# **摘要**

MySQL的最初的核心思想，主要是开源、简便易用。其开发可追溯至1985年，而第一个内部发行版本诞生，已经是1995年。到1998年，MySQL已经可以支持10中操作系统了，其中就包括win平台。

此文档将从安装开始带领着读者一步步深入了解mySQL相关功能，该文由作者多年实战经验的总結而组成，其中包括如下内容：

* 近几十条优化经验
* mySQL集群、主备
* 多种SQL优化分析手段
* mySQL读写分离
* mySQL横向及垂直折分

# **面向读者**

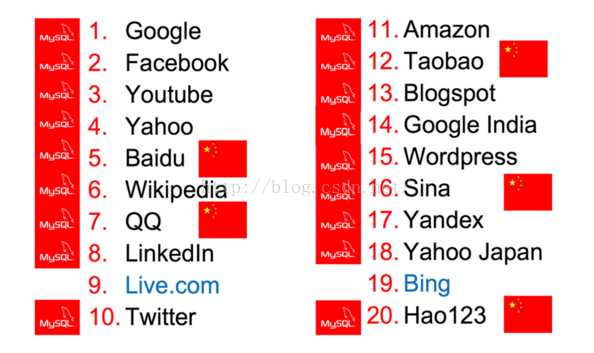
该文适用于Linux CENTOS6.X及以上相关环境，mySQL版本为：5.x及以上。  
本文读者需要具备初级Linux系统使用的相关经验。  
MYSQL的历史  
MySQL的爆发实际是在01、02年，尤其是02年发布的4.0 Beta版，正式选定InnoDB作为默认引擎，对事务处理能力及数据缓存能力有了极大的提高。同年4.1版开始支持子查询，至此MySQL终于蜕变成一个成熟的关系型数据库系统。05年的5.0版本又添加了存储过程、服务端游标、触发器、查询优化以及分布式事务功能，但同年被Oracle抄了后路，InnoDB被Oracle收编。08年，MySQL被Sun收购，09年，Oracle收购了Sun和MySQL。  
MySQL最大的一个特色，就是自由选择存储引擎。每个表都是一个文件，都可以选择合适的存储引擎。常见的引擎有 InnoDB、 MyISAM、 NDBCluster等。但由于这种开放插件式的存储引擎，比如要求数据库与引擎之间的松耦合关系。从而导致文件的一致性大大降低。在SQL执行优化方面，也就有着一些不可避免的瓶颈。在多表关联、子查询优化、统计函数等方面是软肋，而且只支持极简单的HINT。

# **去IOE大潮下的MYSQL与PostgreSQL的比較**

## **PostgreSQL**

PostgreSQL官方宣称的是：“The world’s most advanced open source database”。most advanced我不知道是怎么定义的，因为PosgreSQL还是传统B+树索引的数据库，在一些场景下，比如全插入场景，其还是会比其他一些数据库要来得差很多，比如TokuDB，MongoDB。撇开这部分的因素，不得不承认PostgreSQL是最为强大的开源数据库，或许，但是Oracle依然才是最为强大的关系型数据库。PostgreSQL阵营一直标榜自己在优化器和Oracle可移植性方面的优势，我想这对比MySQL或许是成立的。然而，如果上述都成立的话，为什么PostgreSQL在装机量，流行度等指标上上远远地被后起之秀MySQL给超越了呢？全球前20大网站完全看不到PostgreSQL的身影呢？在写本篇文章的时候，我倏地想到了一个类似的问题，业界公认手机质量最好的Nokia，最终为什么会倒下？

下图是一个mySQL在大型网站和系统中的应用案例统计。



PostgreSQL另一个痛点，我想很多人没有会意识到的，就是在在线事务（OLTP）方面的性能问题。  
PostgreSQL在功能方面或许是比较完整的，但是真的要进入到生产环节，看的不再是简单的功能，因为大部分用户都明白日常所使用的仅是数据库提供的20%功能。MySQL 5.7现在已经可以轻松达到50W QPS的性能，并支持通过NoSQL接口可以达到100W QPS，这是PostgreSQL为什么没有能在互联网时代站住脚跟的一个重要原因之一。在线事务对性能的要求之苛刻，是普通用户所无法感知的。  
PostgreSQL最大的优势是在线分析的场景，因为其优化器对于Join的支持堪称全面，对于复杂查询有着良好的支持，从Oracle迁移到PostgreSQL的成本会比较低。基于PostgreSQL的GreenPlum也已经开源，因此PostgreSQL目前在这方便是较为领先的。

