# **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%

# **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。

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监控管理工具介绍**

MySQL越来越被更多企业接受，随着企业发展，MySQL存储数据日益膨胀，MySQL的性能分析、监控预警、容量扩展议题越来越多。“工欲善其事，必先利其器”，那么我们如何在进行MySQL性能分析、监控预警、容量扩展问题上得到更好的解决方案，就要利用各种工具来对MySQL各种指标进行分析。本文是读书笔记，下面提及的工具，读者可能都用过，或打算准备是使用。  
MySQL服务器的发布包没有包含那些能完成许多常见任务的工具，例如监控服务器的工具、比较服务器间数据的工具。我们把这些工具分成以下几类：界面、监控、分析和辅助工具。

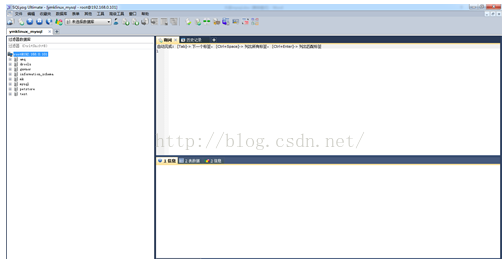
## **带界面的工具**

### **1、MySQL可视化工具**

这些工具都可以免费使用:   
a、MySQL查询浏览器（MySQL Query Browser）：这个不用说了…  
b、MySQL管理员（MySQL Administrator）：功能集中在服务器管理上，所以它最适合DBA使用，而不是开发人员和分析人员。它可以帮助DBA把创建备份 、创建用户并分配权限、显示服务器日志和状态信息等过程进行自动化处理。它还包括了一些基本的监控功能，例如图形化的状态变量显示，但是它没有下文里会提到的交互式监控工具那么灵活。  
c、MySQL迁移工具箱（MySQL Migration Tookit）：可以帮你把数据从别的数据库系统迁移到MySQL里。  
d、MySQL工作台（MySQL Workbench）：MySQL的建模工具。

### **2、SQLyog**

SQLyog是最常用的MySQL可视化工具，只能用于win下。



### **3、phpMyAdmin**

phpMyAdmin是一款很流行的管理工具，它基于Web界面来管理MySQL服务器。有很多支持者，但反对者也不少。

### **监控工具**

MySQL监控是一个很大复杂的任务，不同的应用有着不同的需求。“监控”是大家滥用的术语之一，承载了几重意思。我们讲到的监控工具被分为非互动的和互动的两类。非互动监控常常就是一个自动化系统，它接收系统的测量值，如果有超出安全范围的，就通过发出警告提醒管理员。互动监控工具可以让你实时地观测服务器。   
你可能对监控工具在其他方面的差别也很感兴趣，例如被动监控和主动监控，后者会发送报警信息并作出初步反应（就像Nagios一样）；或者你可能正在寻找这样一工具：它可以创建一个信息仓库，而不仅仅是显示当前的统计信息。有许多监控系统不是专为监控MySQL而设计的，它们就是一个通用系统，里面设计了一个周期性的任务，定时去检查各类资源的状态，例如像服务器、路由器、以及各种软件（包括MySQL）。它们常常会提供一个插件架构，同时有一个预订的MySQL插件可供使用。这样的一些系统能够记录监控对象的状态，并通过web界面用图形化的形式表示出来。当监控对象出现问题，或者状态值超过安全范围时，它们还能发送报警信息，或者执行一个初始化的动作。  
非交互性工具

Nagios

监控界流行一句话：只要用顺了Nagios，你就永远不会再想到其它监控系统。  
对于Nagios本blog会持续推出相关文章，现在只是引用Nagios官方的介绍：  
Nagios是一款用于系统和网络监控的应用程序。它可以在你设定的条件下对主机和服务进行监控，在状态变差和变好的时候给出告警信息。  
Nagios最初被设计为在Linux系统之上运行，然而它同样可以在类Unix的系统之上运行。  
Nagios更进一步的特征包括：监控网络服务（SMTP、POP3、HTTP、NNTP、PING等）；  
监控主机资源（处理器负荷、磁盘利用率等）；  
简单地插件设计使得用户可以方便地扩展自己服务的检测方法；  
并行服务检查机制；  
具备定义网络分层结构的能力，用”parent”主机定义来表达网络主机间的关系，这种关系可被用来发现和明晰主机宕机或不可达状态；  
当服务或主机问题产生与解决时将告警发送给联系人（通过EMail、短信、用户定义方式）；  
具备定义事件句柄功能，它可以在主机或服务的事件发生时获取更多问题定位；  
自动的日志回滚；可以支持并实现对主机的冗余监控；可选的WEB界面用于查看当前的网络状态、通知和故障历史、日志文件等；  
下面是可代替Nagios的工具：

