l  **lower\_case\_table\_names**

   Linux或类Unix平台，对文件名称大小写敏感，也即对数据库、表、存储过程等对象名称大小写敏 感，为减少开发人员的开发成本，为此推荐大家设置该参数使对象名称都自动转换成小写；

l  **max\_connect\_errors**

    max\_connect\_errors默认值为10，也即mysqld线程没重新启动过，一台物理服务器只要连接 异常中断累计超过10次，就再也无法连接上mysqld服务，为此建议大家设置此值至少大于等于10W； 若异常中断累计超过参数设置的值，有二种解决办法，执行命令：FLUSH HOSTS;或者重新启动mysqld服务；

l  **interactive\_timeout  and wait\_timeout**

u  interactive\_timeout

       处于交互状态连接的活动被服务器端强制关闭，而等待的时间，单位：秒；

u  wait\_timeout

       与服务器端无交互状态的连接，直到被服务器端强制关闭而等待的时间，此参数只对基于TCP/IP或基于 Socket通信协议建立的连接才有效，单位：秒；

u  推荐设置

     interactive\_timeout = 172800

     wait\_timeout  = 172800

l  **transaction-isolation  and binlog-format**

u  **transaction-isolation**

      可供设置的值：READ-UNCOMMITTED、READ-COMMITTED、REPEATABLE-READ、

SERIALIZABLE，默认的值为： REPEATABLE-READ，事务隔离级别设置的不同，对二进制日志登记格

式影响非常大，详细信息可见文章[解读MySQL事务的隔离级别和日志登记模式选择技巧](http://www.mysqlops.com/2011/10/2011/04/25/mysql-isolation-binlog.html)；

u  **binlog-format**

       复制的模式，可供设置的值：STATEMENT、ROW、MIXED（注：5.0.\*只有命令行式复制），

5.1.\*版本默认设置：MIXED；

u  **推荐配置**

①   只读为主的业务应用场景

             transaction-isolation =  read-committed

             binlog-format  = mixed            **#5.1.\*版本，5.0.\*只能设置为 statement**

①   非只读为主的业务应用场景

             transaction-isolation = repeatabled-read

             binlog-format  = mixed            **#5.1.\*版本，5.0.\*只能设置为 statement**

l  event\_scheduler

事务调度默认是关闭状态，也推荐源码编译的版本可不编译进来，以及实际生产环境保持默认禁用 状态，当真正需要用的时候，可以临时打开，命令：SET GLOBAL event\_scheduler=1;

l  skip\_external\_locking

外部锁，也即操作系统所实施的锁，只对MyISAM引擎有效，且容易造成死锁发生，为此我们一律禁用；

l  innodb\_adaptive\_hash\_index

InnoDB引擎会根据数据的访问频繁度，把表的数据逐渐缓到内存，若是一张表的数据大量缓存在 内存中，则使用散列索引（注：Hash Index）会更高效。InnoDB内有Hash Index机制，监控数据的访 问情况，可以自动创建和维护一个Hash Index，以提供访问效率，减少内存的使用；

l  innodb\_max\_dirty\_pages\_pct

    InnoDB主线程直接更新Innodb\_buffer\_pool\_size中存在的数据，并且不实时刷回磁盘，而是等待 相关的处罚事件发生，则允许缓存空间的数据量不实时刷回磁盘的最大百分比。比例设置较小，有利于 减少mysqld服务出现问题的时候恢复时间，缺点则是需要更多的物理I/O，为此我们必须根据业务特点 和可承受范围进行一个折中，一般范围建议设置为5%~90%，像我们SNS游戏行业的写非常厉害，综合 各方面因素，设置为20%；

===============================================================================

===============================================================================

===============================================================================

===============================================================================  
l  innodb\_commit\_concurrency