## **MYSQL**

MySQL数据库官方的口号是：“ The world’s most popular open source database.”。对比PostgreSQL，这句话简直无法攻击，并且MySQL官方的目标也一直是成为最为流行的数据库。通过互联网浪潮，移动互联的时代，MySQL是真的做到了。  
MySQL的优势是开源与开放性架构，使其拥有有着各种分支版本与存储引擎可供选择。除了官方的InnoDB存储引擎，还有TokuDB，Infobright引擎可在特定场合下进行使用。也正是因为MySQL的开源与开放，使得大量的开发人员加入到了MySQL的环抱。MySQL是一个非常成功的开源项目，可能很多人忽略了这个重要的因素。  
MySQL被Oracle收购后表现的越来越好，一方面是功能越来越与Oracle数据库接近，很多时候给我的感觉就是开源的Oracle数据库，另一个重要的改进就是bug越来越少，甚至很多遗留了有近10年的bug也已一一修复。官方这样严谨的态度，使得MySQL逐渐站稳了并开始蚕食一部分的企业市场，世界500强的选择就是最好的证明。

下图是一个mySQL在大型网站和系统中的应用案例统计。



MySQL在性能与流行度上的优势我不想再做过多的笔墨，因为这是任何人都无法回避的事实。MySQL数据库之前被PostgreSQL阵营攻击就是优化器，对于多表JOIN的性能以及不支持Hash Join。然而，很多人没有意识到，MySQL已经在5.6版本支持了MRR（Multi-Range Read），ICP（Index Condition Pushdown），BKA（Batched Key Access ）Join这些优化，多表的JOIN性能已经得到了很大幅度的提升。不能否则，MySQL依然不支持Hash Join，但是这些优化的引入已经使得MySQL的Join性能提升到了一个新台阶。同时，在在线分析的领域，用户真的不关心使用Hash Join可以5分钟出报表，而是用MySQL需要8分钟，这些时间完全是可以容忍的。然在在线事务领域，0.1的时间都是所不能容忍的。因此，本人在这里呼吁，尝试升级MySQL到5.6，5.7版本，而不要依然停留在5.1或者5.5版本。  
MySQL替换Oracle另一个被诟病的就是没有Oracle的透明网关(Transparent Gateway)功能，MySQL自带的Fedorate存储引擎支持MySQL数据库间的查询，不支持异构数据库之前的查询。然而，这个问题已经给MariaDB解决，用户只需要通过Connect存储引擎，就能达到类似Oracle透明网关的功能。  
另外，还有用户提出MySQL不支持分区的全局索引，物化视图等，其实这些都可以通过变通的方法实现，如：Amoeba原虫或者是最先进的mycat，这些技术也在网易、淘宝这样的互联网公司使用。  
即使官方的MySQL无法满足你的需求，但是用户依然有InfoBright与TokuDB存储引擎的选择。InfoBright是列存的数据库引擎，非常适用于在线分析领域，这点连PostgreSQL都无法进行匹敌。TokuDB是一种类似LSM数据结构的数据引擎，在大并发的插入生产环境下，其对比各种传统数据库都有着显著的优势，即使对比PostgreSQL与Oracle数据库本身。总之，MySQL能够在各种维度满足用户对于数据库的各种需求。  
PosgreSQL与MySQL对比，最为关键的是整个人才的储备。看看中国的互联网公司基本都已将MySQL数据库作为标配，而PostgreSQL甚至连备胎都无法入选。MySQL在互联网行业积累了大量的高可用架构，分布式架构与灾备经验，但是PostgreSQL几乎为0。再看看图书市场，PostgreSQL凤毛菱角，而MySQL则有很好的书籍供DBA，开发人员，架构师等学习。然即使如此，MySQL离Oracle数据库本身的积累还有很长的路要走。

# **去IOE**

去IOE最早是由淘宝提出，旨在去除IT架构中的IBM小型机，Oracle数据库，EMC存储。去IE是比较简单的事情，因为这仅是硬件的替换。另外，X86技术也越来越成熟，稳定性与小机的差距不断缩小。然而去Oracle数据库才是淘宝去IOE的难点与精华所在。整个去Oracle历时3，4年的时间。其中伴随着功能内部工程师的质疑，大量Oracle人才的流失，但最终已经证明了MySQL数据库替代Oracle的可行性。  
笔者高兴的是传统企业也开始有这样的“觉悟”开始逐步进行去IOE的尝试，不管这种尝试是主动还是被动，但都是值得尊敬的行为。原因在于去Oracle数据库这件事情并不那么简单。数据库是传统企业最为核心的资产，任何损失都是不可接受的。而去年银监会的39号文件也坚定了传统企业的去IOE决心。  
去IOE风潮显现，一大帮的公司开始进入到这个领域，希望借助这阵风来大赚一笔。这点本无可非议，市场与技术相辅相成。然而，有一个非常不好的现象是，很多公司是为了迎合某些领导的需要，而不是真正的为传统企业构建面向互联网+的安全可控的技术架构。而这其中有着一些不为人知的因素。  
首当其冲的是领导们的绩效，传统企业做事，以绩效为导向，这与互联网行业并无不同。但是互联网行业有着技术积累，而且对于技术的选型与转型有着相当的耐心，从淘宝去Oracle用了3，4年就可以看出。而目前摆在传统企业领导面前的现实却是，有ZF文件要求各银行业金融机构对安全可控信息技术的应用以不低于15％的比例逐年增加，直至2019年达到不低于75％的总体占比。  
有一些传统企业的朋友，领导要求他们用PostgreSQL替换Oracle数据库，原因在于这是“最快”的替换Oracle成本，但是他们站在IT从业人员的角度来看这件事是不对的，有种敢怒不敢言。当然，这其中也有部分商业公司在其中推动的关系。但是明白人心里都知道，PostgreSQL国内从业人员寥寥，之前在中国没有大规模的使用经验与架构设计，大多停留在找个文档折腾下的水平上。所谓“最快”的替换方案仅是因为不用进行存储过程的移植，如果只是这样使用PostgreSQL，那么仅是应付上层的文件，而没有真正领会到文件的精神。更有商业公司号称有PostgreSQL的专家，然而非常经不起推敲，玩过GreenPlum的就是PostgreSQL专家？而且GreenPlum也仅做研究性质的用途？与专家交流后发现其对锁与并发，高可用这块的掌握更是让人触目惊心。  
所以笔者一再和身边的朋友说，去IOE不是一件一蹴而就的事情，需要给MySQL时间，否则这件好事情会像着另一个方向而发展，甚至重复当年年Sybase替换Oracle的事件发生。但是好消息是这次的上层领导们终于开始认识到互联网的重要性，理解了安全可控对于一个国家的重要性，而互联网公司的成熟经验具有很好的借鉴意义。

