

nginx安全优化与性能优化

原创 GeorgeKai 2018-03-09 12:09:40 评论(0)

809人阅读

1.1 Nginx优化分类

安全优化（提升网站安全性配置）

性能优化（提升用户访问网站效率）

1.2 Nginx安全优化

1.2.1 隐藏nginx版本信息优化

官方配置参数说明：http://nginx.org/en/docs/http/ngx_http_core_module.html#server_tokens

官方参数：

Syntax: server_tokens on | off | build | string; #后面这俩参数收费的

Default: server_tokens on;

Context: http, server, location

配置举例：

```
[root@web01 ~]# cat /application/nginx/conf/nginx.conf
worker_processes 1;
events {
    worker_connections 1024;
}
http {
    include mime.types;
    default_type application/octet-stream;
    sendfile off;
    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"';
    server {
        listen 80;
        server_name www.nmtui.com;
        server_tokens off;
        location / {
            root html/www;
            index index.html index.htm;
        }
        access_log logs/access_www.log main;
    }
}
```

测试结果：

```
[root@web01 ~]# curl -I 10.0.0.8
HTTP/1.1 200 OK
Server: nginx
Date: Wed, 01 Nov 2017 18:32:40 GMT
Content-Type: text/html
Content-Length: 10
Last-Modified: Wed, 25 Oct 2017 01:20:56 GMT
Connection: keep-alive
ETag: "59efe6f8-a"
Accept-Ranges: bytes
```

1.2.2 修改nginx版本信息

修改版本信息需要修改程序源文件信息

修改内核信息

```
[root@web01 nginx-1.10.2]# vim src/core/nginx.h
# ...
13 #define NGINX_VERSION "6.6.6" #修改版本号
14 #define NGINX_VER "george/" NGINX_VERSION #web软件名
22 #define NGINX_VAR "georgekai"
# ...
```

修改头部信息

```
[root@web01 nginx-1.10.2]# vim src/http/ngx_http_header_filter_module.c
# ...
49 static char ngx_http_server_string[] = "Server: george" CRLF;
# ...
```

修改错误页显示

```
[root@web01 nginx-1.10.2]# vim src/http/ngx_http_special_response.c
# ...
# 此处可以不修改
21 static u_char ngx_http_error_full_tail[] =
22 "<hr><center>" NGINX_VER "</center>" CRLF
```

```

23 "</body>" CRLF
24 "</html>" CRLF
25 ;
# ...
28 static u_char ngx_http_error_tail[] =
29 "<hr><center>kai</center>" CRLF
30 "</body>" CRLF
31 "</html>" CRLF
32 ;
# ...

```

修改完成后重新编译

```

nginx -V #查看之前的编译参数
[root@web01 nginx-1.10.2]# ./configure --prefix=/application/nginx-1.10.2 --user=www --group=www --with-http_stub_status_module --with-http_ssl_module

```

```
make && make install #make失败查看下有没有pcre-devel包
```

重启服务

```
[root@web01 nginx-1.10.2]# /etc/init.d/nginx restart
```

访问测试是否修改成功

```

[root@web01 ~]# curl -I blog.george.com
HTTP/1.1 200 OK
Server: george
Date: Wed, 01 Nov 2017 19:05:43 GMT
Content-Type: text/html
Content-Length: 10
Last-Modified: Wed, 25 Oct 2017 01:20:56 GMT
Connection: keep-alive
ETag: "59efe6f8-a"
Accept-Ranges: bytes

```

1.2.3 修改worker进程的用户

第一种方法：利用编译安装配置参数，设定nginx默认worker进程用户

```

useradd -s /sbin/nologin -M www
./configure --user=www --group=www

```

第二种方式：编写nginx服务配置文件，设定nginx默认worker进程用户

官方配置参数说明：http://nginx.org/en/docs/nginx_core_module.html#user

```

Syntax:user user [group];
Default: user nobody nobody;
Context: main

```

配置举例：

```

[root@web02 conf]# cat nginx.conf
user www www; # 主区块添加user参数
worker_processes 1;
events {
worker_connections 1024;
}

```

查看是否生效

```

[root@web01 nginx-1.10.2]# ps -ef|grep nginx
root 16987 1 0 15:14 ? 00:00:00 nginx: master process nginx
clsn 18484 16987 0 15:22 ? 00:00:00 nginx: worker process
root 18486 9593 0 15:22 pts/0 00:00:00 grep --color=auto nginx

```

1.2.4 上传文件大小的限制（动态应用）

默认语法说明：

```

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

```

举例配置：

```

http {
sendfile on;
keepalive_timeout 65;
client_max_body_size 1m; # 设置上传文件最大值1M，还受博客程序限制
}

```

1.2.5 站点 Nginx站点目录及文件URL访问控制

01. 根据目录或扩展名，禁止用户访问指定数据信息

避免用户上传一些脚本，恶意破会网站程序

```

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

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

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

```

02. 当访问禁止的数据信息时，进行页面跳转

Nginx下配置禁止访问*.txt和*.doc文件。

实际配置信息如下：

```
location ~* \. (txt|doc) $ {
    if (~f $request_filename) {
        root /data/www/www;
        #rewrite ....可以重定向到某个URL
        break;
    }
}

location ~* \. (txt|doc) ${
    root /data/www/www;
    denyall;
}
```