含义：同一时刻，允许多少个线程同时提交InnoDB事务，默认值为0，范围0-1000。

0      — 允许任意数量的事务在同一时间点提交；

N>0  — 允许N个事务在同一时间点提交；

**注意事项：**

①   mysqld提供服务时，不许把 innodb\_commit\_concurrency 的值从0改为非0，或非0的值改为0；

②   mysqld提供时，允许把 innodb\_commit\_concurrency 的值N>0改为M，且M>0；

l  innodb\_concurrency\_tickets

含义：

同一时刻，能访问InnoDB引擎数据的线程数，默认值为500，范围1-4294967295。

补充说明：当访问InnoDB引擎数据的线程数达到设置的上线，线程将会被放到队列中，等待其他线程释放ticket。

**建议：**

    MySQL数据库服务最大线程连接数参数max\_connections，一般情况下都会设置在128-1024的范围，再结合实际业务可能的最大事务并发度，innodb\_concurrency\_tickets保持默认值一般情况下足够。

l  innodb\_fast\_shutdown  and innodb\_force\_recovery

innodb\_fast\_shutdown：

含义：设置innodb引擎关闭的方式，默认值为：1，正常关闭的状态；

0         —  mysqld服务关闭前，先进行数据完全的清理和插入缓冲区的合并操作，若是脏数据

较多或者服务器性能等因素，会导致此过程需要数分钟或者更长时间；

1          — 正常关闭mysqld服务，针对innodb引擎不做任何其他的操作；

2  — 若是mysqld出现崩溃，立即刷事务日志到磁盘上并且冷关闭mysqld服务；没有提交

的事务将会丢失，但是再启动mysqld服务的时候会进行事务回滚恢复；

innodb\_force\_recovery：

含义：

mysqld服务出现崩溃之后，InnoDB引擎进行回滚的模式，默认值为0，可设置的值0~6；

**提示：**

   只有在需要从错误状态的数据库进行数据备份时，才建议设置innodb\_force\_recovery的值大于0。 若是把此参数作为安全选项，也可以把参数的值设置大于0，防止InnoDB引擎的数据变更，设置不同值的作用：

0 — 正常的关闭和启动，不会做任何强迫恢复操作；

1 — 跳过错误页，让mysqld服务继续运行。跳过错误索引记录和存储页，尝试用

SELECT \*  INOT OUTFILE ‘../filename’ FROM tablename;方式，完成数据备份；

2 — 阻止InnoDB的主线程运行。清理操作时出现mysqld服务崩溃，则会阻止数据恢复操作；

3 —  恢复的时候，不进行事务回滚；

4 — 阻止INSERT缓冲区的合并操作。不做合并操作，为防止出现mysqld服务崩溃。不计算

表的统计信息

5 — mysqld服务启动的时候不检查回滚日志：InnoDB引擎对待每个不确定的事务就像提交

的事务一样；

6 — 不做事务日志前滚恢复操作；

**推荐的参数组合配置：**

innodb\_fast\_shutdown = 1

#若是机房条件较好可设置为0（双路电源、UPS、RAID卡电池和供电系统稳定性）

innodb\_force\_recovery =0

#至于出问题的时候，设置为何值，要视出错的原因和程度，对数据后续做的操作

l  innodb\_additional\_mem\_pool\_size

含义：开辟一片内存用于缓存InnoDB引擎的数据字典信息和内部数据结构（比如：自适应HASH索引结构）；

默认值：build-in版本默认值为：1M；Plugin-innodb版本默认值为：8M；

提示：若是mysqld服务上的表对象数量较多，InnoDB引擎数据量很大，且innodb\_buffer\_pool\_size的值设置 较大，则应该适当地调整innodb\_additional\_mem\_pool\_size的值。若是出现缓存区的内存不足，则会直接向操作系统申请内存分配，并且会向MySQL的error log文件写入警告信息；

l  innodb\_buffer\_pool\_size

含义：开辟一片内存用于缓存InnoDB引擎表的数据和索引；

默认值：历史默认值为：8M，现在版本默认值为：128M；

参数最大值：受限于CPU的架构，支持32位还是支持64位，另外还受限于操作系统为32位还是64位；

提示：

innodb\_buffer\_pool\_size的值设置合适，会节约访问表对象中数据的物理IO。官方手册上建议专用的数据库服务器，可考虑设置为物理内存总量的80%，但是个人建议要看物理服务器的物理内存总量，以及考虑： 是否只使用InnoDB引擎、mysqld内部管理占用的内存、最大线程连接数和临时表等因素，官方提供的80%值作为一个参考，举而个例子方便大家作决定（前提：物理服务器为mysqld服务专用，且只用InnoDB引擎,假设数据量远大于物理内存）：

