目录

[1. Nginx Web深入优化 1](#_Toc441564062)

[1.1 Nginx 基本优化 1](#_Toc441564063)

[1.1.1 调整参数隐藏Nginx版本号信息 1](#_Toc441564064)

[1.1.2 更改源码隐藏Nginx软件名及版本号 3](#_Toc441564065)

[1.1.3 更改nginx服务的默认用户 5](#_Toc441564066)

[1.2 根据参数优化Nginx服务性能 6](#_Toc441564067)

[1.2.1 优化nginx服务的worker进程个数 6](#_Toc441564068)

[1.2.2 优化绑定不同的nginx进程到不同的cpu上 8](#_Toc441564069)

[1.2.3 nginx事件处理模型优化 9](#_Toc441564070)

[1.2.4 调整nginx单个进程允许的客户端最大连接数 10](#_Toc441564071)

[1.2.5 配置nginx worker进程最大打开文件数 11](#_Toc441564072)

[1.2.6 优化服务器域名的hash表大小 11](#_Toc441564073)

[1.2.7 开启高效文件传输模式 13](#_Toc441564074)

[1.2.8 优化Nginx连接参数调整连接超时时间 13](#_Toc441564075)

[1.2.9 上传文件大小（http Request body size）的限制（动态应用） 17](#_Toc441564076)

[1.2.10 fastcgi 相关参数调优（配合PHP引擎动态服务） 17](#_Toc441564077)

[1.2.11 配置nginx gzip压缩实现性能优化 21](#_Toc441564078)

[1.2.12 Nginx expires缓存实现性能优化 24](#_Toc441564079)

[1.3 nginx日志相关优化与安全 26](#_Toc441564080)

[1.3.1 编写脚本实现nginx access 日志轮询 26](#_Toc441564081)

[1.3.2 不记录不需要的访问日志 27](#_Toc441564082)

[1.3.3 访问日志的权限设置 27](#_Toc441564083)

[1.4 nginx站点目录及文件URL访问控制 28](#_Toc441564084)

[1.4.1 根据扩展名闲置程序和文件访问 28](#_Toc441564085)

[1.4.2 禁止访问指定目录下所有文件和目录 28](#_Toc441564086)

[1.4.3 限制网站来源IP访问 29](#_Toc441564087)

[1.4.4 配置nginx禁止非法域名解析访问企业网站 30](#_Toc441564088)

[1.5 nginx图片及目录防盗链解决方案 30](#_Toc441564089)

[1.6 nginx错误页面的优雅显示 32](#_Toc441564090)

# 1. Nginx Web深入优化

## 1.1 Nginx 基本优化

### 1.1.1 调整参数隐藏Nginx版本号信息

一般来说，软件的漏洞都和版本有关，这个很像汽车的缺陷，同一批次的要有问题都有问题，别的批次可能就都是好的。因此，我们应尽量隐藏或消除web服务队访问的用户显示各类敏感信息（例如：web软件名称及版本号等信息），这样恶意的用户就很难猜到他攻击的服务器是否是有特定漏洞的软件，或者是否为有对应漏洞的某一特定版本，从而加强web服务的安全性，这在小说中，就相当于隐身术，你隐身了，对手就几乎达不到你了。

想要隐身，首先要了解所使用软件的版本号，对于LInux客户端，可通过命令行查看Nginx版本号，最简单的办法就是在Linux客户端系统命令行执行curl命令

[root@Web01 ~]# curl -I 127.0.0.1

HTTP/1.1 200 OK

Server: nginx/1.6.3 ##暴漏了Web版本号以及软件名

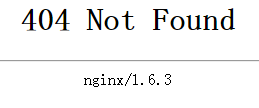
Date: Wed, 20 Jan 2016 06:55:53 GMT

Content-Type: text/html; charset=utf-8

Connection: keep-alive

X-Powered-By: PHP/5.3.27

在Windows客户端哈桑通过浏览器访问WEB服务找不到页面默认报错时如下图：



以上虽然是不同的客户端但是都是获得了Nginx软件名，而且查到了Nginx的版本号，这使得WEB服务器存在风险，因此，应隐藏掉这些敏感信息或用一个其他名字代替，例如百度网站的WEB软件的更名做法

[root@Web01 ~]# curl -I www.baidu.com

HTTP/1.1 200 OK

Date: Wed, 20 Jan 2016 07:01:00 GMT

Server: BWS/1.1 ##将WEB服务软件更名为BWS

事实上，还可以通过配置文件加参数隐藏nginx版本号。

编辑nginx.conf配置文件增加参数实现隐藏Nginx版本号的方式如下：

[root@Web01 ~]# cd /application/nginx/conf/

[root@Web01 conf]# vim nginx.conf

http {

include mime.types;

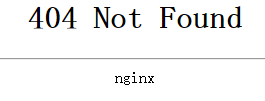
default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

server\_tokens off; ###在http标签下添加这一行

然后再Windows端浏览网站不存在的页面



[root@Web01 conf]# curl -I 127.0.0.1

HTTP/1.1 200 OK

Server: nginx

Date: Wed, 20 Jan 2016 07:13:11 GMT

Content-Type: text/html; charset=utf-8

Connection: keep-alive

X-Powered-By: PHP/5.3.27

提示：

1. 此参数的作用是控制http response header内的web服务版本信息的显示，以及错误信息中web服务版本信息的显示。

2. 此参数的防止位置为http标签内。

server\_tokens参数的官方说明如下：

syntax : server\_tokens on|off; ##此行为参数语法，on为开启状态，off为关闭状态。

default: server\_tokens on; #此行意思是不配置该参数，软件默认情况的结果。

context： http,server,location #此行为server\_tokens参数可以防止的位置

参数作用：激活或禁止nginx的版本信息显示在报错信息以及server的相应首部位置。

配置完保存，重新加载配置文件，在此通过curl查看结果为：

[root@Web01 conf]# curl -I 127.0.0.1

HTTP/1.1 200 OK

Server: nginx ##版本号已消失

Date: Wed, 20 Jan 2016 07:13:11 GMT

Content-Type: text/html; charset=utf-8

Connection: keep-alive

X-Powered-By: PHP/5.3.27

### 1.1.2 更改源码隐藏Nginx软件名及版本号

隐藏了Nginx版本号后，更近一步，可以通过一些手段把Web服务软件的名称也给隐藏，或者是为了迷惑黑客，但软件名字的隐藏修改，一般情况下不会有配置参数和入口，nginx也不例外，可能是由于商业及品牌展示等原因，软件提供商不希望使用者把软件名字都隐藏了。因此，此处需要更改nginx源代码才能解决，具体的解决方法如下。

第一步是依次修改3个Nginx源码文件

1） 修改的第一个文件为/nginx-1.6.3/src/core/nginx.h

[root@Web01 nginx-1.6.3]# cd /home/oldboy/tools/

[root@Web01 tools]# cd nginx-1.6.3

[root@Web01 nginx-1.6.3]# sed -n '13,17p' src/core/nginx.h

#define NGINX\_VERSION "2.4.2" ###修改版本号

#define NGINX\_VER "nginx/" NGINX\_VERSION

## 将nginx修改为想要修改的软件名称，如OWS

#define NGINX\_VAR "GUYUN" ###将nginx修改为想要修改的软件名称，如OWS

#define NGX\_OLDPID\_EXT ".oldbin"

修改后的结果如下：

[root@lnmp nginx-1.6.3]# sed -n '13,17p' src/core/nginx.h

#define NGINX\_VERSION "1.1.1" #已修改为想要显示的版本号

#define NGINX\_VER "nginx/" NGINX\_VERSION

#define NGINX\_VAR "LL" ##已修改为想要显示的名字

#define NGX\_OLDPID\_EXT ".oldbin"

2） 修改nginx-1.6.3/src/http/ngx\_http\_header\_filter\_module.c 的第49行，需要修改的字符创为：

[root@Web01 tools]# vim nginx-1.6.3/src/http/ngx\_http\_header\_filter\_module.c

static char ngx\_http\_server\_string[] = "Server: GUYUN" CRLF;

static char ngx\_http\_server\_full\_string[] = "Server: " NGINX

\_VER CRLF;

3） 修改nginx-1.6.3/src/http/ngx\_http\_special\_response.c，控制对面页面报错展示敏感信息，将修改前的信息打印ngx\_http\_special\_response.c第20~30行

[root@Web01 tools]# vim nginx-1.6.3/src/http/ngx\_http\_special\_response.c

static u\_char ngx\_http\_error\_full\_tail[] =

"<hr><center>" NGINX\_VER" (qq:996126002) </center>" CRLF

"</body>" CRLF

"</html>" CRLF

;

static u\_char ngx\_http\_error\_tail[] =

"<hr><center>GUYUN</center>" CRLF

"</body>" CRLF

"</html>" CRLF

[root@Web01 tools]# /application/nginx/sbin/nginx -V

nginx version: nginx/1.6.3

built by gcc 4.4.7 20120313 (Red Hat 4.4.7-16) (GCC)

TLS SNI support enabled

configure arguments: --prefix=/application/nginx-1.6.3 --user=nginx --group=nginx --with-http\_ssl\_module --with-http\_stub\_status\_module

[root@Web01 tools]# pkill nginx

[root@Web01 tools]# cd nginx-1.6.3

[root@Web01nginx-1.6.3]# ./configure --prefix=/application/nginx-1.6.3 --user=nginx --group=nginx --with-http\_ssl\_module --with-http\_stub\_status\_module

[root@Web01 nginx-1.6.3]# make

[root@Web01 nginx-1.6.3]# make install

[root@Web01 nginx-1.6.3]# /application/nginx/sbin/nginx

[root@Web01 nginx-1.6.3]# curl -I 127.0.0.1

HTTP/1.1 200 OK

Server: GUYUN

Date: Wed, 20 Jan 2016 07:53:09 GMT

Content-Type: text/html; charset=utf-8

Connection: keep-alive

X-Powered-By: PHP/5.3.27



**特别说明：**

提升网站安全，要从最简单、最短板、最低点入手解决问题，门打开着，窗户安装再结实的护栏也没有意义。

向有经验的人及优秀的网站公司学习。

学习看官方文档，根据一手资料去分析。

命令输出结果中含有需要过滤掉或者要保留的内容时，命令自身可能有参数直接实现。

### 1.1.3 更改nginx服务的默认用户

为了让Web服务更安全，我们要尽可能地改掉所有软件的配置，包括端口、用户等。

下面就更改nginx服务的默认用户：

查看nginx服务对应的默认用户

一般情况，nginx服务启动，使用的用户默认是nobody，查看默认配置文件，如下：

[root@Web01 conf]# grep '#user' nginx.conf.default

#user nobody;

为了防止黑客猜到这个web服务的用户，我们需要更改成特殊的用户名，例如：nginx或者特殊点的inca，但是这个用户必须是系统里事先存在的，下面以nginx用户为例进行说明。

1) 为nginx服务建立新用户。

为nginx服务建立新用户的操作过程如下：

[root@lnmp conf]# useradd nginx -s /sbin/nologin –M

##不需要有系统登录权限，应当禁止其登录能力，相当于Apache里的用户一样的。

[root@lnmp conf]# id nginx ##检查用户

uid=888(nginx) gid=888(nginx) groups=888(nginx)

2） 配置nginx服务，让其使用刚建立的nginx用户

更改nginx服务默认使用的用户方法有两种：

1. 第一种为直接更改配置文件参数，将默认的#user nodboy;改为如下内容：

user nginx nginx;

如果注释或不设置上述参数，默认既是nobody用户，不推荐使用nobody用户名，最好采用一个普通用户。

2. 第二种方法为，直接在编译nginx的时候直接指定了用户和组

./configure --prefix=/application/nginx-1.6.3 --user=nginx --group=nginx --with-http\_ssl\_module --with-http\_stub\_status\_module

[root@Web01 conf]# ps -ef |grep nginx

root nginx: master process /application/nginx/sbin/nginx

nginx 9598 9597 0 15:52 ? 00:00:00 nginx: worker process

root 9631 1399 0 16:08 pts/1 00:00:00 grep nginx

3） 检查更改用户的效果

重新加载配置后，检查nginx服务进程的对应用户如下：

[root@lnmp conf]# ps -ef|grep nginx|grep -v grep

root 58191 1 0 21:16 ? 00:00:00 nginx: master process /application/nginx/sbin/nginx

