**第十六章 LNMP调优**

**LNMP调优**

**LNMP的调优着重体现在Nginx服务器上的调优，Linux内核调优有专门的课程为大家讲解，Mysql和PHP的调优方式和LAMP的中的调试方式雷同！**

**16.1.1 Nginx编译前的优化**

**[root@xuegod63 ~]# tar zxvf nginx-1.12.2.tar.gz**

**[root@xuegod63 local]# cd nginx-1.12.2/**

**编译前的优化主要是用来修改程序名等等，例如：**

**[root@xuegod63 nginx-1.12.2]# curl -I <http://www.baidu.com>**

**……**

**Server: BWS/1.1**

**……**

**[root@xuegod63 nginx-1.12.2]# curl -I <http://www.sina.com.cn>**

**……**

**Server: nginx**

**……**

**[root@xuegod63 nginx-1.12.2]# curl -I <http://www.xuegod.cn>**

**HTTP/1.1 200 OK**

**Server: nginx/1.6.2 #我们目标是将nginx更改名字**

**Content-Type: text/html; charset=UTF-8**

**Connection: keep-alive**

**X-Powered-By: PHP/5.4.45**

**Set-Cookie: PHPSESSID=smm0i6u4f9v7bj0gove79ja1g7; path=/**

**Cache-Control: no-cache**

**Date: Mon, 07 Mar 2016 06:09:11 GMT**

**[root@xuegod63 nginx-1.12.2]# vim src/core/nginx.h //目的更改源码隐藏软件名称和版本号**

**改：**

**13 #define NGINX\_VERSION "1.12.2" #此行修改的是你想要的版本号**

**14 #define NGINX\_VER "nginx/" NGINX\_VERSION #此行修改的是你想修改的软件**

**名称**

**为：**

**13 #define NGINX\_VERSION "8.8.2"**

**14 #define NGINX\_VER "XWS/" NGINX\_VERSION**

**[root@xuegod63 nginx-1.12.2]# vim src/http/ngx\_http\_header\_filter\_module.c**

**改：49 static u\_char ngx\_http\_server\_string[] = "Server: nginx" CRLF; //修改HTTP头信息中的connection字段，防止回显具体版本号**

**为：49 static u\_char ngx\_http\_server\_string[] = "Server: XWS" CRLF;**

**拓展：通用http头域**

**通用头域包含请求和响应消息都支持的头域，通用头域包含Cache-Control（缓存控制）、 Connection（连接）、Date（日期）、Pragma（短语）、Transfer-Encoding（传输编码）、Upgrade（升级）、Via。对通用头域的扩展要求通讯双方都支持此扩展，如果存在不支持的通用头域，一般将会作为实体头域处理。那么也就是说有部分设备，或者是软件，能获取到connection，部分不能，要隐藏就要彻底！**

**[root@xuegod63 nginx-1.12.2]# vim src/http/ngx\_http\_special\_response.c**

**//这个文件定义了http错误码的返回，有时候我们页面程序出现错误，Nginx会代我们返回相应的错误代码，回显的时候，会带上nginx和版本号，我们把他隐藏起来**

**改：22 "<hr><center>" NGINX\_VER "</center>" CRLF #这里 就不需要修改了，以前老版本1.10需要修改，这里新版本中直接调用的是一个之前定义的" #define NGINX\_VER "XWS/" NGINX\_VERSION " 这里不需要修改**

**为：22 "<hr><center>" XWS "</center>" CRLF #老版本改这里**

## **16.1.2 Nginx正式安装**

**[root@xuegod63 nginx-1.12.2]# yum install gcc gcc-c++ autoconf automake zlib zlib-devel openssl openssl-devel pcre pcre-devel -y //安装依赖包**

**[root@xuegod63 nginx-1.12.2]# ./configure --prefix=/usr/local/nginx --with-http\_dav\_module --with-http\_stub\_status\_module --with-http\_addition\_module --with-http\_sub\_module --with-http\_flv\_module --with-http\_mp4\_module --with-pcre**

