一.服务器优化
1.磁盘存储采用SSD告诉磁盘
2.内存占有率是存储库占用大小的三倍
3.处理器不低于4核
推荐：8核32G / 300G+
内核优化：

二.服务优化
my.cnf配置：
innodb_buffer_pool_size:这是你安装完InnoDB后第一个应该设置的选项。缓冲池是数据和索引缓存的地方：这个值越大越好，这能保证你在大多数的读取操作时使用的是内存而不是硬盘。典型的值是5-6GB(8GB内存)，20-25GB(32GB内存)，100-120GB(128GB内存)。
innodb_log_file_size：这是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 5.1，它都难于调整，因为一方面你想让它更大来提高性能，另一方面你想让它更小来使得崩溃后更快恢复。幸运的是从MySQL 5.5之后，崩溃恢复的性能的到了很大提升，这样你就可以同时拥有较高的写入性能和崩溃恢复性能了。一直到MySQL 5.5，redo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。
一开始就把innodb_log_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。如果你知道你的应用程序需要频繁的写入数据并且你使用的时MySQL 5.6，你可以一开始就把它这是成4G。
max_connections:如果你经常看到'Too many connections'错误，是因为max_connections的值太低了。这非常常见因为应用程序没有正确的关闭数据库连接，你需要比默认的151连接数更大的值。
max_connection值被设高了(例如1000或更高)之后一个主要缺陷是当服务器运行1000个或更高的活动事务时会变的没有响应。在应用程序里使用连接池或者在MySQL里使用进程池有助于解决这一问题。

InnoDB配置
从MySQL 5.5版本开始，InnoDB就是默认的存储引擎并且它比任何其他存储引擎的使用都要多得多。那也是为什么它需要小心配置的原因。
innodb_file_per_table：这项设置告知InnoDB是否需要将所有表的数据和索引存放在共享表空间里（innodb_file_per_table = OFF） 或者为每张表的数据单独放在一个.ibd文件（innodb_file_per_table = ON）。每张表一个文件允许你在drop、truncate或者rebuild表时回收磁盘空间。这对于一些高级特性也是有必要的，比如数据压缩。但是它不会带来任何性能收益。你不想让每张表一个文件的主要场景是：有非常多的表（比如10k+）。
MySQL 5.6中，这个属性默认值是ON，因此大部分情况下你什么都不需要做。对于之前的版本你必需在加载数据之前将这个属性设置为ON，因为它只对新创建的表有影响。
innodb_flush_log_at_trx_commit：默认值为1，表示InnoDB完全支持ACID特性。当你的主要关注点是数据安全的时候这个值是最合适的，比如在一个主节点上。但是对于磁盘（读写）速度较慢的系统，它会带来很巨大的开销，因为每次将改变flush到redo日志都需要额外的fsyncs。将它的值设置为2会导致不太可靠（reliable）因为提交的事务仅仅每秒才flush一次到redo日志，但对于一些场景是可以接受的，比如对于主节点的备份节点这个值是可以接受的。如果值为0速度就更快了，但在系统崩溃时可能丢失一些数据：只适用于备份节点。
innodb_flush_method: 这项配置决定了数据和日志写入硬盘的方式。一般来说，如果你有硬件RAID控制器，并且其独立缓存采用write-back机制，并有着电池断电保护，那么应该设置配置为

O_DIRECT；否则，大多数情况下应将其设为fdatasync（默认值）。sysbench是一个可以帮助你决定这个选项的好工具。

innodb_log_buffer_size: 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）一般来说已经够用了，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外的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。

其他设置

query_cache_size: query cache（查询缓存）是一个众所周知的瓶颈，甚至在并发并不多的时候也是如此。 最佳选项是将其从一开始就停用，设置query_cache_size = 0（现在MySQL 5.6的默认值）并利用其他方法加速查询：优化索引、增加拷贝分散负载或者启用额外的缓存（比如memcache或redis）。如果你已经为你的应用启用了query cache并且还没有发现任何问题，query cache可能对你有用。这是如果你想停用它，那就得小心了。

log_bin：如果你想让数据库服务器充当主节点的备份节点，那么开启二进制日志是必须的。如果这么做了之后，还别忘了设置server_id为一个唯一的值。就算只有一个服务器，如果你想做基于时间点的数据恢复，这（开启二进制日志）也是很有用的：从你最近的备份中恢复（全量备份），并应用二进制日志中的修改（增量备份）。二进制日志一旦创建就将永久保存。所以如果你不想让磁盘空间耗尽，你可以用 PURGE BINARY LOGS 来清除旧文件，或者设置 expire_logs_days 来指定过多少天日志将被自动清除。