03. 根据IP地址或网络进行访问策略控制

```
location / {
    deny 192.168.1.1;
    allow 192.168.1.0/24;
    allow 10.1.1.0/16;
    deny all;
}
```

04. 采用if判断方式，进行访问控制

```
if ($remote_addr = 10.0.0.7) {
    return 403;
}
```

1.2.6 配置Nginx，禁止非法域名解析访问企业网站

作用：避免恶意用户使用自己的域名也能解析到blog.george.com对应的IP上。

思路：只能通过域名访问网站，IP不可以访问。

第一种方式：配置一个server虚拟主机区块，放置在所有server区块最前面

```
server {
    listen 80;
    server_name - ;
    return 501;
}
```

原理：首先域名解析为IP，接下来是根据7层模型，经过2层MAC，在根据IP访问到nginx服务器，IP如果符合了在根据4层端口匹配对应的server区块，IP和端口都满足了在根据5层域名匹配对应的server区块，都不满足，默认返回给第一个server区块处理。

第二种方式：将计就计，通过你的域名访问时候，自动跳转到我的域名上

```
server {
    listen 80 default_server;
    server_name _;
    rewrite ^ (.* ) http://www.nmtui.com/$1 permanent;
}
```

第三种方式：发现某域名恶意解析到公司的服务器IP，在server标签里添加以下代码即可，若有多个server则要多处添加。

```
if ($host !~ ^www\.nmtui\.com$)
{
    rewrite ^ (.* ) http://www.nmtui.com/$1 permanent;
}
```

1.2.7 Nginx图片及目录防盗链解决方案

什么是资源盗链？

简单地说，就是某些不法网站未经许可，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显示这些调用的资源，达到填充自身网站的效果。

实现盗链过程：

01. 真正的合法网站（盗链的目标） web01 www.nmtui.com www站点目录有一个oldboy.jpg图片

在站点目录下，生成要被盗链的图片信息

```
ls blog/oldboy.jpg
```

配置静态虚拟主机 server {

```
listen 80;
server_name blog.george.com;
location / {
    root html/blog;
    index index.html index.htm;
}
```

确认生成盗链文件

02. 不合法的网站（真正盗链网站） www.daolian.com

编写盗链网站配置文件

```
listen 80;
server_name blog.daolian.com;
location / {
    root html/blog;
    index index.html index.htm;
}
```

编写一个html盗链文件

```
<html>
<head>
<title>george</title>
</head>
<body bgcolor=green>
george的博客！
```

```
<br>我的博客是
<a
href="http://blog.daollian.com" target="_blank">博客地址
</a>
          #盗取的图片
</body>
</html>
```

编写盗链虚拟主机

```
server {
listen 80;
server_name blog.daollian.com;
location / {
root html;
index index.html index.htm;
}
}
```

至此就实现了盗链。

03 常见防盗链解决方案的基本原理

1) 根据HTTP referer实现防盗链

利用referer, 并且针对扩展名rewrite重定向, 下面的代码为利用referer且针对扩展名rewrite重定向, 即实写防盗链的Nginx配置。

```
# 在站点目录下, 生成要被盗链后要跳转的图片
ls blog/nolink.jpg
```

```
~ .*\.
location ~* /\. (jpg|gif|swf|flv|wma|wmv|asf|mp3|mmf|zip|rar) $ {
root html/blog;
valid_referers none blocked *.george.com george.com;
if ($invalid_referer) {
rewrite ^/ http://blog.george.com/nolink.png;
}
}
```

注: 盗链网站盗取的是你的jpg图片, 上面被盗网站的location中匹配了jpg,但属于valid_referers, 所以会跳转为http://blog.george.com/nolink.png, 也就实现了防盗链。

如果rewrite跳转的后的图片也是jpg的图片, 那么就造成无线循环。

但是这样网站资源是实现了防盗, 但是流量还是盗取的你的, 所以使用下面的方法来实现将用户提示的图片到用户本地缓存:

设置expires的方法如下: 在location加了缓存期限

```
[root@clsn www]# cat /application/nginx/conf/extra/www.conf server {
listen 80;
server_name www.nmtui.com;
root html/www;
index index.html index.htm;
access_log logs/www_access.log main;
#Preventing hot linking of images and other file types
location ~* ^.+\. (gif|jpg|swf|flv|rar|zip) $ {
valid_referers none blocked server_names *.nmtui.comnmtui.com;
if ($invalid_referer) {
rewrite ^/ http: //www.nmtui.com/img/nolink.png;
}
access_log off;          #在location加了缓存期限
root html/www;
expires 1d;
break;
}
}
```