**--with-http\_dav\_module #启用支持（增加PUT,DELETE,MKCOL：创建集合，COPY和MOVE方法）**

**默认关闭，需要编译开启**

**--with-http\_stub\_status\_module #启用支持（获取Nginx上次启动以来的工作状态）**

**--with-http\_addition\_module #启用支持（作为一个输出过滤器，支持不完全缓冲，分部分相应请求）**

**--with-http\_sub\_module #启用支持（允许一些其他文本替换Nginx相应中的一些文本）**

**--with-http\_flv\_module #启用支持（提供支持flv视频文件支持）**

**--with-http\_mp4\_module #启用支持（提供支持mp4视频文件支持，提供伪流媒体服务端支持）**

**--with-pcre #需要注意，这里指的是源码,用#./configure --help |grep pcre查看帮助**

**[root@xuegod63 nginx-1.12.2]# make -j 4 && make install**

### **启动nginx**

**[root@xuegod63 ~]# cd /usr/local/nginx/ #查看nginx启动前目录**

**[root@xuegod63 nginx]# ls**

**conf html logs sbin**

**[root@xuegod63 nginx]# ./sbin/nginx**

**[root@xuegod63 nginx]# ls #启动后，会多一些临时文件**

**client\_body\_temp fastcgi\_temp logs sbin uwsgi\_temp**

**conf html proxy\_temp scgi\_temp**

**[root@xuegod63 nginx-1.12.2]# /usr/local/nginx/sbin/nginx**

**[root@xuegod63 nginx-1.12.2]# netstat -antup | grep nginx**

**tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 52553/nginx**

### **测试是否隐藏了版本和软件名**

**[root@xuegod63 nginx-1.12.2]# curl -I 192.168.1.63**

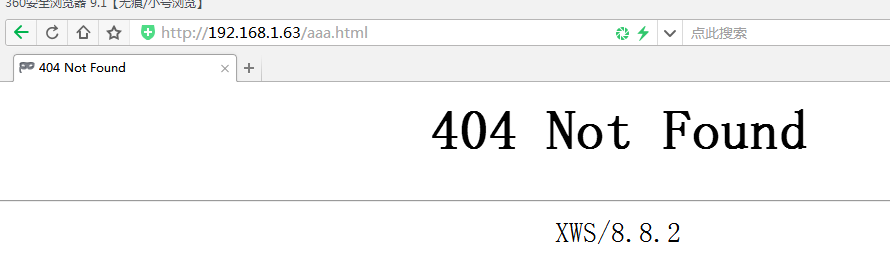
**HTTP/1.1 200 OK**

**Server: XWS/8.8.2**

**错误代码测试（尽量使用firefox或者类360流览器）**

**[root@xuegod63 nginx-1.12.2]# iptables -F #防火墙清空一下**

**http://192.168.1.63/aaa.html**

****

## **Nginx运行用户**

**[root@xuegod63~]# useradd -M -s /sbin/nologin nginx //修改nginx默认运行用户**

**[root@xuegod63 ~]# ps -aux | grep nginx //默认是nobody用户**

**nobody 52554 0.0 0.1 22660 1568 ? S 14:39 0:00 nginx: worker process**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**改：#user nobody;**

