

Nginx 常见应用技术指南[Nginx Tips] 第二版

作者:NetSeek <http://www.linuxtone.org> (IT运维专家网|集群架构|性能调优)

欢迎转载,转载时请务必以超链接形式标明文章原始出处和作者信息及本声明.

首发时间: 2008-11-25 更新时间:2009-1-14

目 录

- 一、Nginx 基础知识
- 二、Nginx 安装及调试
- 三、Nginx Rewrite
- 四、Nginx Redirect
- 五、Nginx 目录自动加斜线:
- 六、Nginx Location
- 七、Nginx expires
- 八、Nginx 防盗链
- 九、Nginx 访问控制
- 十、Nginx 日志处理
- 十一、 Nginx Cache
- 十二、 Nginx 负载均衡
- 十三、 Nginx 简单优化
- 十四、 如何构建高性能的 LEMP 环境
- 十五、 Nginx 服务监控
- 十六、 常见问题与错误处理.
- 十七、 相关资源下载

【前言】:

编写此技术指南在于推广普及NGINX在国内的使用,更方便的帮助大家了解和掌握NGINX的一些使用技巧.本指南很多技巧来自于网络 and 工作中或网络上朋友们问我的问题.在此对网络上愿意分享的朋友们表示感谢和致意!欢迎大家和我一起丰富本技术指南提出更好的建议! 请朋友们关注: <http://www.linuxtone.org> 技术分享社区! 互想学习共同进步!

一、Nginx 基础知识

1、简介

Nginx ("engine x") 是一个高性能的 HTTP 和 [反向代理](#) 服务器,也是一个 IMAP/POP3/SMTP [代理服务](#)器。 Nginx 是由 [Igor Sysoev](#) 为俄罗斯访问量第二的 Rambler.ru 站点开发的,它已经在该站点运行超过两年半了。Igor 将源代码以类BSD许可证的形式发布。尽管还是测试版,但是, Nginx 已经因为它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名了。

更多的请见官方wiki: <http://wiki.codemongers.com/>

2、Nginx 的优点

nginx 做为 HTTP 服务器,有以下几项基本特性:

- 1) 处理静态文件,索引文件以及自动索引;打开文件描述符缓冲.

- 2) 无缓存的反向代理加速，简单的负载均衡和容错。
- 3) FastCGI，简单的负载均衡和容错。
- 4) 模块化的结构。包括 gzipping, byte ranges, chunked responses, 以及 SSI-filter 等 filter。如果由 FastCGI 或其它代理服务器处理单页中存在的多个 SSI，则这项处理可以并行运行，而不需要相互等待。
- 5) 支持 SSL 和 TLS SNI。

Nginx 专为性能优化而开发，性能是其最重要的考量，实现上非常注重效率。它支持内核 Poll 模型，能经受高负载的考验，有报告表明能支持高达 50,000 个并发连接数。

Nginx 具有很高的稳定性。其它 HTTP 服务器，当遇到访问的峰值，或者有人恶意发起慢速连接时，也很可能会导致服务器物理内存耗尽频繁交换，失去响应，只能重启服务器。例如当前 apache 一旦上到 200 个以上进程，web 响应速度就明显非常缓慢了。而 Nginx 采取了分阶段资源分配技术，使得它的 CPU 与内存占用率非常低。nginx 官方表示保持 10,000 个没有活动的连接，它只占 2.5M 内存，所以类似 DOS 这样的攻击对 nginx 来说基本上是毫无用处的。就稳定性而言，nginx 比 lighthttpd 更胜一筹。

Nginx 支持热部署。它的启动特别容易，并且几乎可以做到 7*24 不间断运行，即使运行数月也不需要重新启动。你还能够在不间断服务的情况下，对软件版本进行进行升级。

Nginx 采用 master-slave 模型，能够充分利用 SMP 的优势，且能够减少工作进程在磁盘 I/O 的阻塞延迟。当采用 select()/poll() 调用时，还可以限制每个进程的连接数。