## 1.2 根据参数优化Nginx服务性能

### 1.2.1 优化nginx服务的worker进程个数

在高并发、高访问量的web服务场景，需要事先启动好更多的nginx进程以保证快速响应并处理大量并发用户的请求。

这类似于开饭店，在营业前，需要事先招聘一定数量的服务员准备接待顾客，但这里就有一个问题，如果饭店对客流没有正确的预估，就会导致一些问题发生，例如：服务员人数招多了，但是客流很少，那么大家就可能很闲，没事儿干，饭店的成本也高了；如果客流很大，而服务员人数少了，可能就导致接待不过来顾客，导致顾客吃饭体验差。因此，饭店要根据客户的流量计并发量来调整接待的服务人员数量，然后根据检测顾客量变化及时调整最佳的配置。

nginx服务就相当于饭店，网站用户就相当于顾客，nginx的进程就相当于服务员，下面就来优化nginx进程的个数。

1、优化Nginx进程对应的配置

优化Nginx进程对应nginx服务的配置参数如下

worker\_processes 1; # 指定了nginx要开启的进程数，结尾的数字就是进程的个数。

[root@Web01 conf]# vim nginx.conf

[root@Web01 conf]# cat nginx.conf

worker\_processes 8; ###最好是cpu的核数或者x2

上述参数调整的是Nginx服务的Worker进程数，nginx有Master进程和worker之分，Master为管理进程，真正接待顾客的是worker进程

2、优化Nginx进程个数的策略

前面已经说过，worker\_processes参数大小的设置最好和网站的用户关联，那么新配置不知道网站的用户数量该如何配置呢？

搭建服务器时，worker进程数最开始的设置可以等于CPU的核数，且worker进程数要多一些，这样起始提供服务时就不会出现访问量快速增加而临时启动新进程提供服务的问题，减少了系统的瞬时开销和提供服务的时间，提升了服务用户的速度。高流量高并发场合也可以考虑提高至CPU核数\*2的进程数，具体情况要根据实际的业务来选择，因为这个参数，除了要和CPU核数匹配外，和硬盘存储的数据以及系统的负载也会有关，设置为CPU的个数或核数是一个好的起始配置，这也是官方的建议。

3、查看Web服务器CPU硬件资源信息

通过/proc/cpuinfo可查看CPU个数及总核数。

查看CPU总核数：

[root@Web01 /]# grep processor /proc/cpuinfo |wc -l

1

[root@Web01 /]# grep -c processor /proc/cpuinfo

1

看看CPU总颗数的示例如下

[root@Web01 /]# grep 'physical id' /proc/cpuinfo|sort|uniq|wc -l

1

通过执行top命令然后按数字1，即可显示所有的CPU核数

4. 实践修改nginx配置

假设服务器的CPU可数为1课，核数为4核，则初始的配置可通过查看默认的nginx.conf里的worker\_processes数来了解：

[root@lnmp conf]# grep worker\_processes nginx.conf

worker\_processes 1;

这里修改参数值为CPU的总核数4，然后重新加载nginx服务，操作过程及结果。

修改配置的方式如下：

[root@lnmp conf]# sed -i 's#worker\_processes 1;#worker\_processes 4;#g' nginx.conf

[root@lnmp conf]# grep worker\_processes nginx.conf

worker\_processes 4;

优雅重启nginx，使修改生效，如下：

[root@lnmp conf]# /application/nginx/sbin/nginx -t

nginx: the configuration file /application/nginx-1.6.3/conf/nginx.conf syntax is ok

nginx: configuration file /application/nginx-1.6.3/conf/nginx.conf test is successful

[root@lnmp conf]# /application/nginx/sbin/nginx -s reload

现在检查修改后的worker进程数量，如下：

[root@lnmp conf]# ps -ef|grep nginx|grep -v grep

nginx 58412 58191 0 22:06 ? 00:00:00 nginx: worker process

nginx 58413 58191 0 22:06 ? 00:00:00 nginx: worker process

nginx 58414 58191 0 22:06 ? 00:00:00 nginx: worker process

nginx 58415 58191 0 22:06 ? 00:00:00 nginx: worker process

从 “worker process 4;”可知，worker的进程数为4个。nginx master主进程不包含在这个参数内，nginx master的主进程为管理进程，负责调度和管理worker进程。

有关worker\_processes参数的官方说明：

syntax: worker\_processes number; #此行参数语法，number为数量。

default： worker\_processes 1; #此行意思是不配置该参数，软件默认情况数量为1.

context： main #此行为worker\_processes参数可以放置的位置。

worker\_processes为定义worker进程数的数量，建议设置为CPU的核数或CPU核数\*2的进程数，具体情况要根据实际的业务来选择，因为这个参数，除了要和CPU核数的匹配外，和硬盘存储的数据以及系统的负载也会有关，设置为CPU的个数或核数是一个好的起始配置。

### 1.2.2 优化绑定不同的nginx进程到不同的cpu上

默认情况nginx的多个进程由可能跑在某一个或某一核的CPU上，导致nginx进程使用硬件的资源不均，本节的优化是尽可能第分配不同的nginx进程给不同的CPU处理，达到充分有效利用硬件的多CPU多喝资源的目的。

1. 优化nginx进程对应不同的CPU配置

在优化不同放到nginx进程对应不同CPU配置时，四核CPU服务器的参数配置参考如下：

worker\_processes 4;

worker\_cpu\_affinity 0001 0010 0100 1000;

##worker\_cpu\_affinity就是配置nginx进程CPU亲和力的参数，即把不同的进程分给不同的CPU处理。这里0001 0010 0100 1000是掩码，分别代表第1、2、3、4核CPU核心，由于worker\_processes进程数为4，因此，上诉配置会把每个进程分配一核CPU处理，默认情况下进程不会绑定任何CPU，参数位置为main段。

八核CPU服务器的参数配置参考如下：

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

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

下面是绑定的示例配置。

worker\_processes 4;

worker\_cpu\_affinity 0001 0010 0100 1000;

binds each worker process to a separate CPU, while

worker\_processes 2;

worker\_cpu\_affinity 0101 1010;

binds the first worker process to CPU0/CPU2, and the second worker process to CPU1/CPU3. The second example is suitable for hyper-threading.The directive is only available on FreeBSD and Linux.

From : http://nginx.org/en/docs/ngx\_core\_module.html by oldboy

taskset –retrieve or set a process’s CPU affinity 命令本身也有分配CPU的功能，留给大家测试（例如：taskset –c 1,2,3 /etc/init.d/mysql start）

### 1.2.3 nginx事件处理模型优化

nginx的连接处理机制在于不同的操作系统会采用不同的I/O模型，在Linux下，nginx使用epoll的I/O多路复用模型，在FreeBSD中使用kqueue的I/O多路复用模型，在Solaris中使用/dev/poll方式的I/O多路复用模型，在windows使用的是icop，等等。

要根据系统类型选择不同的事件处理模型，可供使用的选择有“use [ kqueue|rtsig|epoll|/dev/poll|select|poll];”。此次使用的是centos6.7Linux，因此将nginx的事件处理模型调整为epoll模型。

具体的配置参数如下：

[root@Web01 conf]# vim nginx.conf

events { ###是设定nginx的工作模式及连接数上限

worker\_connections 1024;

use epoll;

} ###use是和事件模块指令，用来指定nginx的工作模式，nginx支持的工作模式有很多，其中select和poll都是标准的工作模式，kqueue和epoll是高效的工作模式，不同的是epoll用在Linux平台上，而kqueue用在BSD系统中。对于Linux系统Linux2.6+的内核，推荐选择epoll工作模式，这是高并发的设置。

根据nginx的官方文档建议，也可以不指定事件处理模型，nginx会自动选择最佳的事件处理模型服务。

events区块及use事件处理参数的官方说明如下：

syntax: events{...} #语法配置

default： — #缺省没有设置

context： main #events标签的放置位置，放在main段。

events区块是一个用来设置连接进程的区块，例如：设置nginx的网络I/O模型，以及连接数等。

use事件的处理参数说明如下：

syntax: use method; #网络模型配置，method选择模型之一。

default： — #缺省没有设置

context： events #网络模型配置放置于events区块内。

设置使用连接进程的方法，通常不需要任何设置，nginx会自动选择最有效的方法。

### 1.2.4 调整nginx单个进程允许的客户端最大连接数

接下来，调整nginx单个进程允许客户端的最大连接数，这个控制连接数的参数为worker\_connections；

worker\_connections设置说明：

这个参数的值要根据具体服务器性能和程序的内存使用量来指定（一个进程启动使用更多内存程序确定）

events { ###是设定nginx的工作模式及连接数上限

worker\_connections 4096; ##3万并发

}

worker\_connections也是事件模块指令，用于定义nginx每个进程的最大连接数，默认是1024；最大客户端连接数由worker\_processes和worker\_connections决定，即Max\_client=worker\_processes\*worker\_connections。进程的最大连接数受Linux系统进程的最大打开文件数限制，在执行操作系统命令“ulimit –HSn 65535”或配置相应文件后worker\_connections的设置才能生效。

有关worker\_connections的官方说明如下：

参数语法：worker\_connections number；

默认配置：worker\_connections 512；

防止位置：events

说明：worker\_connections用来设置一个worker process支持的最大并发连接数，这个连接数包括了所有连接，例如：代理服务器的连接、客户端的连接等，实际的并发连接数除了受worker\_connections参数控制外，还和最大打开文件数worker\_rlimit\_nofile有关。nginx总并发连接=worker数量\* worker\_connections。

### 1.2.5 配置nginx worker进程最大打开文件数

调整配置nginx worker 进程最大打开文件数，这个控制连接数的参数为worker\_rlimit\_nofile：

worker\_rlimit\_nofile参数配置

worker\_rlimit\_nofile 65535;

最大打开文件数，可设置为系统优化后的ulimit-HSn的结果。

调整配置nginx worker 进程最大打开文件数，这个控制连接数的参数为worker\_rlimit\_nofile number的官方说明如下：

参数语法：worker\_rlimit\_nofile number;

默认配置：无

防止位置：主标签段

说明：此参数的作用是改变worker processes能打开的最大文件数。

### 1.2.6 优化服务器域名的hash表大小

先要确切名字和通配符名字存储在哈希表中。哈希表和监听端口关联，每个端口都最多关联到三张表：确切名字的哈希表，以星号起始的通配符名字的哈希表和以星号结束的通配符名字的哈希表。哈希表的尺寸在配置阶段进行了优化，可以以最小的CPU缓存命中失败来找到名字。nginx首先会搜索确切名字的哈希表，如果没有找到，则搜索以星号起始的通配符名字的哈希表，如果还是没有找到，继续搜索以星号结束的通配符名字的哈希表。因为名字是按照域名的节来搜索的，所以搜索通配符名字的哈希表比搜索确切名字的哈希表慢。注意.nginx.org存储在通配符名字的哈希表中，而不再确切名字的哈希表中。正则表达式是一个一个串行的测试，所以是最慢的，而且不可扩展。

鉴于以上原因，请尽量使用确切的名字。举个例子，如果使用nginx.org和[www.nginx.org](http://www.nginx.org)来访问服务器时最频繁的，那么将它们明确的定义出来就更为有效：

server {

listen 80;

server\_name [www.nginx.org](http://www.nginx.org) \*.nginx.org;

…

}

下面这种方法相比更简单，但是效率也更低：

server {

listen 80;

server\_name .nginx.org;

…

}

如果定义了大量的名字，或者定义了非常长的名字，那就需要在http配置块中调整server\_nmaes\_hash\_max\_szie和server\_names\_hash\_bucket\_size的值。

server\_names\_hash\_bucket\_size的默认值可能是32或者64，也可能是其他的值，这取决于CPU的缓存行的长度。如果这个值是32，那么定义“too.long.server.name.nginx.org”作为虚拟主机名就会失败，此时会显示下面的错误信息：

could not build the server\_names\_hash,

you should increase server\_names\_hash\_bucket\_szie:32

出现了这种情况，那就需要将设置值扩大一倍：

server {

server\_names\_hash\_bucket\_size 64;