记录二进制日志不是没有开销的，所以如果你在一个非主节点的复制节点上不需要它的话，那么建议关闭这个选项。

skip_name_resolve：当客户端连接数据库服务器时，服务器会进行主机名解析，并且当DNS很慢时，建立连接也会很慢。因此建议在启动服务器时关闭skip_name_resolve选项而不进行DNS查找。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个已有系统中必须格外小心。

参考：

```
[client]
port = 3306
socket = /var/lib/mysql/mysql.sock

[mysql]
#这个配置段设置启动MySQL服务的条件；在这种情况下，no-auto-rehash确保这个服务启动得比较快。
no-auto-rehash

[mysqld]
user = mysql
port = 3306
socket = /var/lib/mysql/mysql.sock
basedir = /usr/local/mysql
datadir = /data/mysql/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80。

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。默认151

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。默认100

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4MB，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。 系统默认2M，使用默认值即可

#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。系统默认2M，使用默认值即可

thread_cache_size = 300
#默认38
# 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
#系统默认为10，使用10先观察
# 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

query_cache_size = 64M
#在MyISAM引擎优化中，这个参数也是一个重要的优化参数。但也爆露出来一些问题。机器的内存越来越大，习惯性把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。

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 #开启失败

#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256KB，使用默认观察

transaction_isolation = READ-COMMITTED
# 设定默认的事务隔离级别.可用的级别如下:READ UNCOMMITTED-读未提交 READ COMMITTE-读已提交 REPEATABLE READ -可重复读 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

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
#默认128K
# 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 默认8M

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 5.6无此值设置
#myisam_repair_threads = 1   默认为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 设置过大导致报错，默认12M观察
#表空间文件 重要数据

#innodb_file_io_threads = 4   不明确，使用默认值
#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   使用默认8M
#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  使用默认48M
#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   使用默认2
#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  使用默认75观察
#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
#默认为50秒
# 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
#默认为No
#独享表空间（关闭）

[mysqldump]
quick
# max_allowed_packet = 32M

[mysqld_safe]
log-error=/data/mysql/mysql_oldboy.err
pid-file=/data/mysql/mysqld.pid

sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES
```

三.SQL优化

1.索引

注意不需要加索引的场景（索引失效）：

· 全表扫描

· or前后不一致，无论前者或者后者不是索引都无效

· heap表

· 复合索引，前缀索引

· like 以%起始

where子句中的常量非字符串的将索引失效。

2.where子句

禁忌null判断如：is null ,is not null

应尽量避免在 where 子句中使用!=或<>操作符

where子句中使用索引的操作符：<，<=，=，>，>=，BETWEEN，IN

应尽量避免在where子句中使用or作为连接符，可以将or改造成合并查询

当存储引擎采用myisam时or支持索引

In, not in慎用，容易导致全表扫描

where子句中使用参数将导致全表扫描

如：select id from t where num=@num

可以改为强制查询使用索引： select id from t with(index(索引名)) where num=@num

应尽量避免在 where 子句中对字段进行表达式操作

这将导致引擎放弃使用索引而进行全表扫描。如：

select id from t where num/2=100

应改为: select id from t where num=100*2

应尽量避免在where子句中对字段进行函数操作

这将导致引擎放弃使用索引而进行全表扫描。如：

select id from t where substring(name,1,3)='abc' --name

select id from t where datediff(day,createdate,'2005-11-30')=0--‘2005-11-30’

生成的id 应改为:

select id from t where name like 'abc%'

select id from t where createdate>='2005-11-30' and createdate<‘2005-12-1'

不要在 where 子句中的“=”左边进行函数、算术运算或其他表达式运算，

否则系统将可能无法正确使用索引。

索引字段不是复合索引的前缀索引

例如 在使用索引字段作为条件时，如果该索引是复合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用，并且应尽可能的让字段顺序与索引顺序相一致

并不是所有的索引对查询都有效:

一个表中有ename，该字段默认可以为空，并且该表的记录该字段50%都没值，此时加索引反而不能加快查询速度。

禁忌使用*查询

尽量不使用临时表

如果非要使用临时表，务必要完成删除临时表操作，释放资源。

尽量避免使用游标

count（*）优于count(primary_key)

Order by

对order by的字段做索引

当索引被where子句占用时要将order by的字段与where子句的字段建立复合索引，不然索引排序失效。

当order by多个排序的时候，需所有字段都升序或者都降序排序，否则索引失效。

四.数据库设计优化

1.设计表时的类型选择:

· 能用短类型就不要用长类型

· 能用数字类型就不要用字符串类型

· 能用varchar的都不要用char

· 能用字符串类型就不要用文本对象类型

· 能用时间戳的就别用日期类型或者字符串日期

2.设计表时的属性选择:

· 尽量给所有设计的的字段可为空的都设置默认值，如果不指定默认值，系统会默认数字类型为0，字符串为null

· 尽量保证一个表存在一个主键自增字段

· 最好选择数字类型作为自增计数器。