Nginx 代码质量非常高，代码很规范，手法成熟，模块扩展也很容易。特别值得一提的是强大的 Upstream 与 Filter 链。Upstream 为诸如 reverse proxy，与其他服务器通信模块的编写奠定了很好的基础。而 Filter 链最酷的部分就是各个 filter 不必等待前一个 filter 执行完毕。它可以把前一个 filter 的输出做为当前 filter 的输入，这有点像 Unix 的管线。这意味着，一个模块可以开始压缩从后端服务器发送过来的请求，且可以在模块接收完后端服务器的整个请求之前把压缩流转向客户端。

*Nginx 采用了一些 os 提供的最新特性*如对 sendfile (Linux 2.2+), accept-filter (FreeBSD 4.1+), TCP_DEFER_ACCEPT (Linux 2.4+) 的支持，从而大大提高了性能

二、Nginx 安装及调试

1、Pcre 安装

```
./configure
make && make install
cd ../
```

2. nginx 编译安装

```
/configure --user=www --group=www --prefix=/usr/local/nginx/ --with-http_stub_status_module
--with-openssl=/usr/local/openssl
make && make install
```

更详细的模块定制与安装请参照官方 wiki.

3、Nginx 配置文件测试:

```
# /usr/local/nginx/sbin/nginx -t //Debug 配置文件的关键命令需要重点掌握.
```

```
2008/12/16 09:08:35 [info] 28412#0: the configuration file /usr/local/nginx/conf/nginx.conf
syntax is ok
2008/12/16 09:08:35 [info] 28412#0: the configuration file /usr/local/nginx/conf/nginx.conf was
tested successfully
```

3、Nginx 启动:

```
# /usr/local/nginx/sbin/nginx
```

4、Nginx 配置文件修改重新加载:

```
# kill -HUP `cat /usr/local/nginx/logs/nginx.pid`
```

、

三、Nginx Rewrite

1. Nginx Rewrite 基本标记(flags)

last - 基本上都用这个 Flag。

※相当于 Apache 里的[L]标记, 表示完成 rewrite, 不再匹配后面的规则

break - 中止 Rewrite, 不再继续匹配

redirect - 返回临时重定向的 HTTP 状态 302

permanent - 返回永久重定向的 HTTP 状态 301

※原有的 url 支持正则 重写的 url 不支持正则

2. 正则表达式匹配, 其中:

```
* ~      为区分大小写匹配
* ~*     为不区分大小写匹配
* !~和!~* 分别为区分大小写不匹配及不区分大小写不匹配
```

3. 文件及目录匹配, 其中:

```
* -f 和!-f 用来判断是否存在文件
* -d 和!-d 用来判断是否存在目录
* -e 和!-e 用来判断是否存在文件或目录
* -x 和!-x 用来判断文件是否可执行
```

3. Nginx 的一些可用的全局变量, 可用做条件判断:

```
$args
$content_length
```

```
$content_type
$document_root
$document_uri
$host
$http_user_agent
$http_cookie
$limit_rate
$request_body_file
$request_method
$remote_addr
$remote_port
$remote_user
$request_filename
$request_uri
$query_string
$scheme
$server_protocol
$server_addr
$server_name
$server_port
$uri
```

四、Nginx Redirect

将所有linuxtone.org与netseek.linuxtone.org域名全部自跳转到<http://www.linuxtone.org>

```
server
{
    listen      80;
    server_name linuxtone.org netseek.linuxtone.org;
    index index.html index.php;
    root /data/www/wwwroot;
    if ($host !~ "^www\.linuxtone\.org$") {
        rewrite ^(.*) http://www.linuxtone.org$1 redirect;
    }

    .....
}
```

五、Nginx 目录自动加斜线:

```
if (-d $request_filename){
    rewrite ^/(.*)([^\/]$) http://$host/$1$2/ permanent;
}
```