**为：user nginx;**

**[root@xuegod63 ~]# /usr/local/nginx/sbin/nginx -s reload**

**[root@xuegod63 ~]# ps -aux | grep nginx**

**nginx 52555 0.0 0.1 22660 1568 ? S 14:39 0:00 nginx: worker process**

**在这里我们还可以看到在查看的时候，work进程是nginx用户了，但是master进程还是root**

**其中，master是监控进程，也叫主进程，work是工作进程.**

**所以我们可以master监控进程使用root，可以是降级使用普通用户，如果都是用普用户，注意编译安装的时候，是用普通用户执行，sudo方式操作！可以直接理解为master是管理员，work进程才是为用户提供服务的！**

**Nginx运行进程个数**

**Nginx运行进程个数一般我们设置CPU的核心或者核心数x2，如果你不了解，top命令之后按1也可以看出来（一般直接追到线程即可）。**

**我是4核cpu，这里设为4**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**改： 4 worker\_processes 4;**

**为：4 worker\_processes 4;**

**[root@xuegod63 ~]# /usr/local/nginx/sbin/nginx -s reload**

**[root@xuegod63 ~]# ps -axu | grep nginx**

**[root@xuegod63 nginx-1.12.2]# ps -axu | grep nginx #运行4个worker进程**

**[root@xuegod63 ~]# ps -axu | grep nginx**

**root 6483 0.0 0.0 20580 1400 ? Ss 21:05 0:00 nginx: master process ./sbin/nginx**

**root 6648 0.0 0.2 151796 5372 pts/2 S+ 21:12 0:00 vim conf/nginx.conf**

**nginx 6871 0.0 0.0 23088 1512 ? S 21:22 0:00 nginx: worker process**

**nginx 6872 0.0 0.0 23088 1512 ? S 21:22 0:00 nginx: worker process**

**nginx 6873 0.0 0.0 23088 1512 ? S 21:22 0:00 nginx: worker process**

**nginx 6874 0.0 0.0 23088 1512 ? S 21:22 0:00 nginx: worker process**

**16.1.3 Nginx运行CPU亲和力**

**这个要根据你的CPU线程数配置**

**比如4核4线程配置**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**4 worker\_processes 4;**

**5 worker\_cpu\_affinity 0001 0010 0100 1000;**

**比如8核8线程配置**

**4 worker\_processes 8;**

**5 worker\_cpu\_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;**

**那么如果我是4线程的CPU，我只想跑两个进程呢？**

**worker\_processes 2;**

**worker\_cpu\_affinity 0101 1010;**

**注：表示第一个进程在第一个和第三个cpu上运行，第二个和第四个cpu上运行，两个进程分别在这两个组合上轮询！**

**worker\_processes最多开启8个，8个以上性能提升不会再提升了，而且稳定性变得更低，所以8个进程够用了。**

## **Nginx最多可以打开文件数**

**7 worker\_rlimit\_nofile 102400;**

**这个指令是指当一个nginx进程打开的最多文件描述符数目，理论值应该是最多打开文件数（ulimit -n）与nginx进程数相除，但是nginx分配请求并不是那么均匀，所以最好与ulimit -n的值保持一致。**

## **Nginx事件处理模型**

**events {**

**use epoll;**

**worker\_connections 1024;**

**}**

**扩展：**

**select，poll，epoll都是IO多路复用的机制。I/O多路复用就通过一种机制，可以监视多个描述符，一旦某个描述符就绪（一般是读就绪或者写就绪），能够通知程序进行相应的读写操作。**

**Epoll 在Linux2.6内核中正式引入，和select相似，其实都I/O多路复用技术。**

**epoll优势：**

**1、Epoll没有最大并发连接的限制，上限是最大可以打开文件的数目，这个数字一般远大于2048, 一般来说这个数目和系统内存关系很大，具体数目可以cat /proc/sys/fs/file-max查看。**

**[root@xuegod63 nginx-1.12.2]# cat /proc/sys/fs/file-max**

**95094**

**2、 效率提升，Epoll最大的优点就在于它只管你“活跃”的连接，而跟连接总数无关，因此在实际的网络环境中，Epoll的效率就会远远高于select和poll。**

**3、 内存拷贝，Epoll在这点上使用了“共享内存”，这个内存拷贝也省略了**

**16.1.4 单个进程允许客户端最大并发连接数**

**17 worker\_connections 102400; #改第17行**

**这个数值一般根据服务器性能和内存来制定，也就是单个进程最大连接数，实际最大并发值就是work进程数乘以这个数。**

**如何设置，可以根据设置一个进程启动所占内存，top -u nginx，但是实际我们填入一个102400，足够了，这些都算并发值，一个网站的并发达到这么大的数量，也算一个大站了！**

**例：**

**top -u nginx**

**PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND**

**47221 nginx 20 0 23088 1764 832 S 0.0 0.1 0:01.06 nginx**

**47223 nginx 20 0 23088 1772 840 S 0.0 0.1 0:00.77 nginx**

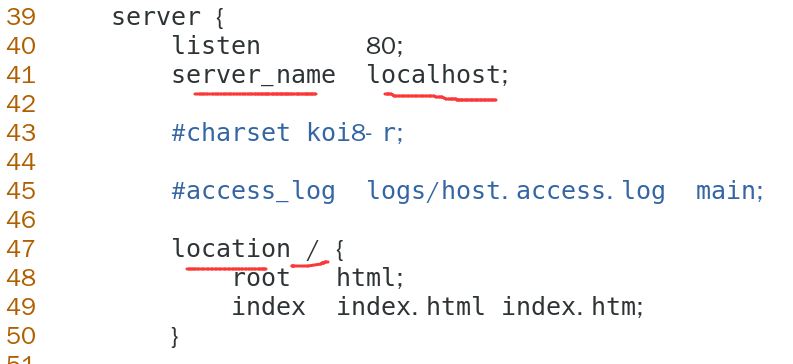
**47225 nginx 20 0 23088 1768 836 S 0.0 0.1 0:00.64 nginx**