1).内存配置：24G    则 innodb\_buffer\_pool\_size=18G

1).内存配置：32G    则 innodb\_buffer\_pool\_size=24G

出现下列哪些情况，则可以考虑减小innodb\_buffer\_pool\_size的值：

1).出现物理内存的竞争，可能导致操作系统的分页；

2).InnoDB预分配额外的内存给缓冲区和结构管理，当分配的总内存量超过innodb\_buffer\_pool\_size值的10%；

3).地址空间要求必须为连续的，在windows系统有一个严重问题，DLL需要加载在特定的地址空间；

4).初始化缓冲区的时间消耗，与缓冲区的大小成正比。官方提供的数据 Linux X86 64位系统 初始化 innodb\_buffer\_pool\_size=10G 大概需要6秒钟；

===============================================================================

===============================================================================

===============================================================================

===============================================================================

l  innodb\_flush\_log\_at\_trx\_commit  AND  sync\_binlog

**innodb\_flush\_log\_at\_trx\_commit = N：**

N=0  – 每隔一秒，把事务日志缓存区的数据写到日志文件中，以及把日志文件的数据刷新到磁盘上；

N=1  – 每个事务提交时候，把事务日志从缓存区写到日志文件中，并且刷新日志文件的数据到磁盘上；

N=2  – 每事务提交的时候，把事务日志数据从缓存区写到日志文件中；每隔一秒，刷新一次日志文件，但不一定刷新到磁盘上，而是取决于操作系统的调度；

**sync\_binlog =  N：**

N>0  — 每向二进制日志文件写入N条SQL或N个事务后，则把二进制日志文件的数据刷新到磁盘上；

N=0  — 不主动刷新二进制日志文件的数据到磁盘上，而是由操作系统决定；

**推荐配置组合：**

N=1,1  — 适合数据安全性要求非常高，而且磁盘IO写能力足够支持业务，比如充值消费系统；

N=1,0  — 适合数据安全性要求高，磁盘IO写能力支持业务不富余，允许备库落后或无复制；

N=2,0或2,m(0<m<100)  — 适合数据安全性有要求，允许丢失一点事务日志，复制架构的延迟也能接受；

N=0,0  — 磁盘IO写能力有限，无复制或允许复制延迟稍微长点能接受，例如：日志性登记业务；

l  innodb\_file\_per\_table

启用单表空间，减少共享表空间维护成本，减少空闲磁盘空间释放的压力。另外，大数据量情况下 的性能，也会有性能上的提升，为此建议大家使用**独立表空间**代替  **共享表空间**的方式；

l  key\_buffer\_size

key\_buffer\_size只能缓存MyISAM或类MyISAM引擎的索引数据，而innodb\_buffer\_pool\_size不仅能缓存索引数据，还能缓存元数据，但是对于我们只使用InnoDB引擎的数据库系统而言，此参数值也不能设置过于偏小，因为临时表可能会使用到此键缓存区空间，索引缓存区推荐：64M；

l  query\_cache\_type  and query\_cache\_size

n  query\_cache\_type=N

N=0  —- 禁用查询缓存的功能；

N=1  —- 启用产讯缓存的功能，缓存所有符合要求的查询结果集，除SELECT SQL\_NO\_CACHE..， 以及不符合查询缓存设置的结果集外；

N=2  —- 仅仅缓存SELECT SQL\_CACHE …子句的查询结果集，除不符合查询缓存设置的结果集外；

n  query\_cache\_size

查询缓存设置多大才是合理？至少需要从四个维度考虑：

①   查询缓存区对DDL和DML语句的性能影响；

②   查询缓存区的内部维护成本；

③   查询缓存区的命中率及内存使用率等综合考虑

④   业务类型

**备注：**详细信息可参考文章：[MySQL加速查询速度的独门武器：查询缓存](http://www.mysqlops.com/2011/08/10/mysql-query-cache.html)