六、Nginx Location

1. 基本语法:[和上面 rewrite 正则匹配语法基本一致]

```
location [=|~|~*|^~] /uri/ { ... }
```

* ~ 为区分大小写匹配

* ~* 为不区分大小写匹配

* !~和!~*分别为区分大小写不匹配及不区分大小写不匹配

示例 1:

```
location = / {  
    # matches the query / only.  
    # 只匹配 / 查询。  
}
```

匹配任何查询，因为所有请求都已 / 开头。但是正则表达式规则和长的块规则将被优先和查询匹配

示例 2:

```
location ^~ /images/ {  
    # matches any query beginning with /images/ and halts searching,  
    # so regular expressions will not be checked.
```

匹配任何已 /images/ 开头的任何查询并且停止搜索。任何正则表达式将不会被测试。

示例 3:

```
location ~* \.(gif|jpg|jpeg)$ {  
    # matches any request ending in gif, jpg, or jpeg. However, all  
    # requests to the /images/ directory will be handled by  
}
```

匹配任何已 gif、jpg 或 jpeg 结尾的请求。

七、Nginx expires

1. 根据文件类型 expires

```
# Add expires header for static content  
location ~* \.(js|css|jpg|jpeg|gif|png|swf)$ {  
    if (-f $request_filename) {  
        root /data/www/wwwroot/bbs;  
        expires      1d;  
        break;  
    }  
}
```

2、根据判断某个目录

```
# serve static files
location ~ ^/(images|javascript|js|css|flash|media|static)/ {
    root    /data/www/wwwroot/down;
    expires 30d;
}
```

八、Nginx 防盗链

1. 针对不同的文件类型

```
#Preventing hot linking of images and other file types
location ~* ^.+\. (gif|jpg|png|swf|flv|rar|zip)$ {
    valid_referers none blocked server_names *.linuxtone.org linuxtone.org http://localhost
    baidu.com;
    if ($invalid_referer) {
        rewrite ^/ http://www.linuxtone.org/images/default/logo.gif;
        # return 403;
    }
}
```

2. 针对不同的目录

```
location /img/ {
    root /data/www/wwwroot/bbs/img;
    valid_referers none blocked server_names *.linuxtone.org http://localhost baidu.com;
    if ($invalid_referer) {
        rewrite ^/ http://www.linuxtone.org/images/default/logo.gif;
        #return 403;
    }
}
```

3. 同实现防盗链和 expires 的方法

```
#Preventing hot linking of images and other file types
location ~* ^.+\. (gif|jpg|png|swf|flv|rar|zip)$ {
    valid_referers none blocked server_names *.linuxtone.org linuxtone.org http://localhost ;
    if ($invalid_referer) {
        rewrite ^/ http://www.linuxtone.org/images/default/logo.gif;
    }
    access_log off;
    root /data/www/wwwroot/bbs;
    expires 1d;
    break;
}
```

九、Nginx 访问控制

1. Nginx 身份验证

```
#cd /usr/local/nginx/conf
#mkdir httpasswd
/usr/local/apache2/bin/htpasswd -c /usr/local/nginx/conf/httpasswd/tongji linuxtone
#添加用户名为 linuxtone
New password:    (此处输入你的密码)
Re-type new password:  (再次输入你的密码)
Adding password for user
```

<http://count.linuxtone.org/tongji/data/index.html>(目录存在/data/www/wwwroot/tongji/data/目录下)

将下段配置放到虚拟主机目录, 当访问 <http://count.linuxtone.org/tongji/>即提示要密验证:

```
location ~ ^/(tongji)/ {
    root    /data/www/wwwroot/count;
    auth_basic "LT-COUNT-TongJi";
    auth_basic_user_file /usr/local/nginx/conf/httpasswd/tongji;
}
```

2. Nginx 禁止访问某类型的文件

如, Nginx 下禁止访问*.txt 文件, 配置方法如下.