**47226 nginx 20 0 23088 1764 832 S 0.0 0.1 0:00.87 nginx**

**刚启动是1.7M左右**

## **ServerName匹配和location**

**对应需要修改的配置文件的位置：**

****

## **ServerName匹配**

**1：精确匹配：[www.aa.com](http://www.aa.com)**

**2：左侧通配符匹配：\*.aa.com**

**3：右侧通配符匹配：[www.\*](http://www.*)**

**4：正则表达式：~ ^.\*\.aa\.com$**

**5: default\_server**

**6、服务IP地址**

## **location**

**= 绝对匹配**

**^~：URL前半部分匹配，不检查正则**

**~：正则匹配，区分大小写**

**~\*“正则匹配”不区分大小写**

**\转义**

**\* 配置任意个任意字符**

**$ 以什么结尾**

**例：**

**[root@xuegod63 sbin]# grep bash$ /etc/passwd**

**root:x:0:0:root:/root:/bin/bash**

**=，^~,~,~\***

## **16.1.5 开启高效传输模式**

**Include mime.types; 媒体类型**

**default\_type application/octet-stream; 默认媒体类型 足够**

**31 sendfile on； 开启高效文件传输模式，sendfile指令指定nginx是否调用sendfile函数来输出文件，对于普通应用设为 on，如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络I/O处理速度，降低系统的负载。注意：如果图片显示不正常把这个改成off。**

**32 tcp\_nopush on； 必须在sendfile开启模式才有效，防止网络阻塞，积极的减少网络报文段的数量**

## **16.1.6 连接超时时间**

**主要目的是保护服务器资源，CPU，内存，控制连接数，因为建立连接也是需要消耗资源的，TCP的三次握手四次挥手等，我们一般断掉的是那些建立连接但是不做事儿，也就是我建立了链接开始，但是后续的握手过程没有进行，那么我们的链接处于等待状态的，全部断掉！**

**同时我们也希望php建议短链接，消耗资源少**

**35 keepalive\_timeout 65;**

**tcp\_nodelay on;**

**client\_header\_timeout 15;**

**client\_body\_timeout 15;**

**send\_timeout 15;**

**keepalived\_timeout 客户端连接保持会话超时时间，超过这个时间，服务器断开这个链接**

**tcp\_nodelay；也是防止网络阻塞，不过要包涵在keepalived参数才有效**

**client\_header\_timeout 客户端请求头读取超时时间，如果超过设个时间没有发送任何数据，nginx将返回request time out的错误**

**client\_body\_timeout 客户端求主体超时时间，超过这个时间没有发送任何数据，和上面一样的错误提示**

**send\_timeout 响应客户端超时时间，这个超时时间仅限于两个活动之间的时间，如果超过这个时间，客户端没有任何活动，nginx关闭连接**

## **文件上传大小限制**

**我们知道PHP可以修改上传文件大小限制，nginx也可以修改**

**http {**

**……**

1. **client\_max\_body\_size 10m;**

## **16.2.1 Fastcgi调优**

**Nginx没有配置factcgi，你使用nginx是一个失败的方法**

**配置之前。了解几个概念：**

**Cache：写入缓存区**

**Buffer：读取缓存区**

**Fastcgi是静态服务和动态服务的一个接口**

**fastcgi\_connect\_timeout 300; #指定链接到后端FastCGI的超时时间。**

**fastcgi\_send\_timeout 300; #向FastCGI传送请求的超时时间，这个值是指已经完成两次握手后向FastCGI传送请求的超时时间。**

**fastcgi\_read\_timeout 300; #指定接收FastCGI应答的超时时间，这个值是指已经完成两次握手后接收FastCGI应答的超时时间。**

**fastcgi\_buffer\_size 64k; #指定读取FastCGI应答第一部分需要用多大的缓冲区，这个值表示将使用1个64KB的缓冲区读取应答的第一部分（应答头），可以设置为fastcgi\_buffers选项指定的缓冲区大小。**

**fastcgi\_buffers 4 64k; #指定本地需要用多少和多大的缓冲区来缓冲FastCGI的应答请求，如果一个php脚本所产生的页面大小为256KB，那么会分配4个64KB的缓冲区来缓存，如果页面大小大于256KB，那么大于256KB的部分会缓存到fastcgi\_temp指定的路径中，但是这并不是好方法，因为内存中的数据处理速度要快于磁盘。一般这个值应该为站点中php脚本所产生的页面大小的中间值，如果站点大部分脚本所产生的页面大小为256KB，那么可以把这个值设置为“8 16K”、“4 64k”等。**

**fastcgi\_busy\_buffers\_size 128k; #建议设置为fastcgi\_buffer的两倍，繁忙时候的buffer**

**fastcgi\_temp\_file\_write\_size 128k; #在写入fastcgi\_temp\_path时将用多大的数据库，默认值是fastcgi\_buffers的两倍，设置上述数值设置小时若负载上来时可能报502Bad Gateway**