注: none: 正常的访问 blocked: 锁定这些访问

2) 根据cookie防盗链

3) 通过加密变换访问路径实现防盗链

通过PHP代码实现

类似的可以研究下这个模块: ngx_http_accesskey_module nginx模块

如果不怕麻烦, 有条件实现的话, 推荐使用 NginxHttpAccessKeyModule。↵

其运行方式是: 如 download 目录下有一个 file.zip 的文件。对应的 URI 是 http://www.abc.com/download/file.zip , 使用 ngx_http_accesskey_module 模块后就成了 http://www.abc.com/download/file.zipkey=09093abeac094 , 只有正确地给定了 key 值, 才能下载 download 目录下的 file.zip , 而且 key 值是 与用户的 IP 相关的, 这样就可以避免被盗链了。据说, 现在 NginxHttpAccessKeyModule 连迅雷都可以防了, 读者可以尝试一下。↵

@51CTO博客

4) 在所有网站资源上添加水印或LOGO, 让盗链人员帮你做推广宣传

1.2.8 NGINX错误页面优雅显示

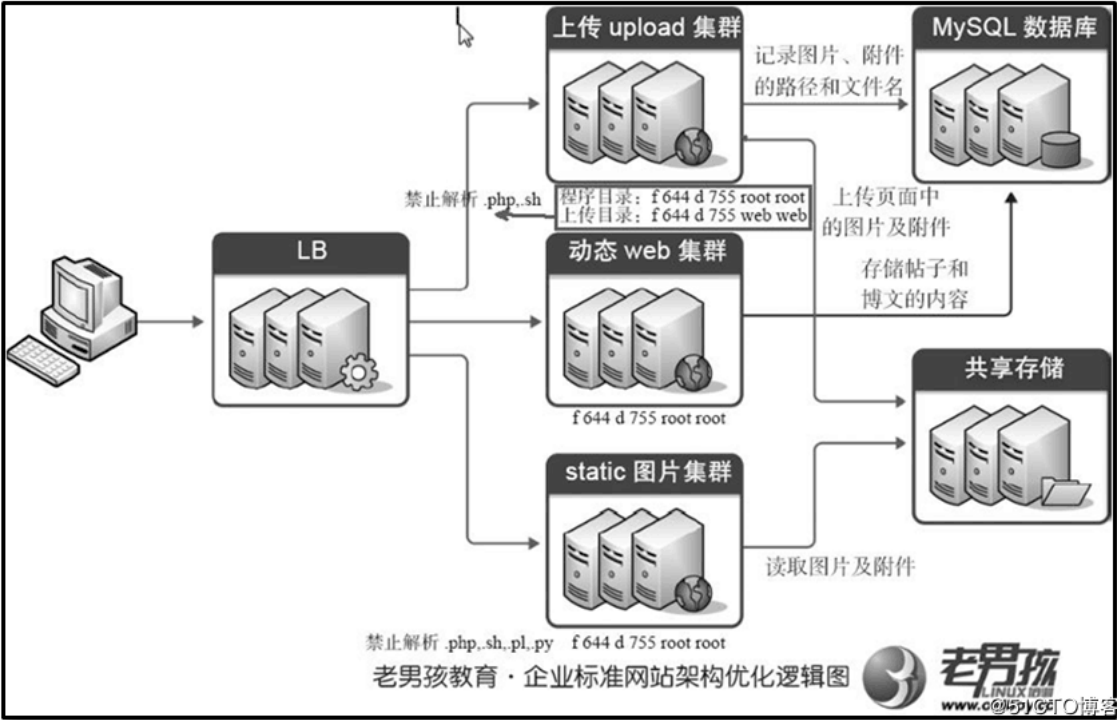
范例1: 对错误代码403实行本地页面跳转, 命令如下:

```
###www server {
listen 80;
server_name www.nmtui.com;
location / {
```

```
root html/www;
index index.html index.htm;
}
error_page 403 /403.html; #<==当出现403错误时，会跳转到403.html页面
}
# 上面的/403.html是相对于站点根目录html/www的。
范例2：50x页面放到本地单独目录下，进行优雅显示。
# 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;
}
注：多个状态码可以在一行用空格分割
范例3：改变状态码为新的状态码，并显示指定的文件内容，命令如下：
error_page 404 =200 /empty.gif;

server {
    listen 80;
    server_name www.nmtui.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4：错误状态码URL重定向，命令如下：
server {
    listen 80;
    server_name www.nmtui.com;
    location / {
        root html/www;
        index index.html index.htm;
        error_page 404 https://clsn.cnblogs.com;
        #<==当出现404错误时，会跳转到指定的URL https://clsn.cnblogs.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;
    }
}
```

1.2.9 Nginx站点目录文件及目录权限优化



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 所用的目录，以及文件用户和组都是root	环境为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 所用的目录，以及文件用户和组都是root	环境为Nginx 文件不能被改，目录 不能被写入，安全系数10
		特别：用户上传的目录设置为755， 用户和组使用Nginx服务配置的用

上传upload集群	目录权限755 文件权限644 所用的目录，以及文件用户和组都是root	户 文件不能被改，目录不能被写入， 但是用户上传的目录允许写入文件 且需要通过Nginx的其他功能来禁止 读文件，安全系数8
------------	--	--

1.2.10 Nginx防爬虫优化

最简单的方法：用wget 的某一个参数来实现爬虫

范例1：阻止下载协议代理，命令如下：

```
## Block download agents ##if ($http_user_agent ~* LWP: : Simple|BBBike|wget)
{
    return 403;
}
```