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

方法 2:

```
location ~* \.(txt|doc)$ {
    root /data/www/wwwroot/linuxtone/test;
    deny all;
}
```

实例:

禁止访问某个目录

```
location ~ ^/(WEB-INF)/ {
    deny all;
}
```

3. 使用 ngx_http_access_module 限制ip 访问

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

详细参见wiki: <http://wiki.codemongers.com/NginxHttpAccessModule#allow>

4. Nginx 下载限制并发和速率

```
limit_zone   linuxtone  $binary_remote_addr 10m;  
server  
{  
    listen      80;  
    server_name down.linuxtone.org;  
    index index.html index.htm index.php;  
    root        /data/www/wwwroot/down;  
    #Zone limit  
    location / {  
        limit_conn  linuxtone 1;  
        limit_rate  20k;  
    }  
    .....  
}
```

只允许客户端一个线程,每个线程 20k.

【注】`limit_zone linuxtone $binary_remote_addr 10m;` 这个可以定义在主的

5. Nginx 实现 Apache 一样目录列表

```
location / {  
    autoindex on;  
}
```

6. 上文件大小限制

主配置文件里加入如下, 具体大小根据你自己的业务做调整。

```
client_max_body_size 10m;
```

十、Nginx 日志处理

1.Nginx 日志切割

```
#crontab -e
```



```
59 23 * * * /usr/local/sbin/logcron.sh /dev/null 2>&1
```

```
[root@count ~]# cat /usr/local/sbin/logcron.sh
```

```
#!/bin/bash
log_dir="/data/logs"
time=`date +%Y%m%d`
/bin/mv ${log_dir}/access_linuxtone.org.log ${log_dir}/access_count.linuxtone.org.$time.log
kill -USR1 `cat /var/run/nginx.pid`
```

更多的日志分析与处理就关注(同时欢迎你参加讨论):<http://bbs.linuxtone.org/forum-8-1.html>

2. 利用AWSTATS 分析NGINX 日志

设置好 Nginx 日志格式,仍后利用 awstats 进行分析.

请参考: <http://bbs.linuxtone.org/thread-56-1-1.html>

3. Nginx 如何不记录部分日志

日志太多, 每天好几个 G, 少记录一些, 下面的配置写到 server{ }段中就可以了

```
location ~ .*\. (js|jpg|JPG|jpeg|JPEG|css|bmp|gif|GIF)$
{
    access_log off;
}
```

十一、Nginx Cache 服务配置

如果需要将文件缓存到本地, 则需要增加如下几个子参数:

```
proxy_store on;
proxy_store_access user:rw group:rw all:rw;
proxy_temp_path 缓存目录;
```

其中,

proxy_store on 用来启用缓存到本地的功能,

proxy_temp_path 用来指定缓存在哪个目录下, 如: proxy_temp_path html;

在经过上一步配置之后, 虽然文件被缓存到了本地磁盘上, 但每次请求仍会向远端拉取文件, 为了避免去远端拉取文件, 必须修改 proxy_pass:

```
if ( !-e $request_filename) {
    proxy_pass http://mysvr;
}
```

即改成有条件地去执行 proxy_pass, 这个条件就是当请求的文件在本地的 proxy_temp_path 指定的目录下不存在时, 再向后端拉取。

更多更高级的应用可以研究ncache, 官方网站: <http://code.google.com/p/ncache/>

详细安装请参照<http://bbs.linuxtone.org> 应用加速版ncache相关的帖子.