# **总结**

MySQL数据库早已不是原来的迷你数据库，其在功能性与性能方面都已经大幅提升，随着SSD的崛起，MySQL数据库已经完全可以替换Oracle数据，而PostgreSQL还需要很长的路要走。但市场是开放的，就像Oracle称雄的年代，还有DB2，Sybase这样的数据库与之一较长短。我相信互联网时代，依然是百花齐放的年代，没有谁可以一直占领优势，即便是MySQL也没有这个能力。

# **安装**

## **升级yum**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. yum update

由于centos刚安装完后它的yum repo不一定为最新，为保持今后使用yum安装的各类软件始终为最新最稳定版，请升级你的yum。

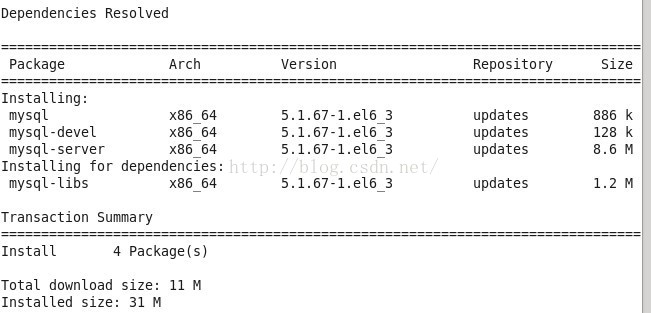
另外，如果不及时升级yum有时还会碰到在yum install时系统提示“找不到相关的安装包”这样的错误。

### **安装mysql**

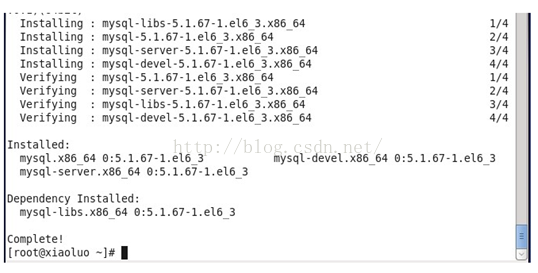
**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. yum install -y mysql-server mysql mysql-devel s.

在等待了一番时间后，yum会帮我们选择好安装mysql数据库所需要的软件以及其它附属的一些软件。



在此，我安装的是mysql5.1，读者可以根据自己的实际环境如：****centos7.x的环境下将安装更高版的mySQL****。  
当出现下面的结果时，就代表mysql数据库安装成功了。



你可以使用如下命令启动和停止mySQL。  
启动：

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. service mysqld start

停止：

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. service mysqld stop

重启：

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. service mysqld restart

# **mySQL的配置与优化**

## **配件文件所在路径**

mySQL安装完后会生成如下几个重要目录：

* 核心配置文件 my.cnf，它位于/etc/目录下
* 日志文件，它们位于/var/log目录
* Binlog、运行时的.sock文件以及数据文件默认位于/var/lib/mysql目录下

### **核心配置文件**

修改my.cnf文件，先来看看我本机的my.cnf的配置

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. [mysqld]
2. init\_connect='SET autocommit=0'
3. server-id=101
4. log-bin=master-bin
5. log-bin-index=master-bin.index
6. lower\_case\_table\_names=1
7. datadir=/var/lib/mysql
8. socket=/var/lib/mysql/mysql.sock
9. user=mysql
10. # Disabling symbolic-links is recommended to prevent assorted security risks
11. symbolic-links=0
12. skip-external-locking
13. skip-name-resolve
14. back\_log=384
15. key\_buffer\_size=512M
16. max\_allowed\_packet=6M
17. thread\_stack=1024k
18. sort\_buffer\_size=8M
19. read\_buffer\_size=8M
20. thread\_cache\_size=64
21. query\_cache\_size=64M
22. tmp\_table\_size=256M
23. max\_connections=500
24. wait\_timeout=600
25. interactive\_timeout=600
26. innodb\_flush\_log\_at\_trx\_commit=2
27. innodb\_log\_buffer\_size=10M
28. innodb\_thread\_concurrency=16