范例2：添加内容防止N多爬虫代理访问网站，命令如下：

这些爬虫代理使用"|"分隔，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;
}
```

注：如果有新的爬虫，需要查看nginx访问日志，来查询，并添加到上面配置中
注：验证码也是防爬虫的一种手段，12306的最难

1.2.11 利用Nginx限制HTTP的请求方法

#Only allow these request methods

#放在server区块下

```
if ($request_method ! ~ ^ (GET|HEAD|POST) $ ) {
    return 501;
}
```

可以自己测试下限制GET:

```
if ($request_method ~ ^ (GET) $ ) {
    return 501;
}
```

#Do not accept DELETE, SEARCH and other methods

1.2.12 使用普通用户启动nginx

1、切换到普通用户家目录下，创建nginx所需文件

```
su - george
[nginx@web01 ~]$ mkdir -p blog/{conf,logs,html}
[nginx@web01 ~]$ cd blog/
[nginx@web01 blog]$ cp /application/nginx/conf/nginx.conf.default ./conf/
[nginx@web01 blog]$ grep -vE "^$|#" conf/nginx.conf.default > conf/nginx.conf
[nginx@web01 blog]$ cp /application/nginx/conf/mime.types conf/
```

2、编写配置文件

```
[nginx@web01 ~]$ cat blog/conf/nginx.conf
worker_processes 4;
worker_cpu_affinity 0001 0010 0100 1000;
worker_rlimit_nofile 65535;
error_log /home/nginx/blog/logs/error.log; #放到上面的nginx.conf中主区块
user inca inca;
pid /home/nginx/blog/logs/nginx.pid; #放到上面的ngxinx.conf中主区块
events {
    use epoll;
    worker_connections 1024;
}
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"'; #也放到nginx.conf的http下
server {
    listen 8080; #普通用户不能用知名端口
    server_name www.etiantian.org;
    root /home/nginx/blog/html;
    location / {
        index index.html index.htm;
    }
    access_log /home/nginx/blog/logs/web_blog_access.log main; #放到nginx.conf
}
```

注意：普通用户不能使用知名端口，需要使用其他端口启动服务

大于1024的端口可以让普通用户管理。

3、检查配置文件语法，并启动nginx服务

```
/application/nginx/sbin/nginx -t -c /home/nginx/blog/conf/nginx.conf
```

或

```
/application/nginx/sbin/nginx -c /home/nginx/blog/conf/nginx.conf &>/dev/null &
```

注意：用户也需要访问8080，所以需要做一些跳转方法：

1.在负载均衡上做接收到用户的请求抛的时候抛到8080

2.防火墙NAT映射将8080映射为80

1.3 Nginx性能优化

1.3.1 优化nginx worker进程个数

nginx服务主要有两个重要进程：

- 01) **master**进程：可以控制**nginx**服务的启动 停止 或重启
- 02) **worker**进程：处理用户请求信息，帮助用户向后端服务进行请求（*php mysql*）

添加worker进程方法

```
vim nginx.conf
```

```
worker_processes 1; # 修改nginx配置文件中worker_processes指令后面的数值
```

建议：1. worker进程数量=等于CPU的核数

1. worker进程数量=等于CPU的核数*2

如何在一个系统中获悉CPU核心是多少？

- ①. 利用top命令--按数字1，获取到CPU核数信息
- ②. `grep processor /proc/cpuinfo|wc -l`
- ③. `lscpu`

查看cpu核心数命令示例

示例一

```
[root@web01 ~]# top # 按数字1
top - 03:22:48 up 9 days, 26 min, 4 users, load average: 1.06, 0.99, 0.92
Tasks: 107 total, 1 running, 106 sleeping, 0 stopped, 0 zombie
Cpu0  : 0.2%us, 0.6%sy, 0.0%ni, 99.0%id, 0.1%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu1  : 0.1%us, 0.1%sy, 0.0%ni, 99.1%id, 0.7%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 485984k total, 452536k used, 33448k free, 24984k buffers
Swap: 786428k total, 5912k used, 780516k free, 242048k cached
```

示例二

```
[root@web01 ~]# lscpu |grep CPU
CPU op-mode(s): 32-bit, 64-bit
CPU(s): 2
```

示例三

```
[root@web01 ~]# grep processor /proc/cpuinfo processor : 0
processor : 1
```

1.3.2 绑定不同的nginx进程到不同的CPU上

4个worker进程分配4核4路CPU资源方法：

```
worker_processes 4;
worker_cpu_affinity 0001 0010 0100 1000;
```

8个worker进程分配8核4路CPU资源方法；

```
worker_processes 8;
worker_cpu_affinity 0001 0010 0100 1000 0001 0010 0100 1000; # 分配8进程方法
worker_cpu_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;
```

4个worker进程分配4核2路CPU资源方法：

```
worker_processes 4;
worker_cpu_affinity 0101 1010; # 将进程分配到两颗CPU上
```

1.3.3 优化nginx事件处理模型

Nginx 的连接处理机制在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 中使用的是 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]；”。本书使用的是 CentOS 6.6 Linux，因此将 Nginx 的事件处理模型调整为 epoll 模型。

官方配置参数说明：http://nginx.org/en/docs/nginx_core_module.html#use

Syntax: use method;

Default: -

Context: events

关于事件处理模型可以参考：https://clsn.cnblogs.com/p/7750615.html#auto_id_10

举例配置：