十二、Nginx 负载均衡

1. Nginx 负载均衡基础知识

nginx 的 upstream 目前支持 4 种方式的分配

1)、轮询（默认）

每个请求按时间顺序逐一分配到不同的后端服务器，如果后端服务器 down 掉，能自动剔除。

2)、weight

指定轮询几率，weight 和访问比率成正比，用于后端服务器性能不均的情况。

2)、ip_hash

每个请求按访问 ip 的 hash 结果分配，这样每个访客固定访问一个后端服务器，可以解决 session 的问题。

3)、fair（第三方）

按后端服务器的响应时间来分配请求，响应时间短的优先分配。

4)、url_hash（第三方）

2. Nginx 负载均衡实例 1

```
upstream bbs.linuxtone.org {#定义负载均衡设备的 Ip 及设备状态
    server 127.0.0.1:9090 down;
    server 127.0.0.1:8080 weight=2;
    server 127.0.0.1:6060;
    server 127.0.0.1:7070 backup;
}
```

在需要使用负载均衡的 server 中增加

```
proxy_pass http://bbs.linuxtone.org/;
```

每个设备的状态设置为：

- a) down 表示单前的 server 暂时不参与负载
- b) weight 默认为 1.weight 越大，负载的权重就越大。
- c) max_fails：允许请求失败的次数默认为 1.当超过最大次数时，返回 proxy_next_upstream 模块定义的错误
- d) fail_timeout:max_fails 次失败后，暂停的时间。
- e) backup：其它所有的非 backup 机器 down 或者忙的时候，请求 backup 机器。所以这台机器压力会最轻。

nginx 支持同时设置多组的负载均衡，用来给不同的 server 来使用。

client_body_in_file_only 设置为 On 可以讲 client post 过来的数据记录到文件中用来做 debug

client_body_temp_path 设置记录文件的目录 可以设置最多 3 层目录

location 对 URL 进行匹配,可以进行重定向或者进行新的代理 负载均衡

3. Nginx 负载均衡实例 2

按访问 url 的 hash 结果来分配请求，使每个 url 定向到同一个后端服务器，后端服务器为缓存时比较有效,也可以用作提高 Squid 缓存命中率。

简单的负载均衡实例:

#vi nginx.conf //nginx 主配置文件核心配置

```
.....
#loadbalance my.linuxtone.org
    upstream my.linuxtone.org {
        ip_hash;
        server 127.0.0.1:8080;
        server 192.168.169.136:8080;
        server 219.101.75.138:8080;
        server 192.168.169.117;
        server 192.168.169.118;
        server 192.168.169.119;
    }
.....
include vhosts/linuxtone_lb.conf;
.....
```

vi proxy.conf

```
proxy_redirect off;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
client_max_body_size 50m;
client_body_buffer_size 256k;
proxy_connect_timeout 30;
proxy_send_timeout 30;
proxy_read_timeout 60;

proxy_buffer_size 4k;
proxy_buffers 4 32k;
proxy_busy_buffers_size 64k;
proxy_temp_file_write_size 64k;
proxy_next_upstream error timeout invalid_header http_500 http_503 http_404;
proxy_max_temp_file_size 128m;
proxy_store on;
proxy_store_access user:rw group:rw all:r;
#nginx cache
#client_body_temp_path /data/nginx_cache/client_body 1 2;
proxy_temp_path /data/nginx_cache/proxy_temp 1 2;
```

#vi linuxtone_lb.conf

```
server
{
    listen 80;
```

```
server_name my.linuxtone.org;
index index.php;
root /data/www/wwwroot/mylinuxtone;
if (-f $request_filename) {
    break;
}
if (-f $request_filename/index.php) {
    rewrite (.*?) $1/index.php break;
}

error_page 403 http://my.linuxtone.org/member.php?m=user&a=login;
location / {
    if ( !-e $request_filename) {
        proxy_pass http://my.linuxtone.org;
        break;
    }
    include /usr/local/nginx/conf/proxy.conf;
}
}
```

十三、Nginx 简单优化

1. 减小 nginx 编译后的文件大小 (Reduce file size of nginx)

默认的 nginx 编译选项里居然是用 debug 模式(-g)的 (debug 模式会插入很多跟踪和 ASSERT 之类), 编译以后一个 nginx 有好几兆。去掉 nginx 的 debug 模式编译, 编译以后只有几百 K