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. [mysqld]
2. #myslqd服务运行时的端口号
3. port=3306
5. #socket文件是在Linux环境下特有的，用户的客户端软件连接可以不通过TCP/IP网络而直接使用unix socket连接到Mysql。
6. socket=/tmp/mysql.sock
8. #避免Mysql的外部锁定，减少出错几率，增强稳定性。
9. skip-external-locking
11. #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NDS解析的时间。但需要注意的是：如果开启该选项，则所有远程主机连
12. 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。
13. skip-name-resolve
15. #back\_log参数的值指出在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对堆栈中，如果系统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。不同的操作系统在这个队列的大小有自己的限制，如果试图将back\_log设定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。
16. back\_log=384
18. #索引缓冲区大小，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，该参数可设置为256M或384M。如果该参数值设置的过大>反而会使服务器的整体效率降低。
19. key\_buffer\_size=384M
21. #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，必须设定为1024的倍数，单位为字节。
22. max\_allowed\_packet=4M
24. #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192K。
25. thread\_stack=256k

28. #设定查询排序时所能使用的缓冲区大小，系统默认大小为2MB，从5.1.23版本开始，在除了WINDOWS 之外的64位平台上可以4GB的限制。该参数
29. 对应的分配内在是每个连接独占的，如果有100个连接，那么实际分配的总排序缓冲区大小为100\*6＝600MB，那么对于内存4GB左右的服务器来>说，推荐将其设置为6MB-8MB。
30. sort\_buffer\_size=6M
32. #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在也是每个连接独享。
33. read\_buffer\_size=4M

36. #设置Thread Cache池中可以缓存的连接池线程最大数量，可设置为0-16384,默认为0。1GB内存我们配置为8，2GB内存我们配置为16，4GB或4GB以上内在我们配置为64。
37. thread\_cache\_size=64
38. #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
39. Qcache\_hits的值非常大，则表明查询缓冲使用的非常频繁
40. query\_cache\_size=64M
42. #设置内在临时表最大值，如果超过该值，则会将临时表写入磁盘，其范围为1KB至4GB。
43. tmp\_table\_size=256M
45. #指定MYSQL允许的最大连接进程数，如果在访问程序时经常出现TOO MANY CONNECTIONS的错误提示，则需要增大该参数值。
46. max\_connections=5000

49. #指定一个请求的最大连接时间，对于4GB左右内在的服务器来说，可以将其设置为5－10
50. wait\_timeout=120
52. #该参数取值为服务器逻辑CPU数量\*2，比如，服务器有两个物理CPU，每个物理CPU支持HT超线程，所以实际取值4\*2=8，这也是目前双四核主流
53. 服务器的配置。
54. thread\_concurrency=8
56. #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的数据库服务器，则不要开启该选项，否则>将无法正常连接。
57. skip-networking

60. innodb\_flush\_log\_at\_trx\_commit
61. #抱怨Innodb比MyISAM慢 100倍？那么你大概是忘了调整这个值。默认值1的意思是每一次事务提交或事务外的指令都需要把日志写入（flush）
62. 硬盘，这是很费时的。特别是使用电池供电缓存（Battery backed up cache）时。设成2对于很多运用，特别是从MyISAM表转过来的是可以的>，它的意思是不写入硬盘而是写入系统缓存。日志仍然会每秒flush到硬 盘，所以你一般不会丢失超过1-2秒的更新。设成0会更快一点，但安>全方面比较差，即使MySQL挂了也可能会丢失事务的数据。而值2只会在整个操作系统 挂了时才可能丢数据。
63. innodb\_flush\_log\_at\_trx\_commit=2
65. #这是 InnoDB 存储引擎的事务日志所使用的缓冲区。类似于 Binlog Buffer，InnoDB 在写事务日志的时候，为了提高性能，也是先将信息写>入 Innofb Log Buffer 中，当满足 innodb\_flush\_log\_trx\_commit 参数所设置的相应条件（或者日志缓冲区写满）之后，才会将日志写到文>件（或者同步到磁盘）中。可以通过 innodb\_log\_buffer\_size 参数设置其可以使用的最大内存空间。
66. innodb\_log\_buffer\_size=2M
68. #这个数字要根据实际的情况来设定，但对于大多数的情况，是一个比较合适的设置
69. innodb\_thread\_concurrency=8
71. #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
72. 高级 GROUP BY 查询，增加 tmp\_table\_size 值。
73. tmp\_table\_size=64M
75. #随机读取数据缓冲区使用内存(read\_rnd\_buffer\_size)：和顺序读取相对应，当 MySQL 进行非顺序读取（随机读取）数据块的时候，会利用>这个缓冲区暂存读取的数据。如根据索引信息读取表数据，根据排序后的结果集与表进行Join等等。总的来说，就是当数据块的读取需要满足>一定的顺序的情况下，MySQL 就需要产生随机读取，进而使用到 read\_rnd\_buffer\_size 参数所设置的内存缓冲区。
76. read\_rnd\_buffer\_size=16M
78. #你最好在定义数据库命名规则的时候就全部采用小写字母加下划线的组合，而不使用任何的大写字母。
79. lower\_case\_table\_names=1
81. #设置校验模式
82. sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES

85. #默认配置没开查询缓存哦亲
86. explicit\_defaults\_for\_timestamp

innodb\_buffer\_pool\_size=1024Mslow\_query\_loglog-short-format slow\_query\_log\_file=/var/log/mysqld-slow-query.log long-query-time = 2 #log-long-format #log-slow-admin-statementslog-queries-not-using-indexes[mysqld\_safe]log-error=/var/log/mysqld.logpid-file=/var/run/mysqld/mysqld.pid  
东西很多，不要急，下面我们一个个参数来作说明。

## **核心配置文件的中的参数说明**

对于参数说明，为了便于学习而不是塞填鸭式的教育，我把它分为了一个简版，一个升级版，

* 简版的参数说明： 适用于一般开发、测试迅速搭建mysql环境。
* 升级版的参数说明：适用于一般DBA和架构师级别。

### **核心配置参数简版**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. [mysqld]
2. #myslqd服务运行时的端口号
3. port=3306
5. #socket文件是在Linux环境下特有的，用户的客户端软件连接可以不通过TCP/IP网络而直接使用unix socket连接到Mysql。
6. socket=/tmp/mysql.sock
8. #避免Mysql的外部锁定，减少出错几率，增强稳定性。
9. skip-external-locking
11. #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NDS解析的时间。但需要注意的是：如果开启该选项，则所有远程主机连
12. 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。
13. skip-name-resolve
15. #back\_log参数的值指出在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对堆栈中，如果系统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。不同的操作系统在这个队列的大小有自己的限制，如果试图将back\_log设定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。
16. back\_log=384
18. #索引缓冲区大小，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，该参数可设置为256M或384M。如果该参数值设置的过大>反而会使服务器的整体效率降低。
19. key\_buffer\_size=384M
21. #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，必须设定为1024的倍数，单位为字节。
22. max\_allowed\_packet=4M
24. #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192K。
25. thread\_stack=256k

28. #设定查询排序时所能使用的缓冲区大小，系统默认大小为2MB，从5.1.23版本开始，在除了WINDOWS 之外的64位平台上可以4GB的限制。该参数
29. 对应的分配内在是每个连接独占的，如果有100个连接，那么实际分配的总排序缓冲区大小为100\*6＝600MB，那么对于内存4GB左右的服务器来>说，推荐将其设置为6MB-8MB。
30. sort\_buffer\_size=6M
32. #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在也是每个连接独享。
33. read\_buffer\_size=4M

36. #设置Thread Cache池中可以缓存的连接池线程最大数量，可设置为0-16384,默认为0。1GB内存我们配置为8，2GB内存我们配置为16，4GB或4GB以上内在我们配置为64。
37. thread\_cache\_size=64
38. #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
39. Qcache\_hits的值非常大，则表明查询缓冲使用的非常频繁
40. query\_cache\_size=64M
42. #设置内在临时表最大值，如果超过该值，则会将临时表写入磁盘，其范围为1KB至4GB。
43. tmp\_table\_size=256M
45. #指定MYSQL允许的最大连接进程数，如果在访问程序时经常出现TOO MANY CONNECTIONS的错误提示，则需要增大该参数值。
46. max\_connections=5000

49. #指定一个请求的最大连接时间，对于4GB左右内在的服务器来说，可以将其设置为5－10
50. wait\_timeout=120
52. #该参数取值为服务器逻辑CPU数量\*2，比如，服务器有两个物理CPU，每个物理CPU支持HT超线程，所以实际取值4\*2=8，这也是目前双四核主流
53. 服务器的配置。
54. thread\_concurrency=8
56. #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的数据库服务器，则不要开启该选项，否则>将无法正常连接。
57. skip-networking