```
user www www;
worker_processes 1;
events {
worker_connections 1024;
use epoll; --- 指定使用的模型为epoll
}
```

1.3.4 调整nginx单个进程允许的客户端最大连接数

查看nginx进程当前的打开文件数

```
[root@clsn ~]# lsof -i:80
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
nginx 10422 root 6u IPv4 11868856 0t0 TCP *:http (LISTEN)
nginx 10424 www 6u IPv4 11868856 0t0 TCP *:http (LISTEN)
nginx 10425 www 6u IPv4 11868856 0t0 TCP *:http
```

修改最大连接数方法

```
vim nginx.conf
events #<==events指令是设定Nginx的工作模式及连接数上限 {
worker_connections 1024;
}
```

说 明 : worker_connections * worker_processes <= 系统最大打开文件数量
此数值设置不要超过系统最大打开文件数量。
ulimit -HSn #查看系统最大打开文件数量
ulimit -n 65535 #修改系统最大打开文件数量

1.3.5 配置Nginx worker进程最大打开文件数

举例配置：

```
[root@web02 conf]# cat nginx.confuser www www;
worker_processes 1;
worker_rlimit_nofile 2048; # 设置worker进程打开文件数
```

1.3.6 优化nginx高效文件传输模式

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 #<==可以放置的标签段
说明：在系统内核中，利用零拷贝方式实现数据传输

实现高效数据传输的两种方式

第一种方式：tcp_nopush

syntax: tcp_nopush on | off; #<==参数语法
default: tcp_nopush off; #<==参数默认大小
context: http, server, location #<==可以放置的标签段
说明：将数据包积攒到一定量时再进行传输
参数作用：

激活或禁用Linux上的TCP_NODELAY选项。这个参数启用只在连接传输进入到 keep-alive状态。TCP_NODELAY和TCP_CORK基本上控制了包的“Nagle化”，Nagle化在这里 的含义是采用Nagle算法把较小的包组装为更大的帧。John Nagle是Nagle算法的发明人，后者就是用他的名字来命名的。

此算法解决的问题就是所谓的silly window syndrome, 中文称“愚蠢窗口症候群”，具体含义是，因为普遍终端应用程序每产生一次击键操作就会发送一个包，很轻易地就能令网络发生拥塞，Nagle化后来成了一种标准并且立即在因特网上得以实现。它现在已经成为缺省配置了，但在我们看来，有些场合下希望发送小块数据，把这一选项关掉也是合乎需要的。

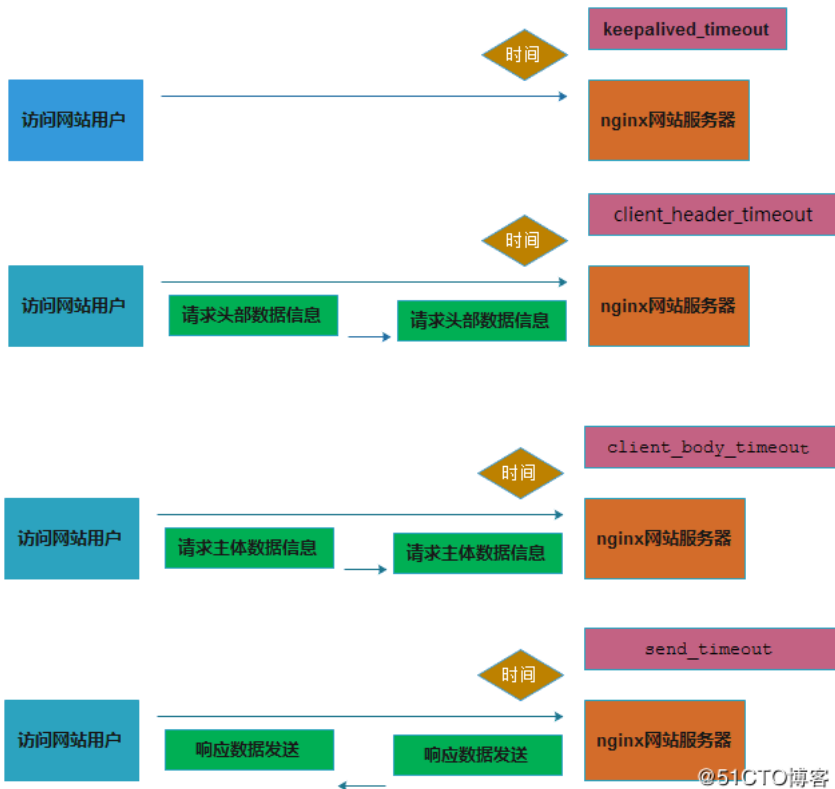
第二种方式：tcp_nodelay

Syntax: tcp_nodelay on | off;
Default: tcp_nodelay on;
Context: http, server, location
说明：只要有数据包产生，不管大小多少，就尽快传输
参数作用：

激活或禁用Linux上的TCP_CORK socket选项，tcp_cork是linux下tcp/ip传输的一个标准了，这个标准的大概的意思是，一般情况下，在tcp交互的过程中，当应用程序接收到数据包后马上发送出去，不等待，而tcp_cork选项是数据包不会马上发送出去，等到数据包最大时，一次性的传输出去，这样有助于解决网络堵塞，已经是默认了。

此选项仅仅当开启sendfile时才生效，激活这个.tcp_nopush参数可以允许把http response header和响应数据文件的开始部分放在一个文件里发布，其积极的作用是减少网络报文段的数量。

强调：两个指令是相悖的，请选择其一开启，不要同时开启；
默认采用tcp_nodelay方式进行传输。



1.3.7 设置nginx服务超时参数

Nginx连接超时的参数设置

1) 设置参数：`keepalive_timeout 60;` # 长连接才有意义

`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

说明：客户端和服务端都没有数据传输时，进行超时时间倒计时，一旦超时时间读取完毕还没有数据传输，就断开连接

2) 设置参数：`client_header_timeout 55;`

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

说明：表示定义客户端请求报文发送的间隔超时时间，客户端发送的请求报文中请求头信息的间隔时间

3) 设置参数：`client_body_timeout 55;`

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

说明：表示定义服务端响应报文发送的间隔超时时间，客户端发送的请求报文中请求主体信息的间隔时间

4) 设置参数：`send_timeout 60s`

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

说明：表示定义客户端读取服务端响应报文的间隔超时时间，服务端发送的响应报文间隔时间

1.3.8 配置Nginx gzip压缩实现性能优化

1. Nginx gzip压缩功能介绍

Nginx gzip压缩模块提供了压缩文件内容的功能，用户请求的内容在发送到用户客户端之前，Nginx服务器会根据一些具体的策略实施压缩，以节约网站出口带宽，同时加快数据传输效率，来提升用户访问体验。

2. Nginx gzip压缩的优点

提升网站用户体验：

发送给用户的内容小了，用户访问单位大小的页面就加快了，用户体验提升了，网站口碑就好了。

节约网站带宽成本：

数据是压缩传输的，因此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。

此功能既能提升用户体验，又能使公司少花钱，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。

官方用法参考链接：http://nginx.org/en/docs/http/nginx_http_gzip_module.html#gzip