****Zenoss****

Zenoss是用Python编写的，拥有基于浏览器的用户界面，并使用了Ajax使操作更加快捷而富有效率。它将监控、报警、趋势显示、图表显示和记录历史数据等功能合成在一个统一的工具里，它还能在网上自动发现资源，在默认情况下，Zenoss使用SNMP从远程机器上收集数据，但它也可以使用SSH，并且支持Nagios插件。

****Hyperic HQ****  
Hyperic HQ是一款基于Java的监控系统，它的目标跟其他同类别的软件不太一样，它要成为企业级的监控系统。跟Zenoss一样，它也能自动发现资源，支持Nagios插件，但是它的逻辑组织和架构很不一样，显得有点庞大。至于它是不是合适你的需求，那要看你的参数设置和监控的方式了。

****OpenNMS****  
OpenNMS是由Java编写的，拥有一个活跃的开发者社区。它具备了常规的功能，例如监控和报警，也加入了图表和趋势显示的功能。它的目标是高性能、伸缩性、自动化以及良好的兼容性。跟Hyperic一样，它也企图成为一款企业级的监控软件，可以用于大型的关键系统上。

****Groundwork Open Source****  
Groundwork Open Source实际上是基于Nagios的，它把Nagios和其他几个工具集成为一个系统，并安上一个统一的门户界面。描述它的最好方法可能就是：如果你对Nagios、Cacti及其他工具很熟悉，并且能够花大量的时间把它们无缝地集成在一起的话，你也能在家庭作坊里做一个出来。

****Zabbix****Zabbix是一个开源监控系统，在许多方面跟Nagios很相像，但是也有一些关键的不同点。例如：它把所有配置信息和其他数据都存放在一个数据库里，而不是放在配置文件里；它比Nagios存储了更多类型的数据，这样可以生成更好的趋势图和历史报告。它的网络图表和可视化功能也优于Nagios。很多使用它的人发现它更易配置，更具有兼容性。说起来它也能比Nagios少，它的报警功能也不够高级。

### **MySQL监控和建议服务**

MySQL自己的监控方案就是设计用来监控MySQL实例的，但也能够监控主机的一些关键方面。这个工具不是开源的，需要MySQL企业订阅费。  
MONyog  
MONyog是一个轻量级的无代理的监控系统，它跟以上那些工具有着不同实现方法：它的底层是一个JavaScript引擎，所有配置都是通过JavaScrpt对象模型来完成的。它被设计为在桌面系统上运行，运行时它会在一个闲置的端口上打开一个HTTP监听器。这样，你就可以把你的浏览器指向这个端口，查看MySQL服务器的信息了，这信息都是结合了Javascript和Flash来表示的。MONyog实际上有交互式和非交互式两种类型，因此，你可以把两种类型的监控功能都尝试着用用看。

### **基于RRDTool的系统**

严格地说，RRDTool不算是一个监控系统，但是，它很重要，有必要在此提到一下。很多组织里都是使用几种脚本或程序–这些一般都是自制的–从服务器那里读取信息，然后再保存到循环数据库（Round-robin database，RRD）文件里。在许多要获取记录生成图表的环境下，RRD文件是一个很合适的解决方案。它们能聚合输入的数据，如果输入数据值没有按期在随后提交进行时，还能在随后插入这些丢失的数据。它们还都带有强大的图表工具，能够生成漂亮的与众不同的图表。现在已经有一些基于RRDTOOL的系统可供使用了。

a、Muti Router Traffic Grapher，或者叫MRTG就是一款典型的基于RRDTOOL的系统。它真正的设计初衷是记录网络数据流，但是它也被扩展用来记录和图表化表示其他一些东西。

b、Munin是一个能为你采集数据的系统，将它放入RRDTool后，就会根据数据生成不同粒度的图表。它能从配置信息里生成静态的HTML文件，这样你就可以轻松地浏览，查看趋势情况。

c、Cacti是另外一个常用的图表和趋势显示系统。它的工作方式是：从系统里获取数据，然后保存在RRD文件里，然后用PHP Web界面的形式，使用RRDTool把数据以图表的形式展示出来。这个显示界面也是配置和管理界面（配置信息存储在一个MySQL服务器里）。它是模板驱动的，因此，你可以自己定义模板，并放到你的系统里使用。