…

}

如果定义了大量名字，得到了另外一个错误：

could not build the server\_names\_hash,

you should increase either server\_names\_hash\_max\_size: 512

or server\_names\_hash\_bucket\_size:32

那么应该先尝试设置sever\_names\_hash\_max\_szie的值，此值差不多等于名字列表的名字总量。如果还不能解决问题，或者服务器启动非常缓慢，再尝试提高server\_names\_hash\_backet\_size的值。

如果只为一个监听端口配置了唯一的主机，那么nginx就完全不会测试虚拟主机名了（也不会建立那些名字哈希表）。不过，有一个例外，如果server\_name是一个含有捕获组的正则表达式，这时nginx就不得不执行这个表达式以得到捕获组。

server\_names\_hash\_bucket\_size:512;

默认是512kb，一般要查看系统给出确切的值。这里一般是CPU L1的4-5倍

server\_names\_hash\_bucket\_size:32的官方说明如下：

syntax: server\_names\_hash\_bucket\_size size #参数语法

default： server\_names\_hash\_bucket\_size #参数默认大小

context： http #仅能放置在http标签段

参数作用：设置存放域名（server names）的最大哈希表的存储桶（bucket）的大小。默认值依赖CPU的缓存行。

### 1.2.7 开启高效文件传输模式

1、设置参数:sendfile on;

###sendfile 参数用于开启文件的高效传输模式，同时将tcp\_nopush和tcp\_nodelay两个指令设置为on，可防止网站及磁盘IO阻塞，提升nginx工作效率

sendfile 参数的官方说明如下：

syntax: sendfile on | off #参数语法

default： sendfile off; #参数默认大小

context： http,server,location,if in location #可以放置在http标签段

参数作用：激活或禁用sendfile()功能。sendfile()是作用于两个文件描述符之间的数据拷贝函数，这个拷贝操作是在内核之中的，被称为“零拷贝”，sendfile()比read和write函数要高效很多，因为，read和write函数要把数据拷贝到应用层再进行操作。

2、设置参数：tcp\_nopush on;

syntax: tcp\_nopush on | off #参数语法

default： tcp\_nopush off; #参数默认大小

context： http,server,location #可以放置在http标签段

参数作用：激活或者禁用Linux上的TCP\_CORK socket选项，此选项仅仅当开启sendfile时才生效，激活这个.-tcp\_nopush参数可以允许吧http response header 和文件的开始放在一个文件里发布，积极的作用是减少网络报文段的数据量。

细节见：http://nginx.org/en/docs/http/ngx\_http\_core\_module.html

### 1.2.8 优化Nginx连接参数调整连接超时时间

**1. 什么是连接超时？**

比喻：某饭店请了服务员招待顾客，但是现在饭店不景气，此时，为多余的服务员发工资使得成本被提高，想节约饭店开支成本旧的解雇服务员。

这里的服务员就相当于Nginx服务建立的而连接，当服务员建立的连接没有接收处理请求时。在指定的时间内就让它超时自动退出。还有nginx和fastcgi服务建立连接请求PHP时，因为一些原因（负载高、停止响应）fastcgi服务无法给nginx返回数据，此时可以通过配置nginx服务参数。使其不会死等，因为前面用户还等着它返回数据呢，例如，可设置为如果请求Fastcgi，10秒内不能返回数据，那么nginx就中断本次请求，想用户汇报取不到数据的错误。

**2. 连接超时的作用**

（1）设置将无用的连接尽快超时，可以保护服务器的系统资源（CPU、内存、磁盘）

（2）当连接很多时，及时断掉那些已经建立好的但有长时间不做事的连接，以减少其占用的服务器资源，因为服务器维维护连接也是消耗资源的。

（3）有时黑客或者恶意用户攻击网站，就会不断地和服务器建立连接，消耗连接数，但是啥也不干，只是持续建立连接这机会大量消耗服务器的资源，此时就应该及时断掉这些恶意占用资源的连接。

（4）LNMP环境中，如果用户请求了动态服务，则nginx就会建立连接请求fastcgi服务以及mysql服务，此时这个nginx连接就要设定一个超时时间，在用户容忍的 时间内返回数据，或者再多等一会后端服务返回数据，具体的策略呀具体业务具体分析，当然了，后端的fastcgi服务以及mysql服务也有对连接的超时控制。

**简单说，连接超时是服务的一种自我管理、自我保护的重要机制。**

**3． 连接超时带来的问题**

（1）服务器建立新连接也是要消耗资源的，因此，超时设置的太短而并发很大，就会导致服务器瞬间无法响应用户的请求，导致体验下降。

（2）企业生产有些PHP程序站点希望设置短连接，因为PHP程序建立连接消耗的而资源和时间要少。而Java城西站点一般建议设置长连接，因为Java程序建立连接消耗的额资源和时间更多，这是语言运行机制决定的。

**4． nginx连接超时的参数设置**

（1）设置参数：keepalive\_timeout 60; ##设置客户端连接保持会话的超时时间为60秒。超时时间，服务器会关闭该连接，此数值参考数值。

keepalive\_timeout参数的官方说明如下：

syntax: keepalive\_timeout timeout {header\_timeout}; ##参数语法

default: keepalive\_timeout 75s ##参数默认大小

context: http,server,location ##可以放置的标签段

参数作用：

keep-alive作用可以使客户端到服务器端已经建立的连接一致工作不退出，当服务器有持续请求时，keep-alive可以使用正在建立的连接提供服务，从而避免服务器重新建立新连接处理请求。

此参数是设置一个keep-alive客户端连接在服务器端保持多久后退出，这个参数的单位是秒，和http相应header域的“keep-alive:timeout=time”参数有关，这些header信息也会被客户端浏览器识别并处理，不过有些客户端并不能按照服务器端的设置处理，例如：MSIE大约60秒回关闭keep-alive连接。

**细节见：http://nginx.org/en/docs/http/ngx\_http\_core\_module.html**

2、设置参数：tcp\_nodelay on;

#激活tcp\_nodelay功能，提高IO性能。

tcp\_nodelay参数的官方说明如下：

syntax: tcp\_nodelay on | off #参数语法

default： tcp\_nodelay off; #参数默认大小

context： http,server,location #可以放置在http标签段

参数作用：

默认情况下数据发送时，内核并不会马上发送，可能会等待更多的字节组成一个数据包，这样可以提高I/O性能，但是，在每次只发送很少字节的业务场景，使用tcp\_nodelay功能，等待时间会比较长。

参数生效条件：激活或禁用TCP\_NODELAY选项，当一个连接进入到keep-alive状态时生效。

**细节见：http://nginx.org/en/docs/http/ngx\_http\_core\_module.html**

3、设置参数：client\_header\_timeout 15;

#设置读取客户端请求头数据的超时时间。如超过这个时间，客户端还没有发送完整的header数据，服务端将返回“Request time out（408）”错误，次数的数值15单位是秒，为经验参考值。

client\_header\_timeout参数的官方说明如下：

syntax: client\_header\_timeout time #参数语法

default： client\_header\_timeout 60s #参数默认大小

context： http,server #可以放置的标签段

参数作用：

设置读取客户端请求头数据的超时时间，如超过这个时间，客户端还没有发送完整的header数据，服务端将返回“Request time out(408)”错误，指定一个超时时间防止客户端利用http协议进行攻击。

**细节见：http://nginx.org/en/docs/http/ngx\_http\_core\_module.html**

4、设置参数：client\_body\_timeout 15;

#设置读取客户端请求主体的超时时间，默认是60

client\_body\_timeout参数的官方说明如下：

syntax: client\_body\_timeout time #参数语法

default： client\_body\_timeout 60s #默认值是60秒

context： http,server，location #可以放置的标签段

参数作用：

设置读取客户端请求主体的超时时间。这个超时仅仅为两次成功的读取操作之间的一个超时，非请求整个主体数据的超时时间，如在这个超时时间内，客户端没有发送任何数据，nginx将返回“Request time out(408)”错误，默认值是60.

**细节见：http://nginx.org/en/docs/http/ngx\_http\_core\_module.html**

5、 设置参数：send\_timeout 25;

#指定响应客户端的超时时间，这个超时仅限于两个连接活动之间的时间，如果超过这个时间，客户端没有任何活动，nginx将会关闭连接，默认值为60s，可以改为参考值25s。

send\_timeout参数的官方说明如下：

syntax: send\_timeout time; #参数语法

default： send\_timeout 60s; #默认值是60秒

context： http,server，location #可以放置的标签段

参数作用：

设置服务器端传送http响应信息到客户端的超时时间，这个超时仅仅为两次成功握手后的一个超时，非请求整个响应数据的超时时间，如在这个超时时间内，客户端没有接收任何收据，连接将被关闭。

**细节见：**[**http://nginx.org/en/docs/http/ngx\_http\_core\_module.html**](http://nginx.org/en/docs/http/ngx_http_core_module.html)

### 1.2.9 上传文件大小（http Request body size）的限制（动态应用）

下面介绍如何调整上传文件大小（http Request body size）限制。

首先，在nginx的主配置文件里加入如下参数：

client\_max\_body\_size 8m;

具体大小根据公司的业务做调整，如果不清楚就8M，有关客户端请求主体的解释在前文http原理一节已经解释过了，一般http的post方法提交数据才会携带请求主体信息。

client\_max\_body\_size参数的官方说明如下：

syntax: client\_max\_body\_size size; #参数语法

default： client\_max\_body\_size 1m #默认值是1m

context： http,server，location #可以放置的标签段

参数作用：

设置最大的允许的客户端请求主题大小，在请求头域有”Content-Length”，如果超过了此配置值，客户端会受到413错误，意思是请求的条目过大，有可能浏览器不能正确显示这个错误，设置为0表示禁止检查客户端请求主体大小，此参数对服务端的安全有一定的作用。

**细节见：**[**http://nginx.org/en/docs/http/ngx\_http\_core\_module.html**](http://nginx.org/en/docs/http/ngx_http_core_module.html)

### 1.2.10 fastcgi 相关参数调优（配合PHP引擎动态服务）

fastCGI参数是配合Nginx向后请求PHP动态引擎服务的相关参数，具体的逻辑图如下图所示。



此处讲解的参数均为nginx fastcgi客户端向后请求PHP动态引擎服务（php-fpm（fastcgi服务端））的先关参数，属于nginx的配置参数。下面将nginx fastcgi常见参数列表说明：