```
gzip on;
gzip_min_length 1k;
gzip_buffers 4 16k;
gzip_http_version 1.1;
gzip_comp_level 4;
gzip_types text/css text/xml application/javascript;
gzip_vary on;
```

说明：将服务端响应的数据信息进行压缩，可以有效节省带宽，提高用户访问效率

gzip_type 可参考mime.conf媒体资源类型选择压缩类型。

需要和不需要压缩的对象

- 1. 纯文本内容压缩比很高，因此，纯文本的内容最好进行压缩，例如：html、js、css、xml、shtml等格式的文件。
- 2. 被压缩的纯文本文件必须要大于1KB，由于压缩算法的特殊原因，极小的文件压缩后可能反而变大。
- 3. 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。
- 4. 如果再压缩很可能不会减小或减小很少，或者有可能增大，同时压缩时还会消耗大量的CPU、内存资源。

压缩配置参数说明

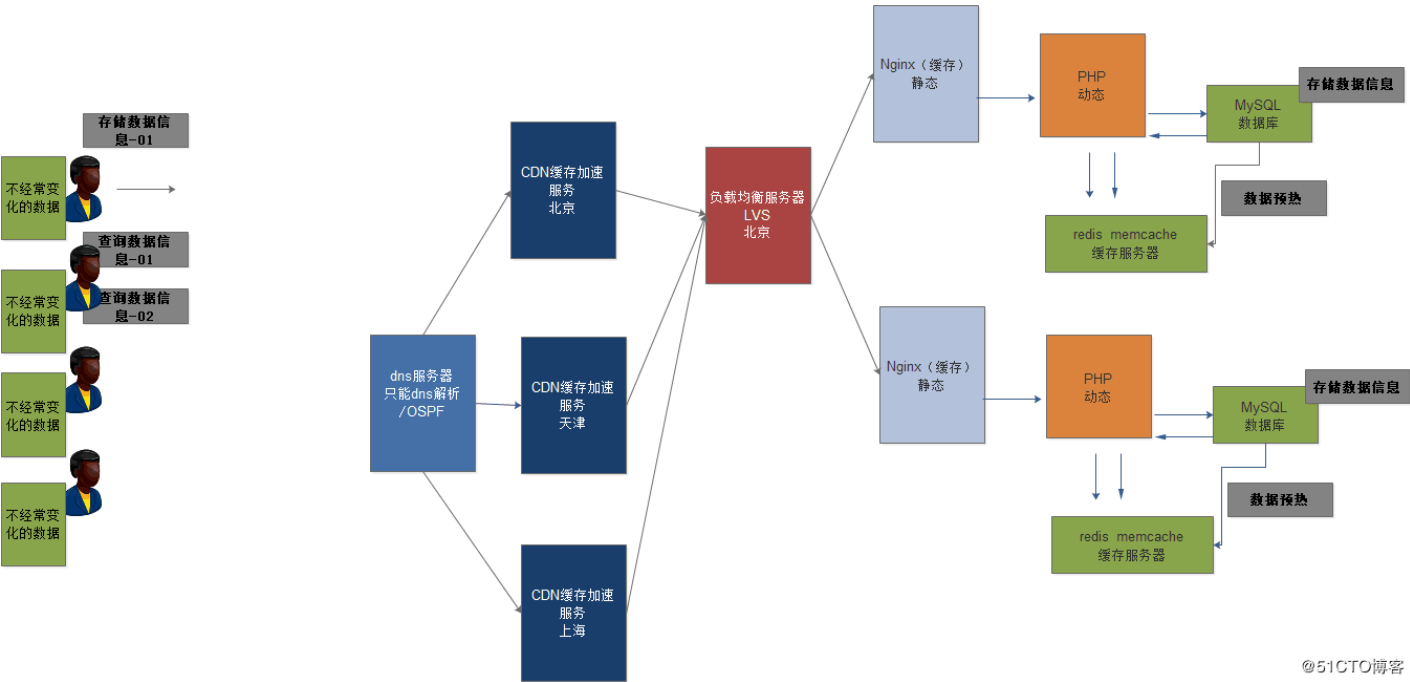
gzip on ;
#<==开启gzip压缩功能。
gzip_min_length 1k;
#<==设置允许压缩的页面最小字节数，页面字节数从header头的Content-Length中获取。默认值是0,表示不管页面多大都进行压缩。建议设置成大于1K,如果小于1K可能会越压越大。
gzip_buffers 4 16k;
#<==压缩缓冲区大小。表示申请4个单位为16K的内存作为s压缩结果流缓存，默认值是申请与原始数据大小相同的内存空间来存储gzip压缩结果。
gzip_http_version 1.1 ;
#<==压缩版本（默认1.1,前端为squid2.5时使用1.0）,用于设置识别HTTP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。 用什么版本压缩的，用什么版本解压。
gzip_comp_level 2 ;
#<==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，也比较消耗CPU资源。
gzip_types text/plain application/x-javascript text/css application/xml ;
#<==用来指定压缩的类型，"text/html"类型总是会被压缩，这个就是HTTP原理部分讲的媒体类型。
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。

1.3.9 配置Nginx expires缓存实现性能优化

简单地说，Nginx expires的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时，浏览器会检查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。

Nginx expires功能优点

- 1. expires可以降低网站的带宽，节约成本。
- 2. 加快用户访问网站的速度，提升用户访问体验。
- 3. 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。
- 4. 对于几乎所有的Web服务来说，这是非常重要的功能之一，Apache服务也有此功能。



实践配置