d、Cricket是一个用Perl编写的跟Cacti类似的系统，使用的是基于文件的配置系统。Ganglia也跟Cacti类似，但它的设计初衷是永远监控群集和系统网络，因此，你可以查看到由许多服务器信息聚合得到的结果，也可以按照你的意愿，查看单独某台服务器的信息。（Cacti和Cricket无法显示聚合数据。）  
以上这些系统都可以被用作MySQL系统信息的收集、记录、图表化数据和报告，它们在用途方面差异较小，都具备了不同程度的兼容性。但是，它们缺乏真正意义上的兼容性，比如当某些东西出错时，它要能够有针对性地发送报警信息给某些人。它们中的一些甚至没有“错误”的概念。所以，有些人把这一点看作是此类系统的一大缺点，觉得最好还是把记录、图表化表示、报警这几项功能都独立开来。事实上，Munin特地设计了使用Nagios来作为它的报警系统。然而，对于其他几个来说，这的确是缺点。另外还有一个缺点就是安装和配置这样一个系统，使其能完全满足你需求，须投入很多时间和努力，不过，这一点也并不是这样。

最后，RRD文件无法让你使用SQL或其他标准方法来查询它里面的数据。而且，在默认情况下，它永远会以一种恰好的粒度来存储数据，许多MySQL管理员就不愿意接受这种限制，转而选择一个关系数据库来存储这些历史数据。

### **交互性工具**

交互性工具就是那些在你需要时就可以启动起来，并以视图显示的形式不断获取最新服务器状态的软件。

#### **innotop**

innotop是一个通过文本模式显示MySQL和InnoDB的监测工具。它有很多特点，快速的配置的，易于使用等。它吸取了MYTOP的精华这使它变得更加强大。innotop是用PERL语言写成的，这是它能更加灵活的使用在各种操作平台之上，它能详细的的监控出当前MYSQL和INNODB运行的状态，以便维护人员根据结果合理的优化MYSQL，让MYSQL更稳定更高效的运行。安装INNOTOP工具非常的简单，既然是由PERL写的，当然需要PERL环境和相关的工具包。在安装之前先要确定你的系统安装了Time::HiRes,Term::ReadKey,DBI,DBD::mysql这四个包。安装可以把包下载下来通过编译安装完成也可以用PERL模块安装方式来完成。

#### **mtop**

是一个显示MySQL服务器查询状态的监视器。功能包括显示完成的查询进程，显示查询优化的信息及杀死一个查询。附加功能包括服务器性能统计，配置信息和调整技巧提示。

#### **mytop**

mytop就是類似top的MySQL監察工具。執行mytop後，它會每隔幾秒更新一次，而且也可以針對性地監察某一個資料庫。

## **分析工具**

分析工具可以帮你自动化那些单调乏味的工作，如监测服务器，找出还可以优化和调优的功能区域。这些工具可以作为解决性能问题的良好开端。

### **HackMySQL 工具（这个很出名，可惜已停止更新）**

1、Mysqlreport  
2、Mysqlsla  
3、Maatkit分析工具

### **MySQL的辅助工具**

MySQL里有几个工具是为了消除MySQL提供的功能与它自带的命令行工具之间的隔阂。  
a、MySQL Proxy  
b、Dormando的MySQL代理

# **phpMyAdmin安装详细步骤**

## **安装nginx**

Nginx需要依赖下面3个包

1. gzip 模块需要 zlib 库 ( 下载: http://www.zlib.net/ )  zlib-1.2.8.tar.gz  
2. rewrite 模块需要 pcre 库 ( 下载: http://www.pcre.org/ )  pcre-8.21.tar.gz  
3. ssl 功能需要 openssl 库 ( 下载: http://www.openssl.org/ )  openssl-1.0.1.tar.gz

****注意：****

****如果用源码安装的话，后面nginx安装的时候需要指定 --with-pcre 对应的压缩包路径，如果用二进制包安装则不需指定  
依赖包一键安装： yum -y install zlib zlib-devel openssl openssl--devel pcre pcre-devel****

### 采用源码包安装命令

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. openssl：
2. tar -xzvf openssl-1.0.1.tar.gz
3. cd openssl-1.0.1
4. ./config(注意) && make && make install
6. pcre:
7. tar -xzvf pcre-8.21.tar.gz
8. cd pcre-8.21
9. ./configure  && make && make install
11. zlib:
12. tar -xzvf zlib-1.2.8.tar.gz
13. cd zlib-1.2.8
14. ./configure  && make && make install