在 auto/cc/gcc, 最后几行有:

```
# debug
CFLAGS="$CFLAGS -g"
```

注释掉或删除这几行, 重新编译即可。

2. 修改 Nginx 的 header 伪装服务器

1) 修改 nginx.h

```
#vi nginx-0.7.30/src/core/nginx.h
#define NGINX_VERSION    "1.8"
#define NGINX_VER        "LTWS/" NGINX_VERSION

#define NGINX_VAR        "NGINX"
#define NGX_OLDPID_EXT   ".oldbin"
```

2) 修改 ngx_http_header_filter_module

```
#vi nginx-0.7.30/src/http/ngx_http_header_filter_module.c
```

将如下

```
static char ngx_http_server_string[] = "Server: nginx" CRLF;
```

修改为

```
static char ngx_http_server_string[] = "Server: LTWS" CRLF;
```

- a) 修改 ngx_http_header_filter_module
#vi nginx-0.7.30/src/http/ngx_http_special_response.c
将如下:

```
static u_char ngx_http_error_full_tail[] =  
"<hr><center> NGINX_VER </center>" CRLF  
"</body>" CRLF  
"</html>" CRLF  
;
```

```
static u_char ngx_http_error_tail[] =  
"<hr><center>nginx</center>" CRLF  
"</body>" CRLF  
"</html>" CRLF  
;
```

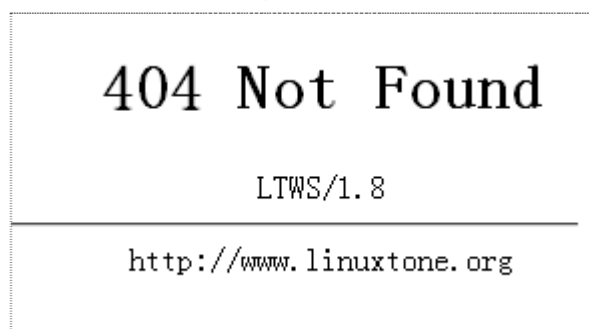
修改为:

```
static u_char ngx_http_error_full_tail[] =  
"<center> "NGINX_VER" </center>" CRLF  
"<hr><center>http://www.linuxtone.org</center>" CRLF  
"</body>" CRLF  
"</html>" CRLF  
;
```

```
static u_char ngx_http_error_tail[] =  
"<hr><center>LTWS</center>" CRLF  
"</body>" CRLF  
"</html>" CRLF  
;
```

修改后重新编译一下环境,

404 错误的时候显示效果图 (如果没有指定错误页的话):



利用 curl 命令查看服务器 header

```
[root@monitoring ~]# curl -I http://221.122.255.45/
HTTP/1.1 200 OK
Server: LTWS/1.8
Date: Wed, 14 Jan 2009 14:46:26 GMT
Content-Type: text/html; charset=gb2312
Connection: keep-alive
Vary: Accept-Encoding
```

3. 为特定的 CPU 指定 CPU 类型编译优化.

默认 nginx 使用的 GCC 编译参数是 -O

需要更加优化可以使用以下两个参数

--with-cc-opt='-O3' \

--with-cpu-opt=opteron \

使得编译针对特定 CPU 以及增加 GCC 的优化.

此方法仅对性能有所改善并不会有很大的性能提升, 供朋友们参考.