```
[root@web02 extra]# cat blog.conf server {
listen 80;
server_name blog.etiantian.org;
server_tokens off;
# 静态请求处理的location
location / {
root html/blog;
index index.php index.html index.htm;
}
# 动态请求处理的location
location ~* \.php$ {
root html/blog;
fastcgi_pass 127.0.0.1:9000;
fastcgi_index index.php;
include fastcgi.conf;
}
location ~* \.(gif|jpg|jpeg|png|bmp|swf)$ {
}
```

```
expires 10y;
root html/blog;
}
location ~ .*\. (js|css)$
{
expires 30d;
root html/blog;
}
}
location / {
expires 3650d;
}
```

注意事项：

01. 进行缓存的时间要设置合理
02. 不是所有请求的数据信息都可以进行缓存
万一缓存错误，将相应缓存的数据信息名称进行更改（在服务器端站点目录下进行更改）

企业网站有可能不希望被缓存的内容

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

1.3.10 配置FastCGI优化

FastCGI Cache资料见：

http://nginx.org/en/docs/http/nginx_http_fastcgi_module.html#fastcgi_cache

FastCGI常见参数的Nginx配置示例如下：

```
[root@nginx conf]# cat nginx.conf
worker_processes 4;
worker_cpu_affinity 0001 0010 0100 1000;
worker_rlimit_nofile 65535;
user nginx;
events {
    use epoll;
    worker_connections 10240;
}
http {
    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    tcp_nopush on;
    keepalive_timeout 65;
    tcp_nodelay on;
    client_header_timeout 15;
    client_body_timeout 15;
    send_timeout 15;
    log_format main '$remote_addr - $remote_user [$time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '"$http_user_agent" "$http_x_forwarded_for"';
    server_tokens off;
    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/nginx_fcgi_tmp;
    fastcgi_cache_path /data/nginx_fcgi_cache levels=2:2 keys_zone=ngx_fcgi_cache:512m inactive=1d max_size=40g;
    #web..... server {
    listen 80;
    server_name blog.nmtui.com;
    root html/blog;
    location / {
    root html/blog;
    index index.php index.html index.htm;
    }
    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;
}
access_log logs/web_blog_access.log main;
}
upstream blog_etiantian{
server 10.0.0.8:8000 weight=1;
}
server {
listen 8000;
server_name blog.nmtui.com;
location / {
proxy_pass http://blog_etiantian;
proxy_set_header Host $host;
proxy_set_header X-Forwarded-For $remote_addr;
}
access_log logs/proxy_blog_access.log main;
}
}
```

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服务器连接的超时时间，默认值为60秒，这个参数值通常不要超过75秒，因为建立的连接越多，消耗的资源就越多
fastcgi_send_timeout	设置nginx传输请求到FastCGI服务器的超时时间，这个超时时间不是整个请求的超时时间，而是两个成功请求的之间间隔时间为超时时间，如果这个时间内，FastCGI服务没有收到任何信息，连接将关闭
fastcgi_read_timeout	设置nginx从FastCGI服务器读取响应信息的超时时间来示连接建立成功后，nginx等待后端服务器的响应时间，是nginx进入后端的排队之中的等候处理的时间，实际上是读取FastCGI响应成功信息的间隔时间，
fastcgi_buffer_size	这是Nginx FastCGI的缓冲区大小参数，设定用来读取从FastCGI服务器端收到的第一部分响应信息的缓冲区大小，这里的第一部分通常会包含一个小的响应头部s默认情况下，这个参数的大小等价于_个内存页。不是4k就是8k 根据相应系统平台来决定，也可以更小。