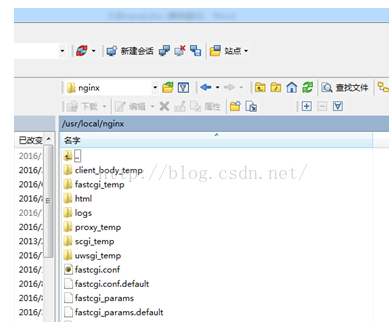
### **nginx安装命令**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. ./configure --sbin-path=/usr/local/nginx/nginx --conf-path=/usr/local/nginx/nginx.conf --pid-path=/usr/local/nginx/nginx.pid --with-http\_ssl\_module --with-pcre=../pcre-8.38 --with-zlib=../zlib-1.2.8 --with-openssl=../openssl-1.0.2g
2. --with-http\_stub\_status\_module --user=nginx --group=nginx

### **安装后的配置**

本文主要讲mySQL对于nginx的优化和基本操作不作描述，读者可以自行参考相关文档。  
按照上述过程nginx会被安装在 /usr/local/nginx目录。



因此在更改了nginx的核心配置文件nginx.conf后可以使用如下命令进行启动

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. /usr/local/nginx/nginx –c /usr/local/nignx/nginx.conf

## **安装FCGI组件**

Nginx不像apache，默认支持php功能具有php相关的模块。  
Nginx是通过fcgi套件结合php来实现支持解析php功能的。

### **安装fcgi**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. yum install –y spawn-fcgi fcgi-devel fcgi

以上步骤执行完毕后，我们安装了：

****spawn-fcgi****

这个东西特别有意思，因为nginx默认是不支持cgi的，而gitweb是用cgi写的，因此我们才安装fastcgi，而fastcgi又要通过spawn-fcgi来启动。。。因此。。。必须要装spawn-fcgi。

****fcgi-devel 和 fcgi****它们都属于fastcgi运行时的lib库。

从以下网址下载该组件：  
[https://codeload.github.com/gnosek/fcgiwrap/legacy.tar.gz/master](https://codeload.github.com/gnosek/fcgiwrap/legacy.tar.gz/master" \t "http://blog.csdn.net/lifetragedy/article/details/_blank)  
运行以下命令

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. cd fcgiwrap
2. autoreconf –I
3. configure
4. make
5. make install

全部安装完后注意检查以下必要文件是否存在

* /etc/init.d 目录下有一个spawn-fcgi程序
* /usr/bin 目录下有一个spawn-fcgi程序
* /var/www/git目录下是gitweb所有的文件（网页版git）
* /etc/目录下有一个gitweb.conf文件
* /etc/sysconfig/目录下有一个spawn-fcgi文件

### **修改/etc/sysconfig/spawn-fcgi文件**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. # You must set some working options before the "spawn-fcgi" service will work.
2. # If SOCKET points to a file, then this file is cleaned up by the init script.
3. #
4. # See spawn-fcgi(1) for all possible options.
5. #
6. # Example :
7. #SOCKET=/var/run/php-fcgi.sock
8. #OPTIONS="-u apache -g apache -s $SOCKET -S -M 0600 -C 32 -F 1 -P /var/run/spawn-fcgi.pid -- /usr/bin/php-cgi"

以上是原来文件的内容  
修改后的内容如下显示：

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

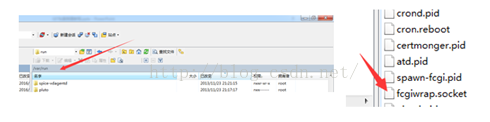
1. # You must set some working options before the "spawn-fcgi" service will work.
2. # If SOCKET points to a file, then this file is cleaned up by the init script.
3. #
4. # See spawn-fcgi(1) for all possible options.
5. #
6. # Example :
7. #SOCKET=/var/run/php-fcgi.sock
8. #OPTIONS="-u apache -g apache -s $SOCKET -S -M 0600 -C 32 -F 1 -P /var/run/spawn-fcgi.pid -- /usr/bin/php-cgi"
10. FCGI\_SOCKET=/var/run/fcgiwrap.socket
11. FCGI\_PROGRAM=/usr/local/sbin/fcgiwrap
12. FCGI\_USER=nginx
13. FCGI\_GROUP=nginx
14. FCGI\_EXTRA\_OPTIONS="-M 0700"
15. OPTIONS="-u $FCGI\_USER -g $FCGI\_GROUP -s $FCGI\_SOCKET -S $FCGI\_EXTRA\_OPTIONS -F 1 -P /var/run/spawn-fcgi.pid -- $FCGI\_PROGRAM"