60. innodb\_flush\_log\_at\_trx\_commit
61. #抱怨Innodb比MyISAM慢 100倍？那么你大概是忘了调整这个值。默认值1的意思是每一次事务提交或事务外的指令都需要把日志写入（flush）
62. 硬盘，这是很费时的。特别是使用电池供电缓存（Battery backed up cache）时。设成2对于很多运用，特别是从MyISAM表转过来的是可以的>，它的意思是不写入硬盘而是写入系统缓存。日志仍然会每秒flush到硬 盘，所以你一般不会丢失超过1-2秒的更新。设成0会更快一点，但安>全方面比较差，即使MySQL挂了也可能会丢失事务的数据。而值2只会在整个操作系统 挂了时才可能丢数据。
63. innodb\_flush\_log\_at\_trx\_commit=2
65. #这是 InnoDB 存储引擎的事务日志所使用的缓冲区。类似于 Binlog Buffer，InnoDB 在写事务日志的时候，为了提高性能，也是先将信息写>入 Innofb Log Buffer 中，当满足 innodb\_flush\_log\_trx\_commit 参数所设置的相应条件（或者日志缓冲区写满）之后，才会将日志写到文>件（或者同步到磁盘）中。可以通过 innodb\_log\_buffer\_size 参数设置其可以使用的最大内存空间。
66. innodb\_log\_buffer\_size=2M
68. #这个数字要根据实际的情况来设定，但对于大多数的情况，是一个比较合适的设置
69. innodb\_thread\_concurrency=8
71. #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
72. 高级 GROUP BY 查询，增加 tmp\_table\_size 值。
73. tmp\_table\_size=64M
75. #随机读取数据缓冲区使用内存(read\_rnd\_buffer\_size)：和顺序读取相对应，当 MySQL 进行非顺序读取（随机读取）数据块的时候，会利用>这个缓冲区暂存读取的数据。如根据索引信息读取表数据，根据排序后的结果集与表进行Join等等。总的来说，就是当数据块的读取需要满足>一定的顺序的情况下，MySQL 就需要产生随机读取，进而使用到 read\_rnd\_buffer\_size 参数所设置的内存缓冲区。
76. read\_rnd\_buffer\_size=16M
78. #你最好在定义数据库命名规则的时候就全部采用小写字母加下划线的组合，而不使用任何的大写字母。
79. lower\_case\_table\_names=1
81. #设置校验模式
82. sql\_mode=NO\_ENGINE\_SUBSTITUTION,STRICT\_TRANS\_TABLES

85. #默认配置没开查询缓存哦亲
86. explicit\_defaults\_for\_timestamp

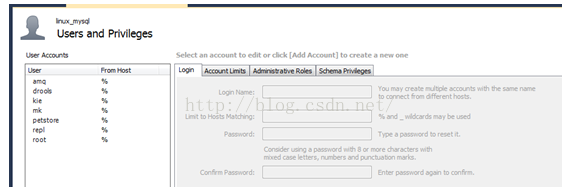
一口气写了这么多，呵呵！  
每一个参数我都作了注解，这样看起来不知道大家是不是会觉得更实用些？  
下面，我们就来一个“升级”版的，所谓升级版内容一定会很丰富，我总結了差不多有****77条****，下面我们就一条条来看吧，这也是本文几个核心价值中的一个重要部分。

### **核心配置参数升级版**

1. [client]  
2. port   = 3306 # 客户端端口号为3306  
3. socket  = /data/3306/mysql.sock  
4. default-character-set = utf8   
# 客户端字符集,(控制character\_set\_client、character\_set\_connection、character\_set\_results)  
5. [mysql]  
6. no-auto-rehash  # 仅仅允许使用键值的updates和deletes  
7. [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。  
8. 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。  
9. 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。  
10. 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。  
11. 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  
12. datadir = /data/3306/data    
# MySQL数据存放文件(极其重要)  
13. 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)  
14. log-error=/data/3306/mysql.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。)  
15. pid-file=/data/3306/mysql.pid    
# MySQL\_pid文件记录的是当前mysqld进程的pid，pid亦即ProcessID。  
16. skip-locking  
# 避免MySQL的外部锁定，减少出错几率，增强稳定性。  
17. 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！  
18. 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！  
19. open\_files\_limit    = 1024  
# MySQLd能打开文件的最大个数,如果出现too mant open files之类的就需要调整该值了。  
20. 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。  
21. 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。  
22. 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。  
23. 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。  
24. 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。  
25. external-locking = FALSE    
# MySQL选项可以避免外部锁定。True为开启。  
26. 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  
27. 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  
28. 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。  
29. 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。  
30. 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。  
31. 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。  
32. query\_cache\_limit = 2M    
# 只有小于此设置值的结果才会被缓存  
33. 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。  
34. default\_table\_type = InnoDB    
# 默认表的类型为InnoDB  
35. 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))  
36. 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。  
37. max\_heap\_table\_size = 64M    
# 独立的内存表所允许的最大容量。  
38. 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。  
39. 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。  
40. long\_query\_time = 1    
# 慢查询的执行用时上限,默认设置是10s,推荐(1s~2s)  
41. 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)  
42. 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)  
43. 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  
44. 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  
45. 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。  
46. 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。  
47. max\_binlog\_cache\_size = 8M    
# 最大的二进制Cache日志缓冲尺寸。  
48. 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。  
49. 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)  
50. 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！  
51. 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。  
52. 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。  
53. 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。  
54. 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  
55. lower\_case\_table\_names = 1    
# 实现MySQL不区分大小。(发开需求-建议开启)  
56. 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)。  
57. replicate-ignore-db=mysql    
# 在做主从的情况下,设置不需要同步的库。  
58. 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。  
59. 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。  
60. 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。  
61. 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.  
62. myisam\_recover  
# 自动检查和修复没有适当关闭的 MyISAM 表.  
63. 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。  
64. 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。  
65. 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。  
66. 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  
67. 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。  
68. 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。  
69. innodb\_log\_buffer\_size = 2M    
# 默认为1MB，通常设置为8~16MB就足够了。  
70. 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。  
71. 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。  
72. innodb\_max\_dirty\_pages\_pct = 90    
# InnoDB主线程刷新缓存池中的数据。  
73. 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。  
74. 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。）  
75. [mysqldump]  
  