Nginx FastCGI常见参数列表说明

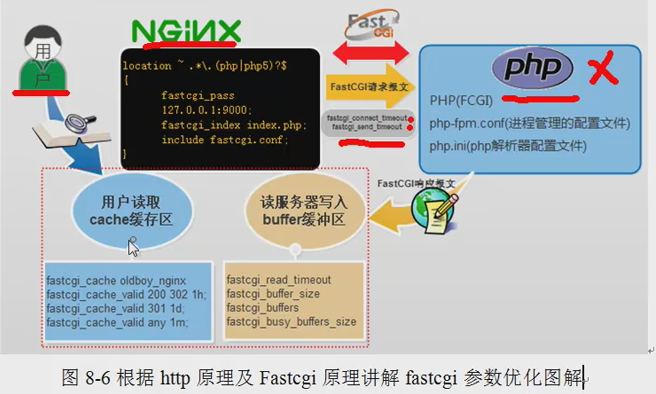
|  |  |
| --- | --- |
| **Nginx FastCGI相关参数** | **说明** |
| fastcgi\_connect\_timeout | 表示Nginx服务器和后端FastCGI服务器李连杰的超时时间，默认值为60秒，这个参数值通常不要超过75s，因为建立的连接数越多消耗的资源就越多 |
| fastcgi\_send\_timeout | 设置Nginx允许FastCGI服务端返回数据的超时时间，即在规定时间之内后端服务器必须传完所有的数据，否则，Nginx将断开这个连接，默认值为60s |
| fastcgi\_read\_timeout | 设置Nginx从FastCGI服务端读取响应信息的超时时间，表示连接建立成功后，nginx等待后端服务器的响应时间，是Nginx已经进入都断的排队之中等候处理的时间 |
| fastcgi\_buffer\_size | 这是Nginx FastCGI的缓冲区大小参数，设定用来读取从FastCGI服务端收到的第一部分响应信息的缓存区大小，这里的第一部分通常会包含一个小的响应头部，默认情况，这个参数是由fastcgi\_buffers指定的一个缓冲区 |
| fastcgi\_buffers | 设定用来读取从FastCGI服务端收到的响应信息的缓冲区大小以及缓冲区的数量，默认值是fastcgi\_buffers 8 4k|8K  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，那么会为其分配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，但这并不是好方法，因为内存中的数据处理速度要快于硬盘，一般这个值应该为站点中的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那可以把这个值设置为“16 16k”、“4 64K”等 |
| proxy\_busy\_buffers\_size | 用于设置系统很忙，可以使用proxy\_ buffers大小，官方推荐的大小为proxy\_buffers\*2 |
| fastcgi\_busy\_buffers\_size | 用于设置系统很忙时可以使用fastcgi\_ buffers大小，官方推荐大小为fastcgi\_ buffers \*2  默认值fastcgi\_busy\_buffers\_size 8k|16k |
| fastcgi\_temp\_file\_write\_size | fastc临时文件的大小，可设置128-256k |
| Fastcgi\_cache oldbo\_nginx | 表示开启fastcgi缓存并未其指定一个名称。开启缓存非常有用，可以有效降低CPU的负载，并且防止502错误的发生，但是开启缓存也可能会引起其他问题，要根据具体情况选择。 |
| Fastcgi\_cache\_path | 示例：fastcgi\_cache\_path /data/ngx\_fcgi\_cache level=2:2 keys\_zone=fcgi\_cache:512min active=1d max\_size=40g;  Fastcgi\_cache缓存目录，可以设置目录哈希层级，比如2:2会生成256\*256个子目录，keys\_zone是这个缓存空间的名字，cache是用多少内存（这样热门的内容nginx直接放内存，提高访问速度），inactive表示默认失效时间，max\_size表示最多用多少硬盘空间，需要注意的是fastcgi\_cache缓存是先写在fastcgi\_temp\_path再移到fastcgi\_chache\_path，所以这两个目录最好在同一个分区，从0.8.9之后可以在不同的分区，不过还是建议放同一个分区。 |
| Fastcgi\_cache\_valid | 示例：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应答缓存一个小时  示例：fastcgi\_cache\_valid 301 1d;  将301应答缓存1天  示例：fastcgi\_cache\_valid any 1m;  将其他应答缓存为1分钟。 |
| Fastcgi\_cache\_min\_uses | 示例：fastcgi\_cache\_min\_user 1;  设置请求几次之后相应将被缓存，示例表示1此即被缓存。 |
| Fastcgi\_cache\_use\_stale | 示例：fastcgi\_cache\_use\_stale error timeout invalid\_header http\_500;  定义那些情况下用过期缓存。 |
| Fastcgi\_cache\_key | 示例：fastcgi\_cache\_key $request\_method://$host$request\_uri;  Fastcgi\_cache\_key <http://$host$request_uri>;  定义fastcgi\_cache的key，示例中就以请求的URL作为缓存的key，nginx会取这个key的MD5作为缓存文件，如果设置了缓存哈希目录，nginx会从后往前取相应的位数作为目录。注意一定要加上$request\_method作为cache key，否则如果head类型的先请求会导致厚点的get请求返回为空。 |

fastcgi cache资料：<http://nginx.org/en/docs/http/ngx_http_fastcgi_module.html>

fastcgi常见参数的nginx配置示例如下：

fastcgi\_connect\_timeout 240;  
fastcgi\_send\_timeout 240;  
fastcgi\_read\_timeout 240;  
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 keys\_zone=ngx\_fcgi\_cache:512m inactive=1d max\_size=40g;

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;  
         }



特别注意：nginx的fastcgi的相关参数和反向代理proxy的相关参数非常接近，例如 http proxy模块相关参数说明：

|  |  |
| --- | --- |
| **http proxy模块相关参数** | **说明** |
| Proxy\_connect\_timeout | 表示反向代理与后端节点服务器连接的超时时间，即发起握手等候响应的超时时间。 |
| Proxy\_send\_timeout | 表示代理后端服务器 的数据回传时间，即在规定时间之内后端服务器必须传完所有的数据，否则，nginx将断开这个连接。 |
| Proxy\_read\_timeout | 设置nginx从代理的后端服务器获取信息的时间，表示连接建立成功后，nginx等待后端服务器的响应时间，是nginx已经进入后端的排队之中等候处理的时间。 |
| Proxy\_buffer\_size | 代理请求缓冲区大小，用于缓存头部信息以供nginx进行一些规则处理，默认该缓冲区大小等于指令proxy\_buffers设置的大小。 |
| Proxy\_buffers | 设置缓冲区的数量大小。nginx从代理的后端服务器获取的响应信息，会放置到缓冲区。 |
| Proxy\_busy\_buffers\_size | 用于设置系统很忙时可以使用的proxy\_buffers大小，官方推荐的大小为proxy\_buffers\*2 |
| Proxy\_temp\_file\_write\_size | 指定代理缓存临时文件的大小。 |

### 1.2.11 配置nginx gzip压缩实现性能优化

**1. nginx gzip压缩功能介绍**

nginx gzip压缩模块提供了对文件内容压缩的功能，nginx服务器将用户请求的内容在发送出用户客户端之前会根据一些具体的策略实施压缩，以节约网站出口带宽，同时加快课数据传输效率，提升了用户访问体验。

**2． nginx gzip压缩的优点**

* 提升网站用户体验

由于发给用户打内容小了，所以用户访问单位大小的页面就快开了，用户体验提升了，网站口碑就好了。

* 节约网站带宽成本

由于数据时压缩传输的，因此，此举节省了网站的带宽流量成本，不过压缩时会稍微消耗一些CPU资源，这个一般可以忽略。

此功能既让用户舒服了，公司也少花钱了，一举多得。对于几乎所有的web服务来说，这是一个非常重要的功能，Apache服务也有此功能。

**3． 需要和不需要压缩的对象**

* 纯文本内容压缩比很高，因此，纯文本的内容最好要压缩，例如：html、js、css、xml、shtml等格式的文件。
* 被压缩的纯文本文件必须要大于1k，由于压缩算法的特殊原因，极小的文件压缩可能会变大。
* 图片、视频（流媒体）等文件尽量不要压缩，因为这些文件大多都是经过压缩的，如果再压缩很可能不会减少或减少很多，或者有可能增大，而在压缩是还会消耗大量的CPU、内存资源。

**4． 参数介绍及配置说明**

此压缩功能类似早起的Apache服务的mod\_deflate压缩功能，nginx的gzip压缩功能依赖于ngx\_http\_gzip\_module模块，默认已安装。

对应的压缩参数说明如下：

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-legnth中获取。默认值为0，表示不管页面多大都进行压缩。建议设置成大于1k，如果小于1k可能会越压越大。  
gzip\_buffers     4 16k;

#压缩缓冲区大小。表示申请4个单位为16K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;

压缩比率。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，但处理最慢，也比较消化CPU资源。  
#gzip\_types  text/plain application/x-javascript text/css application/xml; #<==用来指定压缩的类型，”test/html”类型总是被压缩，这个就是http原理的媒体类型。  
gzip\_types  text/css text/xml application/javascript;   
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。

完整的配置如下：

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;

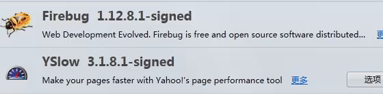
gzip\_types类型不同的版本配置可能会有不同，上述是适合1.6.3的nginx。对应的文件类型查看安装目录下的mime.types文件。

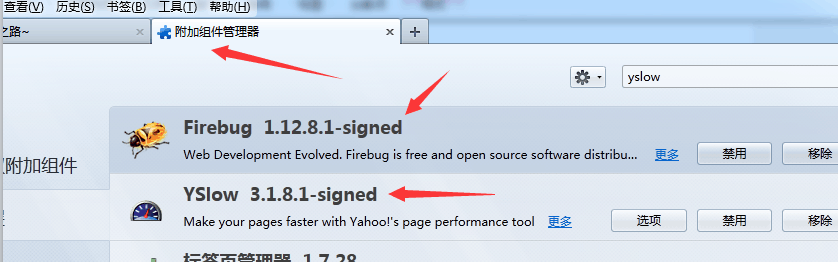
**更多官方资料请看：http://nginx.org/en/docs/http/ngx\_http\_gzip\_module.html**

**5、Nginx压缩配置效果检查**

通过火狐浏览器加yslow插件查看gzip及expires缓存结果：

提前安装好火狐浏览器，并且安装好yslow插件，开启监控，然后打开lnmp时安装的博客地址，就可以看到下图所示的压缩结果。





### 1.2.12 Nginx expires缓存实现性能优化

**1、nginx expires功能介绍**

简单地说, nginx expires功能就是为用户访问的网站内容设定一个过期时间，当用户第一次访问到这些内容时，会把这些内容存储在用户浏览器本地，这样用户第二次及以后继续访问该网站，浏览器会检査加载已经缓存在用户浏览器本地的内容，就不会去服务器下载了，直到缓存的内容过期或被清除为止。

深入一点理解，expires功能就是允许通过Nginx配置文件控制HTTP的"Expires”和Cache-Control"响应头部内容，告诉客户端浏览器是否缓存成及缓存多久访问的内容。这个 expires模块控制Nginx服务器应答时的Expires头内容和 Cache\_Contro1头的 max-age指令。 缓存的有效期可以设置为相对于源文件的最后修改时刻或者客户端的访问时刻。

这些HTTP头向客户端表明了内容的有效性和持久性。如果客户端本地有内容缓存，则内容就可以从缓存（除非已经过期）而不是从服务器读取，然后客户端检查缓存中的副本，看看是否过期或失效，以决定是否重新从服务器获得内容更新。

**2. nginx expires作用介绍**

在网站的开发和运营中，对于视频、图片、CSS、JS等网站元素的更改机会较少，特别是图片，这时可以将图片设置在客户浏览器本地缓存365天或3650天，而将CSS、JS、HTML等代码缓存10~30天，这样用户第一次打开页面后，会在本地的浏览器按照过期日期缓存相应的内容，下次用户再打开类似的页面，重复的元素就无需下载了，从而加快了用户访问速度，由于用户访问请求和数据减少了，因此节省了服务器端大量的带宽。此功能同Apache的expires。

**3. nginx expires功能优点**

* expires可以降低网站的带宽，节约成本。
* 加快用户访问网站的速度，提升了用户访问体验。
* 服务器访问量降低了，服务器压力就减轻了，服务器成本也会降低，甚至可以节约人力成本。

对于几乎所有web服务来说，这是非常重要的功能之一，Apache服务也有此功能。

**4. nginx expires配置详解**

前面已经介绍了expires的功能原理，接下来就是配置nginx expires的功能，这里以location标签为例进行讲解，通过location URL规则将需要缓存的扩展名列出来，然后指定缓存时间，如果针对所有内容设置缓存，也可以不用location，nginx默认安装了expires功能。

1） 根据文件扩展名进行判断，添加expires功能范例。

范例1：

location ~.\*\. (fig|jpg|jpeg|png|bmp|swk)$

{

expires 3650d;

}

说明：范例1的意思是当用户访问网站URL结尾的文件扩展名为上述指定的各种类型的图片时，设置缓存3650天，即10年。

范例2：

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

{

expires 30d;

}

说明：范例2的意思是当用户访问网站URL结尾的文件扩展名为js,css类型元素时，设置缓存30天，即一个月。

2） 根据URL中的路径（目录）进行判断，添加expires功能范例.