### **启动spawn-fcgi**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. chkconfig --levels 2345 spawn-fcgi on
2. chkconfig --levels 2345 php-fpm on
3. /etc/init.d/spawn-fcgi start
4. service php-fpm start

把spawn-fcgi设为开机启动，并启动该服务，该服务成功启动后会在：  
/var/run目录下生成一个fcgiwrap.socket文件。



## **安装php5模块**

我这边使用的是php-5.6.2，下载地址为：  
[http://cn2.php.net/distributions/php-5.6.2.tar.gz](http://cn2.php.net/distributions/php-5.6.2.tar.gz" \t "http://blog.csdn.net/lifetragedy/article/details/_blank)  
使用命令：

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. wget http://cn2.php.net/distributions/php-5.6.2.tar.gz

### **安装libmcrypt**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. yum  install  php-mcrypt  libmcrypt  libmcrypt-devel

### **编译和安装php5**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. ./configure --prefix=/usr/local/php --with-config-file-path=/usr/local/php/etc --enable-fpm
2. --with-fpm-user=php-fpm --with-fpm-group=php-fpm --with-mysql=mysqlnd
3. --with-mysql-sock=/var/lib/mysql/mysql.sock --with-libxml-dir --with-gd --with-jpeg-dir --with-png-dir
4. --with-freetype-dir --with-iconv-dir --with-zlib-dir --with-mcrypt --enable-soap --enable-gd-native-ttf --enable-ftp --enable-mbstring --enable-exif --disable-ipv6 --with-pear --with-curl --with-openssl
5. Make && make install

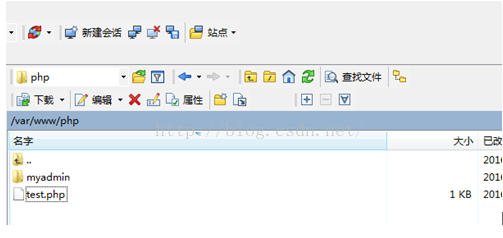
### **测试php5与nginx的结合**

编写一个测试php文件为test.php，内容如下：

**[php]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. <?php
2. phpinfo();
3. ?>

把它放于/var/www.php目录内



### **配置nginx**

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. server {
2. error\_log logs/php.error.log;
3. access\_log logs/php.access.log;
4. listen       82;
5. server\_name  192.168.0.101;
6. root /var/www/php;
7. index index.php;
8. location ~ .php$ {
9. gzip off;
10. #fastcgi\_pass unix:/var/run/fcgiwrap.socket;
11. fastcgi\_pass   127.0.0.1:9000;
12. fastcgi\_index index.php;
13. fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;
14. include fastcgi\_params;
15. fastcgi\_connect\_timeout 300;
16. fastcgi\_send\_timeout 300;
17. fastcgi\_read\_timeout 300;
18. }
19. location = /favicon.ico {
20. log\_not\_found off;
21. access\_log off;
22. }
23. location ~ /\.ht {
24. deny all;
25. }
27. }

### **运行测试.php**

我们打开一个ie使用如：http://192.168.0.101:82/test.php来访问我们的测试页，如果你得到下面类似的界面，那就说明你的php已经和nginx结合成功了，如果你遇到下面这几个错误，本文将给出解决方法（网上的基本都是不对的）。

#### **Nginx在访问时IE出现502 bad gateway错误解决方法**

nginx出现502有很多原因，但大部分原因可以归结为资源数量不够用,也就是说后端php-fpm处理有问题，nginx将正确的客户端请求发给了后端的php-fpm进程，但是因为php-fpm进程的问题导致不能正确解析php代码，最终返回给了客户端502错误。  
服务器出现502的原因是连接超时 我们向服务器发送请求 由于服务器当前链接太多，导致服务器方面无法给于正常的响应,产生此类报错  
因此如果你服务器并发量非常大，那只能先增加机器，然后按以下方式优化会取得更好效果;但如果你并发不大却出现502，一般都可以归结为配置问题，脚本超时问题。

****php-fpm进程数不够用****

使用 netstat -napo |grep "php-fpm" | wc -l 查看一下当前fastcgi进程个数，如果个数接近conf里配置的上限，就需要调高进程数。  
但也不能无休止调高，可以根据服务器内存情况，可以把php-fpm子进程数调到100或以上，在4G内存的服务器上200就可以。  
我们的php是安装在/usr/local/php目录下，更改/usr/local/php/php-fpm.conf文件，找到下面这一行。