CPU 类型确定: # cat /proc/cpuinfo | grep "model name"

编译优化参数参考: http://en.gentoo-wiki.com/wiki/Safe_Cflags

4. Tcmalloc 优化 Nginx 性能

```
# wget http://download.savannah.gnu.org/releases/libunwind/libunwind-0.99-alpha.tar.gz
```

```
# tar zxvf libunwind-0.99-alpha.tar.gz
```

```
# cd libunwind-0.99-alpha/
```

```
# CFLAGS=-fPIC ./configure
```

```
# make CFLAGS=-fPIC
```

```
# make CFLAGS=-fPIC install
```

```
# wget http://google-perftools.googlecode.com/files/google-perftools-0.98.tar.gz
```

```
# tar zxvf google-perftools-0.98.tar.gz
```

```
# cd google-perftools-0.98/
```

```
# ./configure
```

```
# make && make install
```

```
# echo "/usr/local/lib" > /etc/ld.so.conf.d/usr_local_lib.conf
```

```
# ldconfig
```

```
# lsof -n | grep tcmalloc
```

编译 nginx 加载 google_perftools_module:

```
./configure --with-google_perftools_module
```

在主配置文件加入 nginx.conf 添加:

```
google_perftools_profiles /path/to/profile;
```

5. 内核参数优化

```
# vi /etc/sysctl.conf
```

```
#在末尾增加以下内容:
```

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_time = 300
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.ip_local_port_range = 5000    65000
```

```
#使配置立即生效
```

```
/sbin/sysctl -p
```

十四、如何构建高性的 LEMP

请参见: <http://www.linuxtone.org/lemp/lemp.pdf>

- 1、提供完整的配置脚本下载: <http://www.linuxtone.org/lemp/scripts.tar.gz>
 - 2、提供 NGINX 常见配置范例含(虚拟主机, 防盗链, Rewrite, 访问控制, 负载均衡 Discuz 相关程序静态化及等等), 你只要稍稍修改即可线上应用。
 - 3、将原版的 xcache 替换成 EA, 并提供相关简单调优脚本及配置文件。
- 更多的及更新资料请关注: <http://www.linuxtone.org>

十五、Nginx 监控

1、RRDTOOL+Perl 脚本画图监控

先安装好 rrdtool, 关于 rrdtool 本文不作介绍, 具体安装请参照 linuxtone 监控版块.

```
#cd /usr/local/sbin
#wget http://blog.kovyrim.net/files/mrtg/rrd\_nginx.pl.txt
#mv rrd_nginx.pl.txt rrd_nginx.pl
#chmod a+x rrd_nginx.pl
```

```
#vi rrd_nginx.pl //配置脚本文件设置好路径
```

```
#!/usr/bin/perl
use RRDs;
use LWP::UserAgent;

# define location of rrdtool databases
my $rrd = '/data/www/wwwroot/nginx/rrd';
# define location of images
my $img = '/data/www/wwwroot/nginx/html';
# define your nginx stats URL
my $URL = "http:// 219.32.205.13/nginx_status";
.....
```

【注】根据自己具体的状况修改相应的路径.

#crontab -e //加入如下

```
***** /usr/local/sbin/rrd_nginx.pl
```

重启 crond 后,通过配置 nginx 虚拟主机指到/data/www/wwwroot/nginx/html 目录,通过 crond 自动执行 perl 脚本会生成很多图片.

<http://xxx/connections-day.png>即可看到服务器状态图。

2、官方 Nginx-rrd 监控服务（多虚拟主机）（推荐）

网址: <http://www.nginx.eu/nginx-rrd.html>

此解决方案其实是基于上述监控方案的一个改进和增强,同样先安装好 rrdtool 这个画图工具和相应的 perl 模块再做如下操作:

```
# yum install perl-HTML*
```

先建立好生成的库存和图片存放录

```
#mkdir -p /data/www/wwwroot/nginx/{rrd,html}

#cd /usr/local/sbin
#wget http://www.nginx.eu/nginx-rrd/nginx-rrd-0.1.4.tgz
#tar zxvf nginx-rrd-0.1.4.tgz
#cd nginx-rrd-0.1.4
#cd etc/
#cp nginx-rrd.conf /etc
#cd etc/cron.d
#cp nginx-rrd.cron /etc/cron.d

#cd /usr/local/src/nginx-rrd-0.1.4/html
# cp index.php /data/www/wwwroot/nginx/html/