**fastcgi\_cache gnix; #表示开启FastCGI缓存并为其指定一个名称。开启缓存非常有用，可以有效降低CPU的负载，并且防止502的错误发生，但是开启缓存也可能会引起其他问题，要很据具体情况选择**

**fastcgi\_cache\_valid 200 302 1h; #用来指定应答代码的缓存时间，实例中的值表示将200和302应答缓存一小时，要和fastcgi\_cache配合使用**

**fastcgi\_cache\_valid 301 1d; #将301应答缓存一天**

**fastcgi\_cache\_valid any 1m; #将其他应答缓存为1分钟**

**fastcgi\_cache\_min\_uses 1; #请求的数量**

**fastcgi\_cache\_path #定义缓存的路径**

**修改nginx.conf配置文件，在http标签中添加如下：**

**fastcgi\_connect\_timeout 300;**

**fastcgi\_send\_timeout 300;**

**fastcgi\_read\_timeout 300;**

**fastcgi\_buffer\_size 64k;**

**fastcgi\_buffers 4 64k;**

**fastcgi\_busy\_buffers\_size 128k;**

**fastcgi\_temp\_file\_write\_size 128k;**

**#fastcgi\_temp\_path /data/ngx\_fcgi\_tmp;**

**fastcgi\_cache\_path /data/ngx\_fcgi\_cache levels=2:2 #缓存路径，levels目录层次2级**

**keys\_zone=ngx\_fcgi\_cache:512m #定义了一个存储区域名字，缓存大小**

**inactive=1d max\_size=40g; #不活动的数据在缓存中多长时间，目录总大小**

**在server location标签添加如下：**

**location ~ .\*\.(php|php5)?$**

**{**

**fastcgi\_pass 127.0.0.1:9000;**

**fastcgi\_index index.php;**

**include fastcgi.conf;**

**fastcgi\_cache ngx\_fcgi\_cache;**

**fastcgi\_cache\_valid 200 302 1h;**

**fastcgi\_cache\_valid 301 1d;**

**fastcgi\_cache\_valid any 1m;**

**fastcgi\_cache\_min\_uses 1;**

**fastcgi\_cache\_use\_stale error timeout invalid\_header http\_500;**

**fastcgi\_cache\_key http://$host$request\_uri;**

**}**

**fastcgi cache资料：**

**官方文档：**

**<http://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#fastcgi_cache>**

## **16.3.1 gzip调优**

**使用gzip压缩功能，可能为我们节约带宽，加快传输速度，有更好的体验，也为我们节约成本，所以说这是一个重点**

**Nginx启用压缩功能需要你来ngx\_http\_gzip\_module模块，apache使用的是mod\_deflate**

**一般我们需要压缩的内容有：文本，js，html，css，对于图片，视频，flash什么的不压缩，同时也要注意，我们使用gzip的功能是需要消耗CPU的！**

**gzip on; #开启压缩功能**

**gzip\_min\_length 1k; #设置允许压缩的页面最小字节数，页面字节数从header头的Content-Length（内容长度）中获取，默认值是0，不管页面多大都进行压缩，建议设置成大于1K，如果小与1K可能会越压越大。**

**gzip\_buffers 4 32k; #压缩缓冲区大小，表示申请4个单位为32K的内存作为压缩结果流缓存，默认值是申请与原始数据大小相同的内存空间来存储gzip压缩结果。**

**gzip\_http\_version 1.1; #压缩版本（默认1.1，前端为squid2.5时使用1.0）用于设置识别HTTP协议版本，默认是1.1，目前大部分浏览器已经支持GZIP解压，使用默认即可**

**gzip\_comp\_level 9; #压缩比例，用来指定GZIP压缩比，1压缩比最小，处理速度最快，9压缩比最大，传输速度快，但是处理慢，也比较消耗CPU资源。**

**gzip\_types text/css text/xml application/javascript; #用来指定压缩的类型，‘text/html’类型总是会被压缩。**

**gzip\_vary on; #vary header支持，该选项可以让前端的缓存服务器缓存经过GZIP压缩的页面，例如用Squid缓存经过nginx压缩的数据**

**那么配置压缩的过程中，会有一下参数**

**gzip on;**

**gzip\_min\_length 1k;**

**gzip\_buffers 4 32k;**

**gzip\_http\_version 1.1;**

**gzip\_comp\_level 9;**

**gzip\_types text/css text/xml application/javascript;**

**gzip\_vary on;**

## **16.4.1 expires缓存调优**

**缓存，主要针对于图片，css，js等元素更改机会比较少的情况下使用，特别是图片，占用带宽大，我们完全可以设置图片在浏览器本地缓存365d，css，js，html可以缓存个10来天，这样用户第一次打开加载慢一点，第二次，就非常快了！缓存的时候，我们需要将需要缓存的拓展名列出来！**