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. pm.max\_children = 10

也有可能这一行是被注释掉的，把它放开后改动如下：

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. pm.max\_children = 100

****调高调高linux内核打开文件数量****

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. echo 'ulimit -HSn 65536' >> /etc/profile
2. echo 'ulimit -HSn 65536' >> /etc/rc.local
3. source /etc/profile

****脚本执行时间超时****

如果脚本因为某种原因长时间等待不返回 ，导致新来的请求不能得到处理，可以适当调小如下配置。  
nginx.conf里面主要是如下

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. fastcgi\_connect\_timeout 300;
2. fastcgi\_send\_timeout 300;
3. fastcgi\_read\_timeout 300;

php-fpm.conf里如要是如下

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. request\_terminate\_timeout = 10s

****缓存设置比较小****  
修改或增加配置到nginx.conf

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. proxy\_buffer\_size 64k;
2. proxy\_buffers  512k;
3. proxy\_busy\_buffers\_size 128k;

#### **报ERROR: cannot get uid for user '@php\_fpm\_user@'错**

修改php的配制文件（此例我们在/usr/local/php/etc/目录下的php-fpm.conf文件）

**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. user = php-fpm
2. group = php-fpm

在文件中加入上述两行，同时在centos中加入相应的用户和组重启php-fpm服务即可。

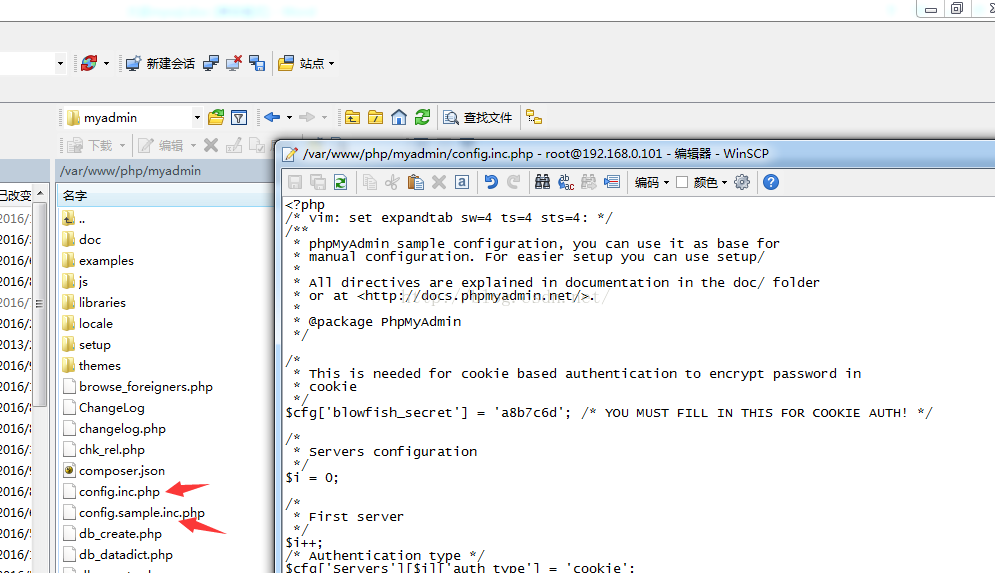
## **安装phpMyAdmin**

我这边使用的是phpMyAdmin-4.0.4.2-all-languages.zip。

****注：****  
phpMyAdmin版本和mySql版本有对应关系，目前较高版的phpMyAdmin支持的是mySql5.x及以上版本。  
因为我安装的是mySQL5.1.x，因此我只能用phpMyAdmin4.0.x。  
我们把phpMyAdmin-4.0.4.2-all-languages.zip解压后的内容全部copy进/var/www/php/目录下的/myadmin目录内。

### **配置config文件**

把/var/www/php/myadmin目录下的config.sample.inc.php文件copy成另一份并命名成config.inc.php的文件并按照如下格式修改：

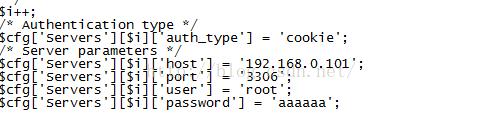


#### **mysql主机信息**

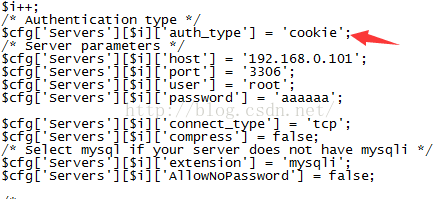
**[plain]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. $cfg['Servers'][$i]['host']
2. $cfg['Servers'][$i]['port']
3. $cfg['Servers'][$i]['user']
4. $cfg['Servers'][$i]['password']