  
76. quick  
#quick  不缓冲查询，直接导出至stdout  
77. 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。

# **mySQL配置远程可连接**

mySQL安装完后，默认只能在本机进行SQL客户端的访问，如果你要使用一台远程主机，比如説：  
mySQL安装在192.168.0.101上，你要通过192.168.0.1上的mySQL客户端登录192.168.0.101访问，默认是不允许的。  
为了能够远程访问mySQL你必须配置远程可连接的用户信息。  
一般来説：  
为了简单，我们可以配置一个用户，用户名为“username@****%****”，这个****%****代表支持远程任何IP地址的客户端可以访问这台mySQL主机。  
但是，一般来説为了安全，我们会把username@后面跟上一个具体的ip。  
所以如果在mySQL刚安装完毕时，root用户只可以作本地访问，为了开启远程访问功能我们一般都有一个root@%这样的用户名。



操作步骤如下：  
1. 先登录本地mysql通过命令：

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. #mysql -uroot –proot的密码 –P(端口号大写的P) -h 主机名
2. mysql -uroot –p111111 -P3307 -h 168.177.101.1

2. 在mysql命令符内执行如下语句:

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY 'password' WITH GRANT OPTION;

# **结合配置中的慢查询，统计有性能问题的SQL**

在我的配置中有这么一段描述

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. [mysqld]
2. …
3. …
4. slow\_query\_log
5. log-short-format
6. slow\_query\_log\_file=/var/log/mysqld-slow-query.log
7. long-query-time = 2
8. #log-long-format
9. #log-slow-admin-statements
10. log-queries-not-using-indexes

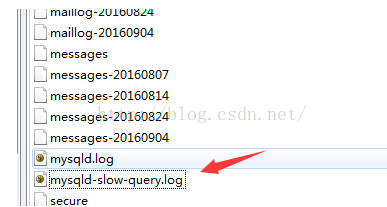
此处注意了，****网上对于“慢查询”的配置，很多都已经deprecated了，请参照我上述这种写法****。

## **配置解释**

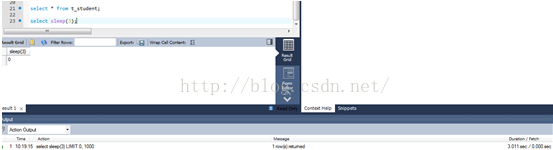
* 首先，我开启了query\_log功能。
* 在日志查询中我使用的是log-short-format格式而不是long-format，因为这样我的mySQL日志更紧凑，要不然一天下来光统计这些日志就要耗掉G的空间了。
* 我指明了我的“慢查询”日志生成的路径位于何处。
* 我配置了对于超过2秒的查询SQL全部记录在慢查询日志中的功能。
* 同时，我还开启了在查询时凡是没有用到索引的SQL全部记录在慢查询日志中。

## **试验**

我们配置完后使用service mysqld restart重启mysql服务后，我们可以看到在/var/log目录下面会有一个mysqld-slow-query.log的文件 。



我们在sql客户端执行如下语句



接下去我们来看看这个mysqld-slow-query.log中的文件内容：



这样看，似乎有一些不太习惯对不对？  
如果这个日志文件中有成千上百个日志，我们如果需要统计 Top10最慢的那些SQL分别是哪些。。。怎么办？  
我们可以使用mysqldumpslow命令

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52582789" \o "copy)

1. mysqldumpslow -s t -t 10 mysqld-slow-query.log



# **mySQL 存储引擎中InnoDB与Myisam的主要区别**

这个问题，我曾经在面试时碰到过有人説“精通MYSQL“，于是我问了他们这么一个问题，结果是超过90%的人无法回答。  
mySQL最重要的两种存储引擎InnoDB与Myisam，这是基础，一定要知道，下面来看。

1） 事务处理  
innodb 支持事务功能，myisam 不支持。  
Myisam 的执行速度更快，性能更好。  
2） select ,update ,insert ,delete 操作  
MyISAM：如果执行大量的SELECT，MyISAM是更好的选择  
InnoDB：如果你的数据执行大量的INSERT或UPDATE，出于性能方面的考虑，应该使用InnoDB表  
3） 锁机制不同  
InnoDB 为行级锁，myisam 为表级锁。  
注意：当数据库无法确定，所找的行时，也会变为锁定整个表。  
如： update table set num = 10 where username like "%test%";  
4） 查询表的行数不同  
MyISAM：select count(\*) from table,MyISAM只要简单的读出保存好的行数，注意的是，当count(\*)语句包含   where条件时，两种表的操作是一样的  
InnoDB ： InnoDB 中不保存表的具体行数，也就是说，执行select count(\*) from table时，InnoDB要扫描一遍整个表来计算有多少行  
5） 物理结构不同