范例3：

location ~ ^/(images|javascripts|js|css|flash|media|staticc)/

{

expires 360d;

}

范例3的意思是当用户访问网站URL中包含了上述路径（例：images、js、css，这些在服务端是程序目录）时，把访问的内容设置缓存360天，即1年。

**6. nginx expire 功能缺点及解决方法**

几乎所有的事物都是有两面性，没有十全十美的人和事。nginx expire功能也不例外，虽然这个功能很好，但是也会给企业带来一些困惑。

当网站被缓存的页面或数据更新了，此时用户端看到的可能还是旧的已经缓存的内容，这样就会影响用户体验，那么如何解决这个问题呢？

**解决办法如下几个：**

1. 对于经常需要变动的图片等文件，可以缩短对象缓存时间，例如：谷歌和百度的首页的图片经常根据不同的日期换成一些节日的图，所以这里可以将这个图片设置为缓存期1天。

2. 当网站改版或更新内容时，可以在服务器将缓存的对象改名（网站代码程序）。

* 对于网站的图片、附件，一般不会被用户直接修改，用户层面上的修改图片，实际上是重新传到服务器，虽然内容一样但是是一个新的图片名了。
* 网站改版会修改JS、CSS元素，若改版的时候对这些元素改了名，会使得前端的CDN以及用户端需要重新缓存内容。

**7. 企业网站缓存日期曾经的案例参考**

若企业的业务和网站访问量不同，那么往后在哪的缓冲期时间设置也是不同的，比如，如下企业所用的缓存日期就是不一样的。

* 51cto：1周。
* sina：15天
* 京东：25年
* 淘宝：10年

**8.企业网站有可能不希望被缓存的内容**

* 广告图片，用于广告服务，都缓存了就不好控制展示了。
* 网站流量统计工具（js代码），都缓存了流量统计就不准了。
* 更新很频繁的文件（Google和logo），这个如果按天，缓存效果还是显著的。

## 1.3 nginx日志相关优化与安全

### 1.3.1 编写脚本实现nginx access 日志轮询

当用户请求一个软件时，绝大多数软件都会记录用户的访问情况，nginx服务也不例外，nginx软件目前还没有类似Apache的通过cronolog或者rotatelog对日志分割处理的功能，但是，运维人员可以通过利用脚本开发及nginx的信号控制功能或者reload重新加载，来实现日志的自动切割、轮询。

详细操作过程如下：

**1. 配置日志切割脚本。**

[root@lnmp server]# cd

[root@lnmp ~]# cd /server/scripts/

[root@lnmp scripts]# vim cut\_nginx\_log.sh

[root@lnmp scripts]# cat cut\_nginx\_log.sh

#!/bin/sh

cd /appcation/nginx/logs &&\

/bin/mv www\_access.log www\_access\_$(date +%F -d -1day).log #将日志按日期改成前一天的名称

/application/nginx/sbin/nginx -s reload #重新加载nginx使得触发重新生成访问日志文件

提示：实际上脚本的功能很简单，就是改名日志，然后加载nginx，重新生成文件记录日志。

**2. 将这段脚本保存后加入到服务器端的定时任务配置里，让此脚本在每天凌晨0点执行，就可以实现日志的每天分割功能，操作结果如下：**

[root@lnmp scripts]# crontab –e

#cut nginx access log by oldboy at 201509

00 00 \* \* \* /bin/sh /server/scripts/cut\_nginx\_log.sh >dev/null 2>&1

### 1.3.2 不记录不需要的访问日志

在实际工作中，对于负载均衡健康减产节点或某些特定文件（比如图片、JS、CSS）的日志，一般不需要记录下来，因为在统计PV时按照页面计算的。而且日志写入太频繁会大量消耗磁盘I/O，降低服务的性能。

**具体配置方法为：**

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

access\_log off;

}

说明：用location标签匹配不记录日志的元素扩展名，然后关掉日志。

### 1.3.3 访问日志的权限设置

加入日志命令为/app/logs，则收钱方法为：

  chown -R root.root /app/logs  
  chmod -R 700 /app/logs

不需要在日志目录上给nginx用户读或写的许可，很多人直接给了nginx或Apache用户，这就成为了安全隐患。没必要给那么大权限，nginx有master进程root用户，控制可以写入日志。

## 1.4 nginx站点目录及文件URL访问控制

### 1.4.1 根据扩展名闲置程序和文件访问

web2.0时代，绝大多数网站都是以用户为中心，例如：bbs、blog、SNS产品，这几个产品都有一个共同特点，就是不但允许用户发布内容到服务器，还允许用户发图片甚至附件到服务器上，由于为用户开了上传的功能，因此给服务器带来了很大的安全风险，虽然很多程序在上传前会做一定的控制，例如：文件大小、类型等，但是，一不小心就会被黑客钻了空子，上传了木马程序。

下面是利用nginx配置禁止访问上传资源目录下的php、shell、Perl、Python程序文件，这样用户即使上传了木马文件也没法去执行，从而加强了网站的安全。

这样用户即使上传了木马文件也没法去执行，从而加强了网站的安全。

**范例1：配置nginx，禁止解析指定目录下的指定程序**

location ~ ^/images/.\*\.(php|php5|.sh|.pl|.py)$   
        {   
       deny all;   
        }   
location ~ ^/static/.\*\.(php|php5|sh|pl|py)$   
        {   
           deny all;   
        }   
location ~\* ^/data/(attachment|avatar)/.\*\.(php|php5)$   
    {   
        deny all;   
    }

禁止http协议GET方法。

对上述目录的闲置必须写在nginx处理php服务location的前面

### 1.4.2 禁止访问指定目录下所有文件和目录

范例1：配置禁止访问指定的单个或多个目录。  
**单目录：**

location ~ ^/(static)/ {

deny all;

}

location ~ ^/static {

deny all;

}

**多目录：**

location ~ ^/(static|js) {

deny all;

}  
 

作用：禁止目录下指定文件被访问，或禁止指定目录所有内容被访问。

最佳应用场景：对于集群的共享存储，一般是存放静态资源文件，所以可禁止指定扩展名程序被执行，例如：php、sh、pl、py。

### 1.4.3 限制网站来源IP访问

下面介绍如何使用ngx\_http\_access\_module限制网站来源IP访问。

案例环境：phpmyadmin数据库的web客户端，内部开发人员用的。

范例1：禁止某目录让外界访问，但允许某IP访问该目录，且支持PHP解析。  
location ~ ^/oldboy/ {

allow 202.111.12.211;

deny all;

}

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

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi\_params;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

}

}

案例环境：phpmyadmin数据库的Web客户端，内部开发人员用的。

范例2：限制及指定IP或IP段访问。

location / {

deny 192.168.1.1;

allow 192.168.1.0/24;

allow 10.1.1.0/16;

deny all;

}

参考：<http://nginx.org/en/docs/http/ngx_http_access_module.html>

**企业问题案例：nginx做反向代理的时候可以限制客户端IP吗？**

解答：可以，具体方法如下。

方法1：使用if来控制。

if ($remote\_addr = 10.0.0.7) {

return 403;

}

if ($remote\_addr = 218.247.17.130) {

set $allow\_access\_root ‘ture’;

}

http://nginx.org/en/docs/varindex.html

http://oldboy.blog.51cto.com/2561410/581383

http://oldboy.blog.51cto.com/2561410/1633952

方法2：利用deny和allow只允许IP访问。

location / {

root html/blog;

index index.php index.html index.htm;

allow 10.0.0.7;

deny all;

}

方法3：只拒绝某些IP访问。

location / {

root html/blog;

index index.php index.html index.htm;

deny 10.0.0.7;

allow all;

}

注意事项：

* deny一定要加一个IP，否则直接跳转到403，不往下执行了，如果403默认页是同一域名下，会造成死循环访问；
* 对于allow的IP端，从允许访问的段位从小到大排列，如127.0.0.0/24的下面才能是10.10.0.0/16，其中：
* 24表示子网掩码：255.255.255.0
* 16表示子网掩码：255.255.255.0
* 8表示子网掩码：255.0.0.0
* 以deny all;结尾，表示出了上面allow的，其他都禁止。

如：

deny 192.168.1.1;

allow 127.0.0.0/24;

allow 192.168.0.0./16;

allow 10.10.0.0/16;

deny all;

### 1.4.4 配置nginx禁止非法域名解析访问企业网站

问题1：nginx如何防止用户IP访问网站（恶意域名解析，也相当于是直接IP访问企业网站）。

方法1：让使用IP访问网站的用户，或者恶意解析域名的用户，收到501错误。

server {

listen 80 default\_server;

server\_name \_;

return 501;

}

说明：直接报501错误，从用户体验上不是很好。

方法2：通过301跳转到主页。

server {

listen 80 default\_server;

server\_name \_;

rewrite ^(.\*) http://blog.etiantian.org/$1 permanent;

}

方法3：发现某域名恶意解析到公司的服务器IP，在server标签里添加一下代码即可，若有多个server要多出添加。

if ($host !~^www/.eduoldboy/.com$){

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

}

提示：意思是如果header信息的host主机名字段非[www.eduoldboy.com，就301](http://www.eduoldboy.com，就301)跳转到www

## 1.5 nginx图片及目录防盗链解决方案

**1. 说明是资源盗链？**

简单的说，就是某些不法的网站未经许可，通过在其自身网站程序里非法调用其他网站的资源，然后在自己的网站上是显示这些调用的资源，达到填充自身网站的效果。这一举动不仅浪费了调用资源网站的网络流量，造成其他网站的宽带及服务压力吃紧，甚至宕机。

**2. 网站资源被盗链带来的问题**

若网站图片及相关资源被盗链，最直接的影响就是网站带宽占用加大了，带宽费用多了，网络流量也可能忽高忽低，nagios/zabbix等报警服务频繁报警。

**3. 网站资源被盗链严重问题企业真实案例**

某日，接到运维的朋友紧急求助，其公司的CDN源站，源站的流量没有变动，但CDN加速那边的流量无故超了好几个GB，不知道怎么处理。

该故障的影响：由于是购买的CDN网站加速服务，因此虽然流量多了几个GB，但是业务未受影响。只是，这么大的异常流量，持续下去可直接导致公司无故损失数万元。解决这个额问题可体现运维的价值。

**那么这样的问题如何及时发现，又如何处理呢？**

（1） 对IDC及CDN带宽做监控报警。

（2） 作为高级运维或者运维经历，每天上班的一个重要任务，就是经常查看网站流量图，关注流量变化，关注异常流量。

（3） 对访问日志做分析，对于异常流量迅速定位，并且和公司市场推广等有比较好的默契沟通交流，以便调度带宽和服务器资源。确保网站正常的访问体验得到保证。

**轻松应对IDC机房带宽突然暴涨问题**

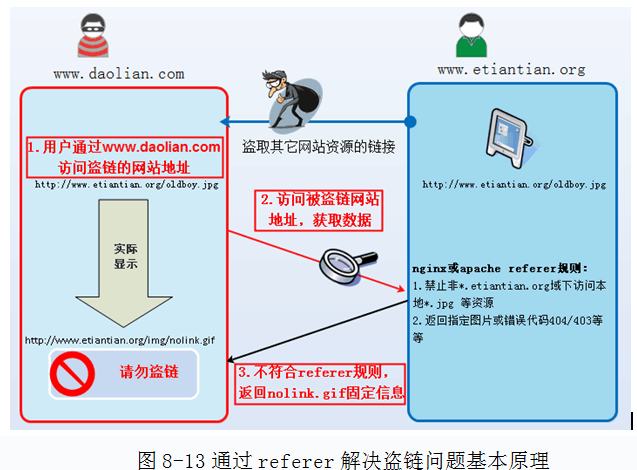
http://oldboy.blog.51cto.com/2561410/909696

**4. 常见防盗链解决方案的基本原理**

（1） 根据http referer实现防盗链

在http协议中，有一个表头字段叫referer，使用URL格式来表示是哪里的连接用了当前网页的资源。通过referer可以检测目标访问的来源网页，如果是资源文件，可以跟踪到显示它的网页地址，一旦检测出来源不是本站，马上进行阻止或返回指定的页面。

HTTP Referer是header的一部分，当浏览器向web服务器发送请求的时候，一般会带上Referer，告诉服务器我是从哪个页面链接过来的，服务器藉此可以获得一些信息用于处理。Apache、nginx、Lighttpd三者都支持根据http referer实现防盗链，referer是目前网站图片、附件、HTML等最常用的防盗链手段。下图是referer防掉链的基本原理图



（2） 根据cookie防盗链

对于一些特殊的业务数据，例如流媒体应用以及通过activeX显示的内容不向服务器提供Referer Header（例如，flash、Windows、media视频、流媒体的RTSP协议等），因为它们不向服务器提供referer header，所以若也采用上述的referer的防盗链手段就达不到想要的效果。

对于flash、Windows media视频这种占用流量较大的业务数据实现防盗链是比较困难的，此时可以采用cookie技术，来解决对flash、windows media视频等的防盗链问题。

例如：activeX插件不传递referer，但会传递cookie。可以在显示activeX的页面的<head> </head>标签内嵌入一段代码，实现用JavaScript设置一段“cookie: Cache=av”: <script> document.cookie=”Cache=av;domain=domain.com;path=/”;</script>然后就可以通过各种手段来判断这个cookie的存在以及验证其值的操作了。

（3） 通过加密变换访问路径

此种方法比较适合视频以及下载类业务数据的网站：

例如：Lighttpd有类似的插件mod\_secdownload，先在服务端配置此模块，设置一个固定用于加密的字符串，比如oldboy，然后设置一个URL前缀，比如/mp4/，再设置一个过期时间，比如1小时，然后在通过一段php代码，利用加密字符串系统时间戳和一个加密字符的MD5数值，当访问系统会对这两个数据进行验证。如果时间不在预期的时间段内，（如1小时内）则失效；如果时间戳符合条件，但是加密的字符串不符合条件也失效，从而达到防盗链的效果。

php代码实例：

<?php

$secret = “oldboy”; //加密字符串，必须跟lighttpd.conf里边保持一致。

$uri\_prefix = “/mp4/”; //虚拟的连接，前缀，必须跟lighttpd.conf里边保持一致。

$file = “/test.mp4”; //世纪文件名，必须要加“/”斜杠

$timestamp = time(); //current timestamp

$t\_hex = sprintf(“%08x”,$timestamp);

$m = md5($secret.$file.$t\_hex);

printf(‘$s’,$uri\_prefix,$m, $t\_hex, $file, $file); //生产url地址串

?>

6.