fastcgi_buffers	设定用来读取从FastCGI服务器端收到的响应信息的缓冲区大小和缓冲区间数是，默认值为fastcgi_buffer 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，但是这并不是好方法，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PHP脚本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"16 16k", "4 64k"等
fastcgi_busy_buffers_size	用于设置系统很忙时可以使用fastcgi_buffers大小，言方推荐的大小为fastcgi_buffers*2 ;默认值为fastcgi_busy_buffers_size 8k 16k
fastcgi_temp_file_write_size	FastCGI临时文件的大小，可以设置为128-256KB ;默认fastcgi_temp_file_write_size 8k 16k;
fastcgi_cache oldboy_nginx	表示开后FastCGI缓存并为其指定一个名称。开后缓存非常有用，可以有效降低CPU的负载，并且防止502错误的发生，但是开后缓存也可能引起其它问题，要根据具体情况来选择
fastcgi_cache_path	实例：fastcgi_cache_path /data/nginx/cache levels = 2:2 keys_zone = ngx_fcgi_cache:512m inactive = ld max_size=40g; fastcgi_cache缓存目录，可以设置目录前列层级，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会直接放入内存，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，云要注意的是fastcgi_cache缓存是先写在fastcgi_temp_path在移到fastcgi_cache_path中去的，所以这个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，不过还是建议放在同_分区。
fastcgi_cache_valid	示例：fastcgi_cache_valid 200 302 lh;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1个小时；示例：fastcgi_cache_valid 301 ld;将301应答缓存1天；
fastcgi_cache_min_uses	示例：fastcgi_cache_min_uses 1; 设置清求几次之后响应将被缓存，1表示一次即被缓存
fastcgi_cache_use_stale	示例：fastcgi_cache_use_stale error timeout invalid_header http_500 定义在哪些情况下使用过期缓存
fastcgi_cache_key	示例：fastcgi_cache_key requestmethod://requestmethod://hostrequesturi;fastcgi.cache.keyhttp://requesturi;fastcgi.cache.keyhttp://host\$request_uri;定义fastcgi_cache的key ,示例中以请求的URI作为缓存的key，nginx会取这个key的md5作为缓存文件，如果设置了缓存散列目录，nginx会从后往前取相应的位数作为目录。注意一定要加作为cache key,否则如果先请求的为head 类型，后面的GET请求返回为空。

1.4

日志方面优化

1.4.1 配置Nginx服务相关日志操作

01. 进行日志的切割

```
[root@clsn ~]# mkdir /server/scripts/ -p
[root@clsn ~]# cd /server/scripts/
[root@clsn scripts]# vim cut_nginx_log.sh
#!/bin/bash
cd /application/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

说明：也可以编辑使用logrotate日志切割服务，进行日志切割

02. 进行日志的选择记录

```
如：进行访问的图片都不记录到日志
location ~ .*\. (js|jpg|JPG|jpeg|JPEG|css|bmp|gif|GIF) $ {
access_log off;
}
```

03. 进行日志文件授权

假如日志目录为/app/logs，则授权方法如下：

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

04.进程日志清理

使用定时任务定时清除没用的日志

05. 日志信息尽量汇总备份

```
[root@clsn ~]# zgrep 456 clsn.tar.gz
```

1.4.2 查看软件编译时的参数

①. 查看nginx安装时编译了哪些参数

```
/application/nginx/sbin/nginx -V
```

②. 查看apache安装时编译了哪些参数

```
cat /application/apache/build/config.nice
```

```
/application/apache/bin/apachectl -V #<--也可查看安装时编译信息，但显示的不全
```

③. 查看mysql安装时编译了哪些参数

```
grep CONFIGURE_LINE /application/mysql/bin/mysqlbug
```

PS：mysql二进制包的安装方式，是无法看到编译参数的，默认为空

④. 查看php安装时编译了哪些参数

```
/application/php/bin/php -i|grep configure
```

小伙伴们可以关注我的公众号：linux运维菜鸟之旅，阅读更方便！



版权声明：原创作品，如需转载，请注明出处。否则将追究法律责任