****MyISAM ：****每个MyISAM在磁盘上存储成三个文件。第一个文件的名字以表的名字开始，扩展名指出文件类型。

* .frm文件存储表定义。
* 数据文件的扩展名为.MYD (MYData)。
* 索引文件的扩展名是.MYI (MYIndex)

****InnoDB：****基于磁盘的资源是InnoDB表空间数据文件和它的日志文件，InnoDB 表的大小只受限于操作系统文件的大小，一般为 2GB

6） anto\_increment 机制不同

这是一道很早的面试题：

一张表,里面有ID自增主键,当insert了17条记录之后,删除了第15,16,17条记录,再把Mysql重启,再insert一条记录,这条记录的ID是18还是15 。

****答案：****  
如果表的类型是MyISAM，那么是18。   
因为MyISAM表会把自增主键的最大ID记录到数据文件里，重启MySQL自增主键的最大ID也不会丢失。   
如果表的类型是InnoDB，那么是15。   
InnoDB表只是把自增主键的最大ID记录到内存中，所以重启数据库或者是对表进行OPTIMIZE操作，都会导致最大ID丢失。

其他） 为什么MyISAM会比Innodb 的查询速度快。  
INNODB在做SELECT的时候，要维护的东西比MYISAM引擎多很多；

* 数据块，INNODB要缓存，MYISAM只缓存索引块，  这中间还有换进换出的减少；
* innodb寻址要映射到块，再到行，MYISAM 记录的直接是文件的OFFSET，定位比INNODB要快；
* INNODB还需要维护MVCC一致；虽然你的场景没有，但他还是需要去检查和维护；
* MVCC ( Multi-Version Concurrency Control )多版本并发控制

****InnoDB：**** 通过为每一行记录添加两个额外的隐藏的值来实现MVCC，这两个值一个记录这行数据何时被创建，另外一个记录这行数据何时过期（或者被删除）。但是InnoDB并不存储这些事件发生时的实际时间，相反它只存储这些事件发生时的系统版本号。这是一个随着事务的创建而不断增长的数字。每个事务在事务开始时会记录它自己的系统版本号。每个查询必须去检查每行数据的版本号与事务的版本号是否相同。让我们来看看当隔离级别是REPEATABLE READ时这种策略是如何应用到特定的操作的：

SELECT InnoDB必须每行数据来保证它符合两个条件：

InnoDB必须找到一个行的版本，它至少要和事务的版本一样老(也即它的版本号不大于事务的版本号)。这保证了不管是事务开始之前，或者事务创建时，或者修改了这行数据的时候，这行数据是存在的。

这行数据的删除版本必须是未定义的或者比事务版本要大。这可以保证在事务开始之前这行数据没有被删除。

# **mySQL使用profiling分析慢sql语句的原因**

MySQL 的 Query Profiler 是一个使用非常方便的 Query 诊断分析工具，通过该工具可以获取一条Query 在整个执行过程中多种资源的消耗情况，如 CPU，IO，IPC，SWAP 等，以及发生的 PAGE FAULTS，CONTEXT SWITCHE 等等，同时还能得到该 Query 执行过程中 MySQL 所调用的各个函数在源文件中的位置。  
****MySQL5.0.37****版本以上支持PROFILING调试功能，让您可以了解SQL语句消耗资源的详细信息。因为它需要调用系统的getrusage()函数，所以只是在Linux/Unix类平台上才能使用，而不能在Windows平台上使用。而且，PROFILING是针对处理进程(process)而不是线程(thread)的，服务器上的其他应用，可能会影响您的调试结果，因此，这个工具适合开发过程中的调试，如果要在生产环境中调试使用，则要注意它的局限性。

## **查看是否已经启用profile，默认是关闭的**

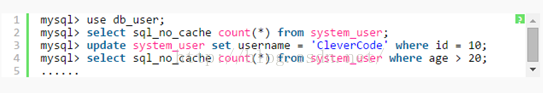


## **启用profiling（变量profiling是用户变量每次都得重新启用）**

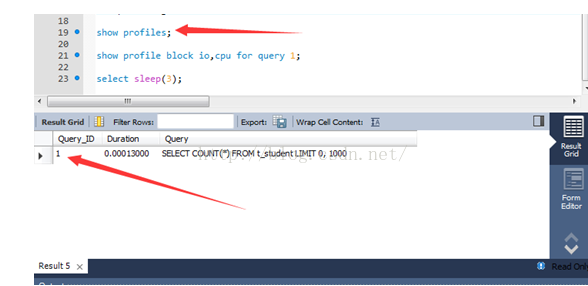


## **使用profiling记录用户执行的SQL**

为避免之前已经把 SQL 存放在 QCACHE 中， 建议在执行 SQL 时， 强制 SELECT 语句不进行 QCACHE 检测。这样可以提交分析的准确性。



## **使用show profile查询最近一条语句的执行信息**



## **查看在服务器上执行语句的列表。(查询id，花费时间，语句)**