## 1.6 nginx错误页面的优雅显示

### 1.6.1 生产环境常见的http状态码列表

企业生产环境常见的http状态码列表如下表所示

|  |  |
| --- | --- |
| **状态代码及英文描述** | **代码描述** |
| 200 –OK  -Standard response for successful HTTP requests. | 服务器成功返回网页，这是成功的http请求，返回的标准状态码。 |
| 301 –Moved Permanently  -This and all future requests should be directed to the given. | 永久跳转，所有请求的网页将永久跳转到被设定的新的位置，例如：从etiantian跳转到www.etiantian.org |
| 403 –Forbidden   * forbidden request（matches a deny filter）   =>HTTP 403  - The request was a legal request,but the server is refusing to respond to it. | 禁止访问，这个请求时合法的，但是服务器端因为匹配了预先设置的规则而拒绝相应客户端的请求，此类问题一般为服务器权限配置不当所致。 |
| 404 – Not Found  -The requested resource could not be found but may be available again in the future. | 服务器找不到客户端请求的指定页面，可能是客户端请求了服务器不存在的资源导致。 |
| 500 –Internal Server Error  —internal error in haproxy => HTTP 500  —A generic error message,given when no more specific message is suitable. | 内部服务器错误，服务器遇到了意料不到的情况，不能完成客户的请求，这是一个较为笼统的报错，一般为服务器的设置或者内部程序问题导致。 |
| 502 —Bad Gateway  —the server returned an invalid or incomplete response => HTTP 502  —The server was acting as a gateway or proxy and received an invalid response from the upstream server. | 坏的网关，一般是代理服务器请求后端服务时，后端服务不可用或者没有完成相应网关服务器。一般为代理服务器下面节点处问题导致。 |
| 503 —Service Unavailable  —no server was available to handle the request =>HTTP 503  —The server is currently unavailable | 服务当前不可用，可能因为服务器超载或停机维护导致，或者是代理服务器后面没有可以提供服务的节点。 |
| 504 —Gateway Timeout  —the server failed to reply in time=> HTTP 504  —The server was acting as a gateway or proxy and did not receive a timely response from the upstream server. | 网关超时，一般是网关代理服务器请求后端服务时，后端服务没有在特定的时间内完成处理请求，一般是服务器过载导致没有在指定的时间内返回数据给代理服务器。 |

### 1.6.2 为什么要配置错误页面优雅显示？

在网站的运行过程中，可能由于页面不存在或者系统过载等原因，导致网站无法正常响应用户的请求，此时web服务默认会返回系统默认的错误码，或者很不友好的页面。

我们可以将403、404等的错误信息页面重定向到网站首页或其他事先指定的页面，提升网站的用户访问体验。

**范例1：对错误代码403实行本地页面跳转。**

###www

server {

listen 80;

server\_name [www.etiantian.org](http://www.etiantian.org);

location / {

root html/www;

index.html index.htm;

}

exxor\_page 403 /403.html; #当页面出现403错误时，会跳转到403.html页面显示给用户。

}

上面的/403.html是相对于站点根目录html/www目录的。

**范例2：对错误代码404实行本地页面优雅显示。**

server {

listen 80;

server\_name [**www.oldboyedu.com**](http://www.oldboyedu.com);

lication / {

root html/www;

index index.html index.htm;

error\_page 404 /404.html;

#当页面出现404错误时，会跳转到404.html页面显示给用户。

access\_log /app/logs/bbs\_access.log commonlog;

}

}

代码中的/404.html是相对于站点根目录html/www目录的。

**范例3:50x页面本地单独目录下文件优雅显示。**

# redirect server error pages to the static page /50x.html

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /data0/www/html;

}

这里指定单独的站点目录放到50x.html文件中。

**范例4:改变状态码为新的状态码，并显示指定的文件内容。**

error\_page 404 =200 /empty.gif;

server {

listen 80;

server\_name [www.linuxpeixun.com](http://www.linuxpeixun.com);

location / {

root /data0/www/bbs;

index index.html index.htm;

fastcgi\_intercept\_errors on;

error\_page 404 = 200 /ta.jpg;

access\_log /app/logs/bbs\_access.log commonlog;

}

}

**范例5：错误状态码url重定向**

server {

listen 80;

server\_name [www.oldboyedu.com](http://www.oldboyedu.com);

location / {

root html/www;

index index.html index.htm;

error\_page 404 <http://oldboy.blog.51cto.com>;

#当页面出现404错误时，会跳转到指定的URL <http://oldboy.blog.51cto.com>页面显示给用户，这个URL一般是企业另外的可用地址。

access\_log /app/logs/bbs\_access.log commonlog;

}

}

**范例6：将错误状态码重定向到一个location**

location / {

error\_page 404 = @fallback;

}

location @fallback {

proxy\_pass <http://backend>;

}

阿里门户网站天猫的Nginx优雅显示配置案例如下：

error\_page 500 501 502 503 504 http://err.tmall.com/error2.html;

error\_page 400 403 404 405 408 410 411 412 413 414 415 http://err.tmall.com/error1.html;

## 1.7 nginx站点目录文件及目录权限优化

**1. 单机lnmp环境目录权限严格控制措施**

为了保证网站不遭受木马入侵，所有站点目录的用户和组都应该为root，所有目录权限是755；所有文件权限是644.设置如下：

[root@lnmp ~]# ls -l /application/nginx/html/blog/|tail -5

-rw-r--r-- 1 root root 8252 Sep 3 11:33 wp-mail.php

-rw-r--r-- 1 root root 11062 Sep 3 11:33 wp-settings.php

-rw-r--r-- 1 root root 25124 Sep 3 11:33 wp-signup.php

-rw-r--r-- 1 root root 4035 Sep 3 11:33 wp-trackback.php

-rw-r--r-- 1 root root 3055 Sep 3 11:33 xmlrpc.php

以上的权限设置可以做到防止黑客上传木马，以及修改站点文件，但是，合理的网站用户上传的内容也会被拒之门外。那么如何解决可以让合法的用户传文件又不至于被黑客利用攻击呢？

**如果是单机的lnmp环境，站点目录和文件属性设置如下：**

先把所有目录权限设置755，所有文件权限设置644，所有目录和文件用户和组都是root。然后把用户上传资源权限设置755，将用户和组设置为nginx服务的用户，最后针对上传资源的目录做资源访问限制

大多数公司不是很安全的对整个站点的常见授权有如下两种：

1） chmod –R 777 /siteidr

2） chown –R nginx.nginx /sitedir

上述两种授权方法虽然不能说错误，但是没有做到授权最小化原则，会给网站带来非常大的安全隐患，特别是木马入侵。

在比较好的网站业务架构中，应把资源文件，包括用户上传的图片，附件等服务和程序服务分离，最好把上传程序服务也分离，这样就可以从容按照前面安全的标准授权了。

**2. nginx企业往后在哪集群超级安全设置**

结合Linux权限体系及nginx大型集群架构进行配置，严格控制针对nginx目录的访问。比如，可根据下面的企业集群架构逻辑图及不同角色提供的不同服务来严格控制不同服务器的nginx目录权限。

95目录—10节视频，2:55

上表为集群架构中不同于前面web业务的权限管理细化。

集群架构中不同角色的授权具体思路说明：

|  |  |  |
| --- | --- | --- |
| **服务器角色** | **权限处理** | **安全系数** |
| 动态web集群 | 目录权限755，文件权限644所用目录和文件用户和组都是root。环境nginx+php | 文件不能被改、目录不能被写入，安全系数10. |
| Static图片集群 | 目录权限755，文件权限644所用目录和文件用户和组都是root。环境nginx | 文件不能被改、目录不能被写入，安全系数10. |
| 上传upload集群 | 目录权限755，文件权限644所用目录和文件用户和组都是root。特别：用户上传难道目录设置为755，用户使用nginx服务的用户。 | 文件不能被改、目录不能被写入，但是用户上传的命令允许写入文件（这里前面通过nginx会禁止读文件）安全系统数8 |

说明：做到上述的设置，网站服务在系统层面被入侵的风险就大大降低了。

## 1.8 nginx防爬虫优化

**1. robots.txt 机器人协议介绍**

robots协议（也成为爬虫协议、机器人协议等）的全称是“网络爬虫排除标准”（robots exclusion protocol），网站通过robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取。

**2. 机器人协议八卦**

京东封杀

2001年10月25日，京东商城正式将一淘网的搜索爬虫屏蔽，以防止一淘网对其内容进行抓取，具体措施如图所示：

95目录—11节视频，2:50

淘宝封杀

2008年9月8日，淘宝网宣布封杀百度爬虫，百度忍痛遵守爬虫协议。因为一旦破坏协议，用户的隐私和利益就无法得到保障，搜索网站就谈不到人性关怀。

淘宝的robots.txt设置如下：

<http://www.taobao.com/robots.txt>

User-agent:Baiduspider

Disallow: /

User-agent:baiduspider

Disallow: /

360搜索无视robots协议

2012年8月，360综合搜索被指违反robots协议。

**3. nginx防爬虫优化配置**

我们可以根据客户端的user-agents信息，轻松地组织制定的爬虫爬取我们的网站，下面来看几个案例

范例1：组织下载协议代理。

##Block download agents ##

if ($http\_user\_agent ~\* LWP::Simple|BBBike|wget) {

return 403;

}

说明：如果用户匹配了if后面的客户端（例如wget）就返回403。

这里根据$http\_user\_agent获取客户端agent，然后判断是否允许或者返回指定错误码。

范例2：添加内容防止N多爬虫代理访问网站。

这些爬虫代理使用“|”分割，具体要处理的爬虫可以根据需求增加或减少，添加的内容如下：

if ($http\_user\_agent ~\* "qihoobot|Baiduspider|Googlebot|Googlebot-Mobile|Googlebot-Image|Mediapartners-Google|Adsbot-Google|Yahoo! Slurp China|YoudaoBot|Sosospider|Sogou spider|Sogou web spider|MSNBot") {   
     return 403;   
}

范例3：测试禁止不同的浏览器软件访问。

示例代码如下：

if ($http\_user\_agent ~\* “Firefox|MSIE”)

{

return 403;

rewrite ^(.\*) <http://blog.etiantian.org/$1> permanent;

}

如果浏览器为Firefox或者IE就会跳转到<http://blog.etiantian.org>。

## 1.9 利用nginx限制http的请求方法

最常用的HTTP方法为GET、POST，我们可以通过nginx限制http请求的方法来达到提升服务器安全的目的，例如，让HTTP智能使用GET、HEAD、和POST方法的配置如下：