把这些值按照你需要管理的mysql的主机信息一一填写完整，user和password你可以使用mysql的root用户信息。



#### **认证方式**

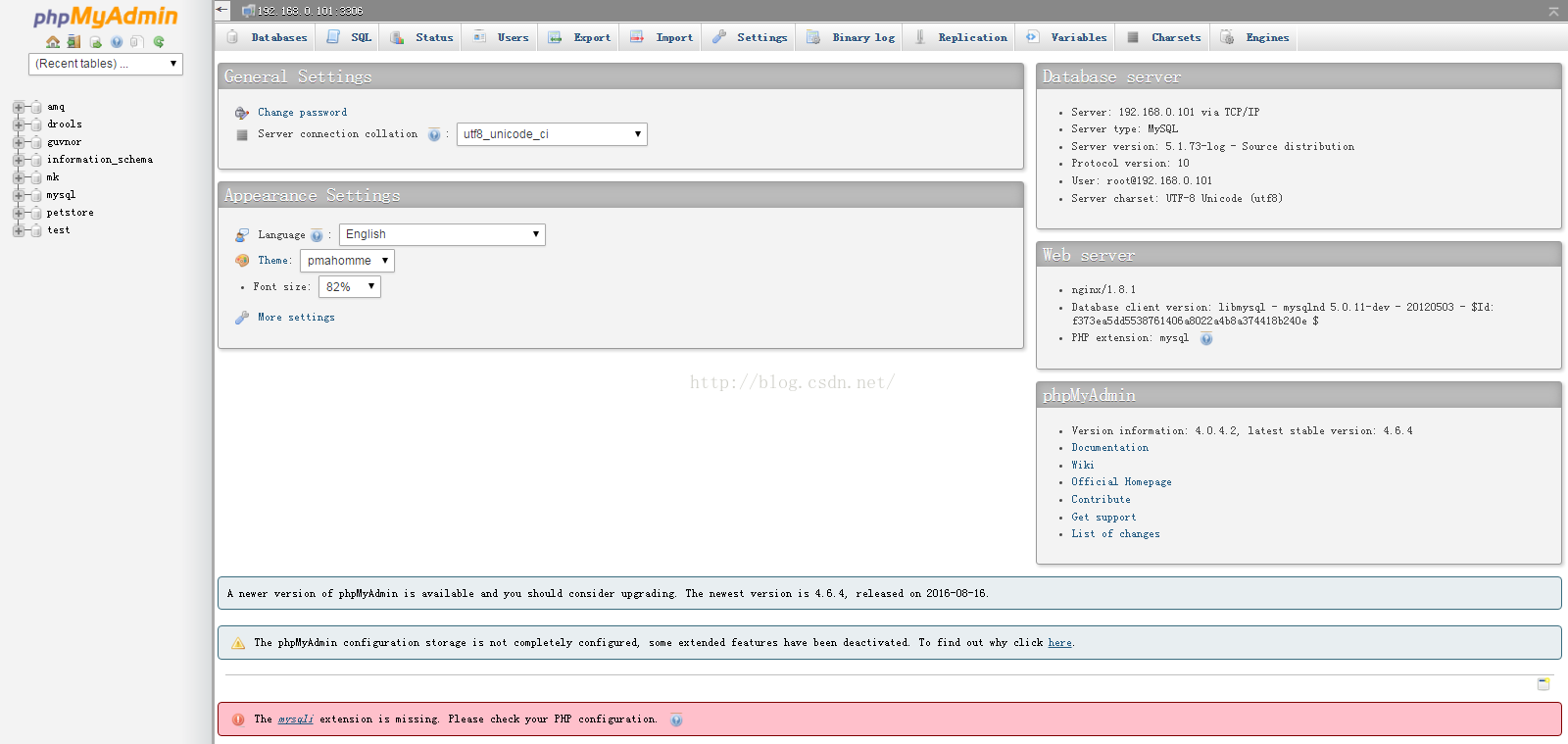


#### **短语密码(blowfish\_secret)的设置**

$cfg['blowfish\_secret'] = '';   
如果认证方法设置为cookie，就需要设置短语密码，置于设置为什么密码，由您自己决定  ，但是不能留空，否则会在登录phpmyadmin时提示错误，如：我设置的就是secret。

### **初始化phpMyAdmin**

一切就绪后你可以使用http://192.168.0.101:82/myadmin/这样的网址登录你的phpMyAdmin了，第一次登录会要求你输入“短语密码”，我们输入事先配置好的secret，然后在用户名和密码处输入你要管理的mysql实例的root用户名和密码即可以通过phpMyAdmin来管理你的mySQL实例了。