**xpires缓存配置在server字段里面**

**location ~ .\*\.(gif|jpg|jpeg|png|bmp|swf)$**

**{**

**expires 3650d;**

**}**

**location ~ .\*\.(js|css)?$**

**{**

**expires 30d;**

**}**

**同时也可以对目录及其进行判断：**

**location ~ ^/(images|javascript|js|css|flash|media|static)/ {**

**expires 360d;**

**}**

**location ~(robots.txt) {**

**expires 7d;**

**break;**

**}**

**expire功能优点**

**（1）expires可以降低网站购买的带宽，节约成本**

**（2）同时提升用户访问体验**

**（3）减轻服务的压力，节约服务器成本，甚至可以节约人力成本，是web服务非常重要的功能。**

**expire功能缺点：**

**被缓存的页面或数据更新了，用户看到的可能还是旧的内容，反而影响用户体验。**

**解决办法：**

**第一个 缩短缓存时间，例如：1天，不彻底，除非更新频率大于1天**

**第二个 对缓存的对象改名**

**a.图片，附件一般不会被用户修改，如果用户修改了，实际上也是更改文件名重新传了而已**

**b.网站升级对于js，css元素，一般可以改名，把css，js，推送到CDN。**

**网站不希望被缓存的内容**

**1）广告图片**

**2）网站流量统计工具**

**3）更新频繁的文件（google的logo）**

## **16.5.1 日志切割优化**

**[root@xuegod63 ~]# cd /usr/local/nginx/logs/**

**日志优化的目的，是为了一天日志一压缩，按天存放，超过10天的删除**

**[root@xuegod63 logs]# vim cut\_nginx\_log.sh //每天日志分割脚本**

**（tar tvf 2017-011-08.tar.bz2 显示文件内容）**

**#!/bin/bash**

**date=$(date +%F -d -1day)**

**cd /usr/local/nginx/logs**

**if [ ! -d cut ] ; then**

**mkdir cut**

**fi**

**mv access.log cut/access\_$(date +%F -d -1day).log**

**mv error.log cut/error\_$(date +%F -d -1day).log**

**/usr/local/nginx/sbin/nginx -s reload**

**tar -jcvf cut/$date.tar.bz2 cut/\***

**rm -rf cut/access\* && rm -rf cut/error\***

**cat >>/var/spool/cron/root<<eof**

**00 00 \* \* \* /bin/sh /usr/local/nginx/logs/cut\_nginx\_log.sh >/dev/null 2>&1**

**eof**

**find -type f -mtime +10 | xargs rm -rf**

**健康检查的日志，不用输入到log中，因为这些日志没有意义，我们分析的话只需要分析访问日志，看看一些页面链接，如200，301，404的状态吗，在SEO中很重要，而且我们统计PV是页面计算，这些都没有意义，反而消耗了磁盘IO，降低了服务器性能，我们可以屏蔽这些如图片，js，css这些不宜变化的内容**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**location ~ .\*\.(js|jpg|jpeg|JPG|JPEG|css|bmp|gif|GIF)$ {**

**access\_log off;**

**}**

**日志目录权限优化**

**[root@xuegod63 ~]# chown -R root.root /xuegod/logs/**

**[root@xuegod63 ~]# chmod -R 700 /xuegod/logs/**

## **日志格式优化**

**#vim /usr/local/nginx/conf/nginx.conf**

**log\_format access ‘$remote\_addr – $remote\_user [$time\_local] “$request” ‘‘$status $body\_bytes\_sent “$http\_referer” ‘‘”$http\_user\_agent” $http\_x\_forwarded\_for’;**