#Only allow these request methods  
     if ($request\_method !~ ^(GET|HEAD|POST)$ ) {  
         return 501;  
     }

# Do not accept DELETE，SEARCH and other methods

从上传服务器传数据到存储服务器时，用户上传写入的目录不得不给nginx对应的用户权限，这样一旦程序有漏洞，木马就有可能被上传到服务器挂载的对应的存储服务器目录里，虽然我们也做了禁止PHP、SH、PL、PY等扩展名的解析限制，但还是会遗漏一些我们想不到的可执行文件，事实上，还可以通过限制上传服务器的web服务（可以具体到文件）使用GET方法，来达到防治用户通过上传服务器访问存储内容，让访问存储渠道只能从静态或图片服务器入口进入。例如：在上传服务器上限制HTTP的GET方法配置如下：

## Only allow GET request methods ##

if ($request\_method !~ ^(GET)$) {

return 501;

}

提示：还可以加一层location更具体的限制文件名。

## 1.10 基于nginx web服务Linux系统内核参数优化

**1. Linux服务器内核参数优化**

所谓内核优化，主要是指在Linux系统中针对业务服务应用而进行的系统内核参数调整，优化并无一定的标准。下面以生产环境下常见的Linux内核优化为例进行讲解。

本优化适合Apache、nginx、squid多种等web应用，特殊的业务有可能需要略作调整。

优化方法是，执行vi /ec/sysctl.conf命令到文件结尾，然后拷贝如下内容并保存。

net.ipv4.tcp\_fin\_timeout = 2

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_tw\_recycle = 1

net.ipv4.tcp\_syncookies = 1

net.ipv4.tcp\_keepalive\_time = 600

net.ipv4.ip\_local\_port\_range = 4000 65000

net.ipv4.tcp\_max\_syn\_backlog = 16384

net.ipv4.tcp\_max\_tw\_bkcuets = 36000

net.ipv4.route.gc\_timeout = 100

net.ipv4.tcp\_syn\_retries = 1

net.ipv4.tcp\_synack\_retries = 1

net.core.somaxconn = 16384

net.core.netdev\_max\_backlog = 16384

net.ipv4.tcp\_max\_orphans = 16384

#以上参数是对iptables防火墙的优化，防火墙不开会提示，可以忽略不理。

net.nf\_conntrack\_max = 25000000

net.netfilter.nf\_conntrack\_max = 25000000

net.netfilter.nf\_conntrack\_tcp\_timeout\_established = 180

net.netfilter.nf\_conntrack\_tcp\_timeout\_time\_wait = 120

net.netfilter.nf\_conntrack\_tcp\_timeout\_close\_wait = 60

net.netfilter.nf\_conntrack\_tcp\_timeout\_fin\_wait = 120

将上面的内核参数值加入/etc/sysctl.conf文件中，然后执行如下命令使之生效：

sysctl -p

如果是centos6环境下面，必须开启ip6tables服务才不会出现上面报错，其实报错也可以暂时不理，这是针对防火墙并没有开启，将来开启了就好了。

也可以通过Linux命令行执行如下命令解决上面内核优化生效的报错问题。

modprobe nf\_conntrack

echo “modprobe nf\_conntrack” >> /etc/rc.local

sysctl.conf内核文件中的参数含义

|  |  |
| --- | --- |
| **参数** | **说明** |
| net.ipv4.tcp\_fin\_timeout | 表示本端要求关闭套接字，这个参数决定了它保持在FIN-WAIT-2状态的时间，默认值是60秒。  该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_fin\_timeout60 |
| net.ipv4.tcp\_tw\_reuse | 表示开启重用。允许将TIME-WAIT sockets重新用于新的TCP连接，默认值为0，表示关闭。  该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_tw\_reuse0 |
| net.ipv4.tcp\_tw\_recycle | 表示开启TCP连接中TIME-WAIT sockets的快速回收。  该参数对应路径为：/proc/sys/net/ipv4/tcp\_tw\_recycle，默认为0，表示关闭。  提示：reuse和recycle这两个参数是为防止生产环境下web、squid等业务服务器time\_wait网络状态数量过多而设置的。 |
| net.ipv4.tcp\_syncookies | 表示开启SYN cookies功能。当出现SYN等待队列溢出时，启用cookies来处理，可防范少量SYN攻击，centos5系列默认值为1，表示开启，因此这个参数也可以不添加。  该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_syncookies，默认为1 |
| net.ipv4.tcp\_keepalive\_time | 表示当keepalive启用的时候，TCP发送keepalive消息的频度。默认是2小时，建议改为10分钟。  该参数对应系统连接为：/proc/sys/net/ipv4/tcp\_keepalive\_time，默认为7200秒。 |
| net.ipv4.ip\_local\_port\_range | 改选项用来设定允许系统打开的端口范围，即用于向外连接的端口范围。  该参数对应系统路径为：/proc/sys/net/ipvr4/ip\_local\_port\_range 32768 61000 |
| net.ipv4.tcp\_max\_syn\_backlog | 表示SYN队列的长度，默认为1024，建议加大队列的长度为8192，这样可以容纳更多等待连接的网络连接数。该参数为服务器端记录那些尚未收到客户端确认信息的连接请求的最大值。  该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_max\_syn\_backlog |
| net.ipv4.tcp\_max\_tw\_buckets | 表示系统同时保持TIME\_WAIT套接字的最大数量，如果超过这个数字，TIME\_WAIT套接字将立刻被清除并打印警告信息。默认为180000，对于Apache、nginx等服务器来说可以将其低调一点，如改为5000~30000，不同业务的服务器也可以给大一点，比如lvs、squid。  此项参数可以控制TIME\_WAIT套接字的最大数量，避免squid服务器被大量的TIME\_WAIT套接字拖死。  该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_max\_tw\_bkckets |
| net.ipv4.tcp\_synack\_retries | 参数的值决定了内核放弃连接之前发送SYN+ACK包的数量。  该参数对应系统路径为：/proc/sys/net/ipv4/tck\_synack\_retries，默认值为5 |
| net.ipv4.tcp\_syn\_retries | 表示在内核放弃建立连接之前发送SYN包的数量。  该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_syn\_retries5 |
| net.ipv4.tcp\_max\_orphans | 用于设定系统中最多有多少个TCP套接字不被关联到任何一个用户文件句柄上。如果超过这个数字，孤立连接将立即被复位并打印出警告信息。这个限制知识为了防止简单的DOS攻击。不能过分依靠这个限制甚至认为减少这个值，更多的情况是增加这个值。  该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_max\_orphans 65536 |
| net.core.somaxconn | 改选项默认值是128，这个参数用于调节系统同时发送的TCP连接数，在高并发的请求中，默认的值可能会导致链接超时或重传，因此，需要结合并发请求数来调节此值。  该参数对应系统路径为：/proc/sys/net/core/somaxconn 128 |
| net.core.netdev\_max\_backlog | 表示当每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许发送到队列的数据包最大数。  该参数对应系统路径为：/proc/sys/net/core/netdev\_max\_backlog，默认值为1000 |

1.11 使用tmpfs文件系统替代频繁访问的目录

1.11.1 使用tmpfs优化tmp等临时目录

tmpfs是一种基于内存的文件系统，使用tmpfs作为数据临时存储通常比本地磁盘快很多，此方法适用于临时使用的各类缓存场景。例如：上传图片时很多软件默认在/tmp下临时缓存切图、存放seeion数据，这是，可以让/tmp使用tmpfs文件系统来加快访问效率，此次试验将/tmp挂载到tmpfs文件系统上，让访问缓存的数据更快。具体操作方法为：

[root@db01 ~]# mount -t tmpfs -o size=16m tmpfs /tmp/

#===创建16M大小的tmpfs类型文件系统挂载到/tmp

[root@db01 ~]# df –h ##检查挂载情况

Filesystem Size Used Avail Use% Mounted on

/dev/sda3 6.9G 4.1G 2.5G 62% /

tmpfs 491M 0 491M 0% /dev/shm

/dev/sda1 190M 36M 145M 20% /boot

tmpfs 16M 0 16M 0% /tmp

[root@db01 ~]# grep eacc /proc/mounts ##检查挂载情况

tmpfs /tmp tmpfs rw,relatime,size=16384k 0 0

[root@db01 ~]# tail -1 /etc/fstab #配置永久挂载，生产场景size可以根据实际情况调整

## 1.12 使用CDN做往后在哪内容加速

1.12.1 什么是CDN？  
CDN的全称是（content delivery network），中文意思是内容分发网络。简单地讲，通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到最接近用户的cache服务器内，通过智能DNS负载均衡技术，判断用户的来源，让用户就近使用和服务器相同线路的带宽访问cache服务器取得所需的内容，例如：天津网通用户访问天津网通cache服务器上的内容，北京电信访问北京电信cache服务器上的内容。这样可以有效减少数据在网络上传输的时间，提高访问速度。

CDN是一套全国或全球的分布式缓存集群，其实质是通过智能DNS判断用户的来源地域以及上网线路，为用户选择一个最接近用户地域以及和用户上网线路相同的服务器节点，因为地域近，且线路相同，所以，可以大幅提升用户浏览网站的体验。

**CDN产生背景之一：**

BGP机房虽然可以提升用户体验，但是价格昂贵，对于用户来说，CDN的诞生可以提供比BGP机房更好的体验（让同一地区、同一线路的用户访问和当地同一线路的网站），BGP机房和普通机房有将近5～10倍的价格差。CDN多使用单线的机房，根据用户的线路以及位置，为用户选择靠近用户的位置以及相同的运营商线路，不但提升了用户体验，价格也降下来了。

CDN的价值：

* 为架设网站的企业省钱。
* 提升企业网站的用户访问体验（相同线路，相同地域，内存访问）。
* 可以阻挡大部分流量攻击，例如：DDOS攻击。

这里有一个小技巧，日100万PV的架构如何设计，这样的问题，下面就简单的给大家解答下思路：

首先应尽量把网站数据放到CDN中缓存，这样计算网站总流量总访问量减去CDN的访问流量后，剩下的访问量规模需要的架构才是我们需要设计考虑的，一个良好的网站架构设计，访问量尽量都交给CDN。

### 1.12.2 CDN的特点

CDN就是一个具备根据用户区域和线路只能调度的是分布式的内存缓存集群。

* 通过服务器内存缓存网站数据，提高了企业站点（尤其含有大量图片、视频等的站点）的访问速度，并大大升高企业站点的稳定性（省钱且用户体验提升）。
* 用户根据智能DNS技术自动选贼最适合的cache服务器，降低了不同运营商之间互联的瓶颈造成的影响，实现了跨运营商的网络加速，保证不同网络中的用户都能得到良好的访问质量。
* 加快了访问速度，减少了原站点的带宽。
* 用户访问时从服务器的内存中读取数据，分担了网络流量，同时减轻了原站点负载压力等。
* 使用CDN可以分担源站的网络流量，同时可以减轻原站点的负载压力，并降低黑客入侵以及各种DDOS攻击对网站的影响，保证网站有较好的服务质量。

下面是通过curl命令访问163网站的header信息，可以看到163网站的首页就使用了CDN进行加速。

[root@lnmp ~]# curl -I [www.163.com](http://www.163.com)

HTTP/1.1 200 0K

Expires:Sat, 25 Feb 2012 02:58:57 GMT

Date:Sat, 25 Feb 2012 02:57:37 GMT

Server:nginx

Content-Type:text/html; charset=GBK

Transfer-Encoding:chunked

Vary:Accept-Encoding

Cache-Control:max-age=80

Vary:User-Agent

Vary:Accept

Age:60

X-Via:1.1 zb200:80(Cdn cache Server V2.0),1.1 jxq206:8107(Cdn Cache Server V2.0)

Connection:keep-alive

提示：（Cdn Cache Server V2.0）这个提示表示163首页使用了CDN加速。

### 1.12.3 企业使用CDN的基本要求

首先要说的是，不是所有的网站都可以一上来就能用CDN的。

要加速的业务数据应该存在独立的域名，例如：

img1-4.etiantian.org/video1-4.etiantian.org，业务内容图片、附件JS、CSS等静态元素。

来看一个DNS解析范例。

DNS服务器加速前的A记录如下：

;A records

img.etiantian.org IN A 124.106.0.21 （企业服务器的IP）

删除上面的记录

img.etiantian.org IN A 124.106.0.21 （服务器的IP）

然后，做下面的别名解析。

; CNAME records

img1.etiantian.org IN A CNAMEbbs

img.etiantian.org 3M IN CNAME img.etiantian.org.cachecn.com.