通过phpMyAdmin你可以感受到相当强大的图形化mySQL管理功能。

# **mySQL 的批量操作-rewriteBatchedStatements参数**

## **默认mySQL的驱动是不支持批量操作的即使用了addBatch**

使用 jdbc的批量操作，就是PreparedStatement 类上的addBatch(),executeBatch()方法。   
在这里要提醒一下大家，MySql的JDBC驱动，是不支持批量操作的，就算你在代码中调用了批量操作的方法，MySql的JDBC驱动，也是按一般insert操作来处理的。   
同样Fetch Size特性MySql的JDBC驱动也不支持。而Oracle的JDBC驱动是都支持的。   
如果你以前使用的是Mysql数据库， 不要指望通过批处理来提高性能了。因此这才有了rewriteBatchedStatements参数的用法。

## **mySQL的批量操作设置**

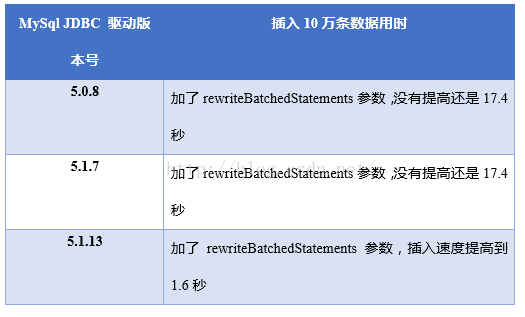
MySql的JDBC连接的url中要加rewriteBatchedStatements参数，并保证5.1.13以上版本的驱动，才能实现高性能的批量插入。  
例如：

**[java]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. String connectionUrl="jdbc:mysql://192.168.1.100:3306/test?rewriteBatchedStatements=true" ;

还要保证Mysql JDBC驱的版本相匹配。

## **Mysql JDBC驱动，各个版本测试结果**



## **通过一个例子来作对比**

### **Java核心代码-非批量（非addBatch）**

**[java]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. **try** {
2. PreparedStatement prest = conn.prepareStatement(sql);
3. **long** a=System.currentTimeMillis();
4. **for**(**int** x = 1; x <= count; x++){
5. prest.setInt(1, x);
6. prest.setString(2, "张三");
7. prest.execute();
8. **if**(x%point==0){
9. conn.commit();
10. }
11. }
12. **long** b=System.currentTimeMillis();
13. print("MySql非批量插入10万条记录",a,b,point);
14. } **catch** (Exception ex) {
15. ex.printStackTrace();
16. }**finally**{
17. close(conn);
18. }

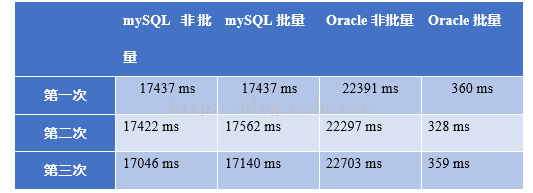
### **Java核心代码-批量（addBatch）**

**[java]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. **try** {
2. PreparedStatement prest = conn.prepareStatement(sql);
3. **long** a=System.currentTimeMillis();
4. **for**(**int** x = 1; x <= count; x++){
5. prest.setInt(1, x);
6. prest.setString(2, "张三");
7. prest.addBatch();
8. **if**(x%point==0){
9. prest.executeBatch();
10. conn.commit();
11. }
12. }
13. **long** b=System.currentTimeMillis();
14. print("MySql批量插入10万条记录",a,b,point);
15. } **catch** (Exception ex) {
16. ex.printStackTrace();
17. }**finally**{
18. close(conn);
19. }

### **运行结果对比（插入10万条记录）**

#### **不加rewriteBatchedStatements的情况下进行对比**



这里通过一个点也可以看出来：

* Mysql批量与非批量性能是一样。
* oracle的JDBC实现的批量操作的性能十分优秀。

#### **给mySQL加上rewriteBatchedStatements的情况下进行对比**

**[java]** [view plain](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "view plain) [copy](http://blog.csdn.net/lifetragedy/article/details/52806186" \o "copy)

1. String connectionUrl="jdbc:mysql://192.168.1.100:3306/test?rewriteBatchedStatements=true" ;

笔者在使用mysql jdbc驱动5.1.13版的基础上 加了rewriteBatchedStatements参数， 并配合使用addBatch方法，10万条记录的一次性插入速度提高到了1.6秒左右。  
以下是使用了rewriteBatchedStatements后再配合使用addBatch的统计结果，供对比：

