MySQL 配置文件详解

```
./configure --prefix=/usr/local/mysql \
--without-debug \              #去除 debug 模式.
--enable-thread-safe-client \      #以线程方式编译客户端.
--with-pthread    \             #强制使用 pthread 线程库编译.
--enable-assembler \          #允许使用汇编模式.
--enable-profiling \             #Build a version with query profiling code (req.community-features)
--with-mysqld-ldflags=-all-static \   #静态编译 mysqld 的额外 link 参数.
--with-client-ldflags=-all-static \   #静态编译 client 的额外 link 参数.
--with-charset=utf8 \          #默认字符 utf8.
--with-extra-charsets=all \      #支持所有的语言字符.
--with-innodb \              #innodb 数据引擎.
--with-plugins=innobase \
--with-plugins=heap \          #内存数据引擎.
--with-mysqld-user=mysql \      #mysql 安装使用的帐号
--without-embedded-server \     #去除安装 embedded-server.
--with-server-suffix=-community \  #社区形式安装.
--with-unix-socket-path=/tmp/mysql.sock
```

MYSQl 启动的一般设置:
/etc/my.cnf 基本部设参数设置.

```
# 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 连接管理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"  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.
#  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
#  在 MYSQL 的连接请求等待队列中允许存放的最大连接请求数.  defautl=50,最大 65535,根据 os 对网络监听队列的情况来设置.
back_log = 20000

# MySQL  服务所允许的同时会话数的上限
#  其中一个连接将被 SUPER 权限保留作为管理员登录.
#  即便已经达到了连接数的上限.
#  整个 Mysql 允许的最大连接数.这个参数会影响 mysql 的应用并发处理能力.有些资料上提到  500-800，以我们的机子来说单个 mysql 实例设置 10000 应该是没有问题的。如果还需可能需要第三方软件解决 php 连接池的问题，提高还需要连接池.
max_connections = 10000
```

```
# 每个客户端连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"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_errors = 10

# 所有线程所打开表的数量.
# 增加此值就增加了 mysqld 所需要的文件描述符的数量
# 这样你需要确认在[mysqld_safe]中 "open-files-limit" 变量设置打开文件数量允许至少
4096
# 根据以下命今进行实际需要设置.
#  mysq> show variables like 'table_cache';
#  mysq> show status lie 'open_tables';
table_cache = 2048

# 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件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他一些软件依赖来锁定 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
字段一起工作时相当必要)
# 每个连接独立的大小.大小动态增加
# 这个是根据 net_buffer 相对应，是 net buffer 的最大值。 default 是 16M
max_allowed_packet = 16M

# 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,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 缓冲中然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 = 1M

# 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尽所有的内存资源.
max_heap_table_size = 64M

# 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
# 如果排序后的数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 "Sort_merge_passes" 状态变量.
# 在排序发生时由每个线程分配
sort_buffer_size = 8M
```

# 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下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_buffer_size = 8M

# 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 中的线程还少于 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的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提升.)
# thread cache 池中应该存放的连接线程数.长连接的应用中,设为 50-100 之间.
thread_cache_size = 80

# 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.
# 此值只对于支持 thread_concurrency() 函数的系统有意义(例如 Sun Solaris).
# 你可可以尝试使用 [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.
# 查看 "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足够高.
# 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,
# 查询缓冲也许引起性能下降而不是性能提升.
query_cache_size = 64M

# 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集将其他所有的查询结果都覆盖.
query_cache_limit = 2M

# 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 = 4
Innodb 相关优化及说明:
#设置存储引擎默认引擎为 InnoDB.
default-storage_engine = 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内存,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, 你一般不需要修改此值.
# SHOW INNODB STATUS 命令会显示当先使用的数量.
# 根据表的多少来确定大小，一般 16M 已能适用于几百个表了.
innodb_additional_mem_pool_size = 16M

# 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,你在存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,你可以设置这个变量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换页颠簸.
# 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,
# 所以不要设置的太高.
# 这个参数影响会较大，在没有其它服务在此计算机上跑，80%是完全可以的。一般 linux 系统给它 800M 问题系统使用已足够
innodb_buffer_pool_size = 14G

# 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保存你的数据,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选择.
# 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 = <directory>

# 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 可能在一个大数值下表现的更好.
# innodb_file_io_threads 只是在 5.4 版本之前使用这个参数，一般设置为 cpu 多少核，就设多少。能达到比较好的效果.
innodb_file_io_threads = 12

# mysql 5.4 版本设置为（在测试过程中以下方式能达到更好的效果）:
innodb_read_io_threads = 12
innodb_write_io_threads = 6

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 开始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式.
过高的值可能导致线程的互斥颠簸.默认是 16，在这里我们不做限制最好。所以设定为 0

```
innodb_thread_concurrency = 0


# 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 ACID 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 2
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盘.
# 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.
# 0 与 1 和 2 影响的性能将是 5 倍以上，强列建议是 0,最多会丢失 1 秒的数据.
innodb_flush_log_at_trx_commit = 0

# 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.
# 这可能极大增加关机时间, 但是取而代之的是 InnoDB 可能在下次启动时做这些操作.
#innodb_fast_shutdown

# 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.
# 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 = 8M

# 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, 请注意一个大的日志文件大小会增加恢复进程所需要的时间.
innodb_log_file_size = 256M

# 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硬盘上或者一个 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 会开始刷新他们防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".
# O_DSYNC 是要配置 innodb_max_dirty_pages_pct 来使用.
```

#innodb_flush_method=O_DSYNC

# 在被回滚前,一个 InnoDB 的事务应该等待一个锁被批准多久.
# InnoDB 在其拥有的锁表中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 InnoDB 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 = 120

Linux 服务定制简化.
vim /etc/inittab
id:3:initdefault: #3 为命令行，不需要启动桌面.

将 3,4,5,6 注释掉，用不了那么多 TTY,节约资源.
Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
#3:2345:respawn:/sbin/mingetty tty3
#4:2345:respawn:/sbin/mingetty tty4
#5:2345:respawn:/sbin/mingetty tty5
#6:2345:respawn:/sbin/mingetty tty6

DB 服务器下需要的服务(简化后):
service init on/off
acpid 2,3,5 on
cpuspeed 2,3,5 on
crond 2,3,5 on
messagebus 2,3,5 on
mysqld 2,3,5 on
network 2,3,5 on
ntpd 2,3,5 on
snmpd 2,3,5 on
sshd 2,3,5 on
syslog 2,3,5 on
xfs 2,3,5 on
xinetd 2,3,5 on