这个img.etiantian.org.cachecn.com地址必须是事先由CDN公司配置好的CDN公司的域名

**国内较大的CDN提供商为：网宿、蓝汛、快网。**

## 1.13 nginx程序架构优化

### 1.13.1 为网站程序解耦

解耦是开发人员中流行的一个名词，简单的说就是把一堆程序代码按照业务用途分开，然后提供服务，例如：注册登录、上传、下载、浏览列表、商品内容页面、订单支付等都应该是独立的程序服务，只不过在客户端看来是一个整体而已。如果中小公司做不到上述细致的解耦，起码也要让下面的几个程序模块独立。

* 网页页面服务。
* 图片附件及下载服务。
* 上传图片服务。

上述三者的功能尽量分离。

分离的最佳方式是分别使用独立的服务器（需要改动程序），如果程序实在不好改，次选方案是在全段负载均衡器haproxy/nginx上，根据URI（例如目录或扩展名）过滤请求，然后抛给后面对应的服务器。

例如：根据扩展名分发，请求<http://www.etiantian.org/a/b.jpg>就应抛给图片服务器（独立的静态服务器最适合使用CDN）；根据URL路径分发，请求<http://www.etiantian.org/upload/index.php>就应抛给上传服务器。不符合上面两个要求的，默认都是抛给web服务器。

特别说明：可以部署3台服务器，认为分布请求服务器。当然了，这适合并发比较高服务器较多的时候。程序架构分离了，效率、安全性都会提高很多。

1.14 使用普通用户启动nginx（监牢模式）

### 1.14.1 为什么要让nginx服务使用普通用户

默认情况下，nginx的master进程使用的是root用户，worker进程使用的是nginx指定的普通用户，使用root用户跑nginx的master进程有两个最大的问题：  
（1）管理权限必须是root，这就使得最小化分配权限原则遇到难题。  
（2）使用root跑nginx服务，一旦网站出现漏洞，用户就可以很容易地获取服务器的root权限。

因此本人想出了一套不用给开发人员，甚至普通运维人员管理员权限，就可以很好的管理nginx服务。

### 1.14.2 给nginx服务降权解决方法

解决方法如下：

2）给nginx服务降权解决方案

* 给nginx服务降权，用inca用户跑nginx服务，给开发及运维设置普通账号，只要和inca同组即可管理nginx，该方案解决了nginx管理问题，防止root分配权限过大。
* 开发人员使用普通账号即可管理nginx服务以及站点下的程序、看日志。
* 采取项目负责制制度，即谁负责项目维护出问题就是谁负责。

很多公司开发和运维为争root权限不可开交，甚至大大出手的也有。

### 1.14.3 给nginx服务降权实战

本架构属架构优化（同样适合其他软件），通过nginx启动命令的-c参数指定不同的nginx配置文件，可以同时启动多个实例，并使用普通用户运行服务。

nginx安装后的启动命令路径为"/application/nginx/sbin/nginx"，可以通过加-h参数查看相关参数的用法：

[root@oldboy ~]# /application/nginx/sbin/nginx -h

nginx version: nginx/1.6.3

Usage: nginx [-?hvVtq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:

-?,-h : this help

-v : show version and exit ##显示版本号后退出

-V : show version and configure options then exit ##显示nginx版本号和哦诶之选项后退出。

-t : test configuration and exit ##测试配置文件是否正确，在运行时需要从此你加载配置的时候，此命令非常重要，用来检测所修改的配置文件是否有语法错误。

-q : suppress non-error messages during configuration testing

-s signal : send signal to a master process: stop, quit, reopen, reload ##发送信号给一个master进程，这里的reload参数很重要，是优雅重启nginx的参数，类似apache的graceful参数。

-p prefix : set prefix path (default: /application/nginx-1.6.3/)

-c filename : set configuration file (default: conf/nginx.conf) ##使用指定的配置文件而不是conf目录下的nginx.conf

-g directives : set global directives out of configuration file

较常见的方法是吧服务跑在指定用户的家目录下面，这样相对且比较安全，同时有利于批量业务部署和上线。

配置普通用户启动nginx和过程如下：

（1）添加用户宁创建相关目录和文件，操作如下：

[root@oldboy ~]# useradd inca

[root@oldboy ~]# su - inca

[inca@oldboy ~]# pwd

/home/inca

[inca@oldboy ~]# mkdir conf logs www

[inca@oldboy ~]# cp /application/nginx/conf/mime.types ~/conf/

[inca@oldboy ~]# echo inca >www/index.html

（2）配置nginx配置文件，配置后的查看命令如下：

[inca@oldboy ~]# cat conf/

mime.types nginx.conf

[inca@oldboy ~]# cat conf/nginx.conf

worker\_processes 4;

worker\_cpu\_affinity 0001 0010 0100 1000;

worker\_rlimit\_nofile 65535;

error\_log /home/inca/logs/error.log;

user inca inca;

pid /home/inca/logs/nginx.pid;

events {

use epoll;

worker\_connections 10240;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request"'

'$status $body\_bytes\_sent "$http\_referer"'

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#web.fei fa daolian......

server {

listen 8080;

server\_name www.etiantian.org;

root /home/inca/www;

location / {

index index.php index.html index.htm;

}

access\_log /home/inca/logs/web\_blog\_access.log main;

}

}

说明：

* 所有参数的值带路径都要改成/home/inca。
* 特权用户root使用的80端口，改为普通用户使用的端口在1024以上，这里8080.

（3）启动nginx，命令如下：

[inca@lnmp ~]$ ps -ef|grep nginx|grep -v gerp

[inca@lnmp ~]$ /application/nginx/sbin/nginx -c /home/inca/conf/nginx.conf &>/dev/null &

[1] 68632

[inca@lnmp ~]$ ps -ef|grep nginx|grep -v gerp

inca 68633 1 0 05:24 ? 00:00:00 nginx: master process /application/nginx/sbin/nginx -c /home/inca/conf/nginx.conf #主进程也是inca用户

inca 68634 68633 17 05:24 ? 00:00:00 nginx: worker process

inca 68635 68633 19 05:24 ? 00:00:00 nginx: worker process

inca 68636 68633 14 05:24 ? 00:00:00 nginx: worker process

inca 68637 68633 17 05:24 ? 00:00:00 nginx: worker process

inca 68639 68600 0 05:24 pts/0 00:00:00 grep nginx

[1]+ Done /application/nginx/sbin/nginx -c /home/inca/conf/nginx.conf &>/dev/null

特别提示：此处启动nginx如果不定向到空会报一些提示，不是错误，可以通过加&>/dev/null进行忽略不见。

（4） 查看访问，命令如下：

[inca@lnmp ~]$ lsof -i:8080

COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME

nginx 68633 inca 5u IPv4 170483 0t0 TCP \*:webcache (LISTEN)

nginx 68634 inca 5u IPv4 170483 0t0 TCP \*:webcache (LISTEN)

nginx 68635 inca 5u IPv4 170483 0t0 TCP \*:webcache (LISTEN)

nginx 68636 inca 5u IPv4 170483 0t0 TCP \*:webcache (LISTEN)

nginx 68637 inca 5u IPv4 170483 0t0 TCP \*:webcache (LISTEN)

（5） 解决普通端口非80提供服务的问题。

用负载均衡器解决web服务非80端口的转换问题，负载均衡器如haproxy、nginx、f5

很多公司开发和运维为争root权限不可开交，甚至大打出手的也有。  
参考资料：到底要不要给开发人员管理服务器的权限？  
http://down.51cto.com/data/844517。

本解决方案的优点如下：

* 给nginx服务降权，让网站更安全。
* 按用户设置站点权限，使站点更独立（无需虚拟化隔离）。
* 开发不需要用root即可完整管理服务机站点。
* 可实现对责任划分：网络问题属于运维的责任，网站打不开就是开发责任或共同承担。

## 1.15 控制nginx并发连接数量

ngx\_http\_limit\_conn\_module这个模块用于限制每个定义的key值的连接数。特别是单IP的连接数。

不是所有的连接数都被计数。一个符合要求的连接是由服务器处理的请求，整个请求头已经被读取。

语法：limit\_conn\_zone key zone=name:size;

上下文：http

设置共享内存取余，key可以是字符串、nginx自有变量或前两个组合，如$binary\_remote\_addr、$server\_name。name为内存区域名称，size为内存的大小。

语法:limit\_conn zone number;

上下文：http,server,location

为指定key设置最大连接数。当超过最大连接数，服务器会返回503（service Temporarily unavailable）错误。

1、 限制单IP并发连接数

nginx的配置文件如下

[root@lnmp ~]# cat /application/nginx/conf/nginx.conf

worker\_processes 1;

error\_log logs/error.log error;

events {

worker\_connections 1024;

use epoll;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

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

server\_tokens off;

server {

listen 80;

server\_name www.etiantian.org;

root html/www;

location / {

index index.php index.html index.htm;

limit\_conn addr 1; #限制单IP的并发连接为1

}

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

fastcgi\_pass 127.0.0.1:9000;

fastcgi\_index index.php;

include fastcgi.conf;

}

access\_log logs/access\_www.log main;

}

在客户端10.0.0.5使用Apache的ab测试工具进行测试。

测试1：模拟并发连接1，访问10次服务器，即执行ab –c 1 –n 10 <http://10.0.0.3/>进行测试

注：-c并发数，-n请求总数，10.0.0.3为nginx的IP地址。

测试过程中查看nginx的访问日志：

[root@lnmp extra]# tail -1 /application/nginx/logs/access.log

10.0.0.5 - - [26/Jan/2016:21:31:33 +0800] "GET / HTTP/1.1" 200 612 "-" "ApacheBench/2.3"

根据上述日志可以看出当并发为1时，返回值都是200，即访问正常。

2． 限制虚拟主机总连接数

不仅可以限制单IP的并发连接数，还可以限制虚拟主机总连接数，可以同时使用。

nginx的配置文件如下

[root@lnmp extra]# cat /application/nginx/conf/nginx.conf

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

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

limit\_conn\_zone $server\_name zone=perserver:10m;

server {

listen 80;

server\_name [www.etiantian.org](http://www.etiantian.org);

location / {

root html;

index index.html index.htm;

#limit\_conn addr 1;

limit\_conn perserver 2; #设置虚拟主机连接数为2

}

}

}

## 1.16 控制客户端请求nginx的速率

ngx\_http\_limit\_req\_module模块用于限制每个IP访问每个定义key的请求速率。

语法：limit\_req\_zone key zone=name:size rate=rate;

上下文：http

设置共享内存区域，key可以是字符串、nginx自有变量或前两个组合，如$binary\_remote\_addr。

name为内存区域的名称，size为内存区域大小，rate为速率，单位为r/s，每秒一个请求。

语法：limit\_req zone=name [burst=number] [nodelay];

上下文：http,server,location

运用了令牌桶原理，burst=num，一共有num块令牌，令牌发完后，多出来的那些请求就会返回503.换句话说，一个银行，只有一个营业员，银行很小，等候室只有5个人的位置。因此，营业员一个时刻只能为一个人提供服务，剩下的不超过5个人可以在银行内等待，超出的人直接不提供服务返回503。

nodelay默认在不超过burst值的前提天会排队等待处理，如果使用此参数，就会处理完num+1次请求，剩余的请求都视为超时，返回503.

用于测试的nginx配置文件如下：

[root@lnmp extra]# cat /application/nginx/conf/nginx.conf

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

llimit\_req\_zone $binary\_remote\_addr zone=one:10m rate=1r/s;

#以请求的客户端IP作为key值，内存区域命名为one，分配10m内存空间，访问速率限制为1秒1次请求（request）

server {

listen 80;

server\_name [www.etiantian.org](http://www.etiantian.org);

location / {

root html;

index index.html index.htm;

limit\_req zone=one burst=5;

#使用前面定义的名为one的内存空间，队列值为5，即可以有5个请求排队等待。

}

}

}