**其中，各个字段的含义如下：**

**1.$remote\_addr 与$http\_x\_forwarded\_for 用以记录客户端的ip地址；**

**2.$remote\_user ：用来记录客户端用户名称；**

**3.$time\_local ： 用来记录访问时间与时区；**

**4.$request ： 用来记录请求的url与http协议；**

**5.$status ： 用来记录请求状态；成功是200，**

**6.$body\_bytes\_s ent ：记录发送给客户端文件主体内容大小；**

**7.$http\_referer ：用来记录从那个页面链接访问过来的；**

**8.$http\_user\_agent ：记录客户端浏览器的相关信息；**

## **16.6.1 目录文件访问控制**

**主要用在禁止目录下指定文件被访问，当然也可以禁止所有文件被访问！一般什么情况下用？比如是有存储共享，这些文件本来都只是一些下载资源文件，那么这些资源文件就不允许被执行，如sh,py,pl,php等等**

**例如：禁止访问images下面的php程序文件**

**location ~ ^/images/.\*\.(php|php5|.sh|.py|.pl)$ {**

**deny all;**

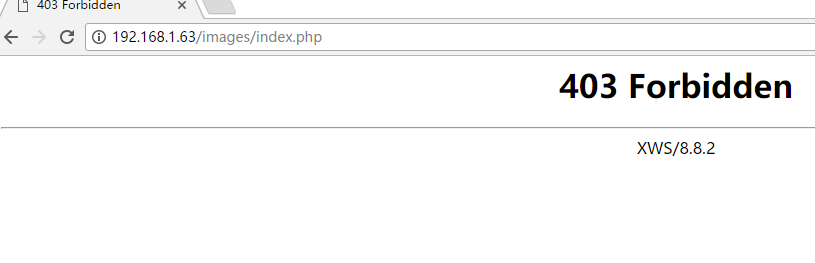
**}**

**[root@xuegod63 ~]# /usr/local/nginx/sbin/nginx -s reload**

**[root@xuegod63 ~]# mkdir /usr/local/nginx/html/images**

**[root@xuegod63 ~]# echo "<?php phpinfo(); ?>" > /usr/local/nginx/html/images/index.php**

**测试访问**



**多目录组合配置方法**

**location ~ ^/images/(attachment|avatar)/.\*\.(php|php5|.sh|.py|.py)$ {**

**deny all;**

**}**

**配置nginx禁止访问\*.txt文件**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf //server字段中**

**location ~\* \.(txt|doc)$ {**

**if ( -f $request\_filename) {**

**root /usr/local/nginx/html;**

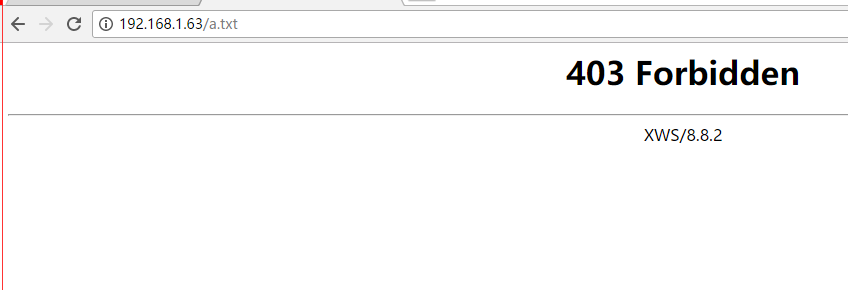
**break;**

**}**

**deny all;**

**}**

**[root@xuegod63 ~]# /usr/local/nginx/sbin/nginx -s reload**



**当然，可以重定向到某一个URL**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**location ~\* \.(txt|doc)$ {**

**if ( -f $request\_filename) {**

**root /usr/local/nginx/html;**

**rewrite ^/(.\*)$ http://www.baidu.com last;**

**break;**

**}**

**}**

**对目录进行限制的方法**

**[root@xuegod63 ~]# mkdir -p /usr/local/nginx/html/{xuegod,godxue}**

**[root@xuegod63 ~]# echo xuegod > /usr/local/nginx/html/xuegod/index.html**

**[root@xuegod63 ~]# echo god > /usr/local/nginx/html/godxue/index.html**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**location /xuegod/ { return 404 ; }**

**location /godxue/ { return 403 ; }**





**上面是直接给了反馈的状态码，也可以把他能够过匹配deny all方式做**

**上面是直接给了反馈的状态码，也可以哦他能够过匹配deny all方式做**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**location ~ ^/(xuegod)/ {**

**deny all;**

**}**

**[root@xuegod63 ~]# /usr/local/nginx/sbin/nginx -s reload**



## **16.7.1 来源访问控制**

**这个需要ngx\_http\_access\_module模块支持，不过，默认会安装**

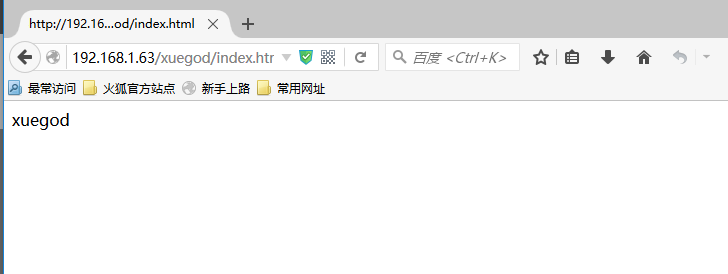
**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf //写法类似Apache**

