\_name | Value |

+---------------+-------+

| Open\_files | 1481 |

+---------------+-------+

mysql> show global status like 'open\_files\_limit';

+------------------+-------+

| Variable\_name | Value |

+------------------+--------+

| Open\_files\_limit | 4509 |

+------------------+--------+

比较合适的设置是：Open\_files/ Open\_files\_limit \* 100% < = 75%

10.InnoDB\_buffer\_pool\_cache合理设置

InnoDB存储引擎的缓存机制和MyISAM的最大区别就在于，InnoDB不仅仅缓存索引，同时还会缓存实际的数据。此参数用来设置InnoDB最主要的Buffer的大小，也就是缓存用户表及索引数据的最主要缓存空间，对InnoDB整体性能影响也最大。

无论是MySQL官方手册还是网络上许多人分享的InnoDB优化建议，都是简单地建议将此值设置为整个系统物理内存的50%~80%。这种做法其实不妥，我们应根据实际的运行场景来正确设置此项参数。

MySQL优化小思想

很多时候我们会发现，通过参数设置进行性能优化所带来的性能提升，并不如许多人想象的那样会产生质的飞跃，除非是之前的设置存在严重不合理的情况。我们不能将性能调优完全依托与通过DBA在数据库上线后进行参数调整，而应该在系统设计和开发阶段就尽可能减少性能问题。(重点在于前期架构合理的设计及开发的程序合理)

# MySQL数据库的可扩展架构方案

如果凭借MySQL的优化任无法顶住压力，这个时候我们就必须考虑MySQL的可扩展性架构了(有人称为MySQL集群)它有以下明显的优势：

成本低，很容易通过价格低廉Pc server搭建出一个处理能力非常强大的计算机集群。

不太容易遇到瓶颈，因为很容易通过添加主机来增加处理能力。

单节点故障对系统的整体影响较小。

目前可行的方案如下：

（1）MySQL Cluter

其特点为可用性非常高，性能非常好。每份数据至少可在不同主机上存一份副本，且冗余数据拷贝实时同步。但它的维护非常复杂，存在部分Bug，目前还不适合比较核心的线上系统，所以暂时不推荐。

（2）DRBD磁盘网络镜像方案

其特点为软件功能强大，数据可在底层快设备级别跨物理主机镜像，且可根据性能可靠性要求配置不同级别的同步。I/O操作会保持顺序，可满足数据库对数据一致性的苛刻要求。但非分布式文件系统环境无法支持镜像数据同时可见，性能和可靠性两者互相矛盾，无法适用于性能和可靠性要求都比较苛刻的环境，维护成本高于MySQL Replication。另外，DRBD是官方推荐的可用于MySQL的搞可用方案之一，大家可根据实际环境来考虑是否部署。

（3）MySQL Replication

在工作中，此种MySQL搞可用、高扩展性架构也是用得最多的，我也推荐此方案，一主多从、双主多从是生产环境常见的高可用架构方案。