**location ~ ^/(xuegod)/ {**

**allow 192.168.1.0/24;**

**deny all;**

**}**

****

**接着上面的实验，就可以访问了，下面是针对整个网站的写法，对/限制就OK**

**location / {**

**allow 192.168.1.0/24;**

**deny all;**

**}**

**当然可以写IP，可以写IP段，但是注意次序，上下匹配**

**同时，也可以通过if语句控制，给以友好的错误提示**

**if ( $remote\_addr = 192.168.1.38 ) {**

**return 404;**

**}**

****

## **16.8.1 IP和301优化**

**有时候，我们发现访问网站的时候，使用IP也是可以得，我们可以把这一层给屏蔽掉，让其直接反馈给403,也可以做跳转**

**跳转的做法：**

**server {**

**listen 80 default\_server;**

**server\_name localhost;**

**rewrite ^ http://www.baidu.com$request\_uri?;**

**}**

**403反馈的做法**

**server {**

**listen 80 default\_server;**

**server\_name localhost;**

**return 403；**

**}**

**301跳转的做法，如我们域名一般在解析的过程中，a.com一般会跳转到[www.a.com](http://www.a.com)**

**301跳转的做法，如我们域名一般在解析的过程中，a.com一般会跳转到[www.a.com](http://www.a.com)**

**server {**

**listen 80;**

**root /usr/share/nginx/html/;**

**server\_name www.a.com a.com;**

**if ($host = 'a.com' ) {**

**rewrite ^/(.\*)$ http://www.a.com/$1 permanent;**

**}**

## **错误页面的提示**

**对于自定义的错误页面，我们只需要将errorpage写入到配置文件**

**error\_page 404 /404.html;**

## **16.9.1 内部身份验证**

**[root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**location /xuegod/ {**

**auth\_basic "haha";**

**auth\_basic\_user\_file /usr/local/nginx/conf/passwd;**

**}**

**用户创建，如果没有htpasswd命令，需要手动安装httpd-tools程序包**

**[root@xuegod63 logs]# rpm -ivh /mnt/Packages/httpd-tools-2.4.6-67.el7.centos.x86\_64.rpm**

**或：使用yum进行安装**

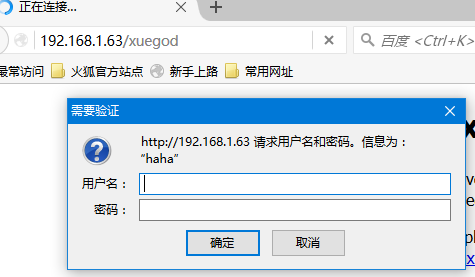
**[root@xuegod63 logs]# yum -y install httpd-tools**

**[root@xuegod63 ~]# htpasswd -cb /usr/local/nginx/conf/passwd aaa 123**

**[root@xuegod63 ~]# chmod 400 /usr/local/nginx/conf/passwd**

**[root@xuegod63 ~]# chown nginx /usr/local/nginx/conf/passwd**

**[root@xuegod63 ~]# /usr/local/nginx/sbin/nginx -s reload**

****

## **16.10.1 防止DDOS攻击**

**通过使用limit\_conn\_zone进行控制单个IP或者域名的访问次数**

**root@xuegod63 ~]# vim /usr/local/nginx/conf/nginx.conf**

**http字段中配置**

**limit\_conn\_zone $binary\_remote\_addr zone=addr:10m;**

**server的location字段配置**

**location / {**

**root html;**

**limit\_conn addr 1;**

**[root@xuegod64 ~]# ab -c 2 -t 1 http://192.168.1.63/xuegod/index.html**

****

**= 开头表示精确匹配**

**^~ 开头表示uri以某个常规字符串开头，理解为匹配 url路径即可。nginx不对url做编码，因此请求为/static/20%/aa，可以被规则^~ /static/ /aa匹配到（注意是空格）。**

**~ 开头表示区分大小写的正则匹配**

**~\* 开头表示不区分大小写的正则匹配**

**!~和!~\*分别为区分大小写不匹配及不区分大小写不匹配 的正则**

**/ 通用匹配，任何请求都会匹配到。**

**多个location配置的情况下匹配顺序为（参考资料而来，还未实际验证，试试就知道了，不必拘泥，仅供参考）:**

**首先匹配 =，其次匹配^~, 其次是按文件中顺序的正则匹配，最后是交给 / 通用匹配。当有匹配成功时候，停止匹配，按当前匹配规则处理请求。**