#cd /usr/local/src/nginx-rrd-0.1.4/usr/sbin
#cp * /usr/sbin/
```

```
#vi /etc/nginx-rrd.conf
```

```
#####
#
# dir where rrd databases are stored
RRD_DIR="/data/www/wwwroot/nginx/rrd";
# dir where png images are presented
WWW_DIR="/data/www/wwwroot/nginx/html";
# process nice level
NICE_LEVEL="-19";
# bin dir
BIN_DIR="/usr/sbin";
# servers to test
# server_utl;server_name
SERVERS_URL="http://219.32.205.13/nginx\_status;219.32.205.13
```



```
http://www.linuxtone.org/nginx_status;www.linuxtone.org"" //根据你的具体情况做调整.
```

SEVERS_URL 格式 http://domain1/nginx_status;domain1 http://domain2/nginx_status;domain2

这种格式监控多虚拟主机连接状态:

重点启crond服务, 仍后通过<http://219.32.205.13/nginx/html/> 即可访问。配置过程很简单!

3、CACTI 模板监控 Nginx

利用 Nginx_status 状态来画图实现 CACTI 监控

nginx 编译时允许 http_stub_status_module

```
# vi /usr/local/nginx/conf/nginx.conf
```

```
location /nginx_status {  
    stub_status on;  
    access_log off;  
    allow 192.168.1.37;  
    deny all;  
}
```

```
# kill -HUP `cat /usr/local/nginx/logs/nginx.pid`
```

```
# wget http://forums.cacti.net/download.php?id=12676
```

```
# tar xvfz cacti-nginx.tar.gz
```

```
# cp cacti-nginx/get_nginx_socket_status.pl /data/cacti/scripts/
```

```
# cp cacti-nginx/get_nginx_clients_status.pl /data/cacti/scripts/
```

```
# chmod 755 /data/cacti/scripts/get_nginx*
```

检测插件

```
# /data/cacti/scripts/get_nginx_clients_status.pl http://192.168.1.37/nginx_status
```

在 cacti 管理面板导入

```
cacti_graph_template_nginx_clients_stat.xml
```

```
cacti_graph_template_nginx_sockets_stat.xml
```

十六、常见问题与错误处理

1、400 bad request 错误的原因和解决办法

配置 nginx.conf 相关设置如下.

```
client_header_buffer_size 16k;
```

```
large_client_header_buffers 4 64k;
```

根据实际情况调整, 一般适当调整值就可以。

2、Nginx 502 Bad Gateway 错误

```
proxy_next_upstream error timeout invalid_header http_500 http_503;
```

或者尝试设置:

```
large_client_header_buffers 4 32k;
```

3、Nginx 出现的 413 Request Entity Too Large 错误

这个错误一般在上传文件的时候会出现，
编辑 Nginx 主配置文件 Nginx.conf，找到 http{} 段，添加
`client_max_body_size 10m;` //设置多大根据自己的需求作调整。

如果运行 php 的话这个大小 client_max_body_size 要和 php.ini 中的如下值的最大值一致或者稍大，这样就不会因为提交数据大小不一致出现的错误。

```
post_max_size = 10M
upload_max_filesize = 2M
```

4、解决 504 Gateway Time-out(nginx)

遇到这个问题是在升级 discuz 论坛的时候遇到的

一般看来，这种情况可能是由于 nginx 默认的 fastcgi 进程响应的缓冲区太小造成的，这将导致 fastcgi 进程被挂起，如果你的 fastcgi 服务对这个挂起处理的不好，那么最后就极有可能导致 504 Gateway Time-out

现在的网站，尤其某些论坛有大量的回复和很多内容的，一个页面甚至有几百 K。
默认的 fastcgi 进程响应的缓冲区是 8K，我们可以设置大点
在 nginx.conf 里，加入：`fastcgi_buffers 8 128k`

这表示设置 fastcgi 缓冲区为 8×128k

当然如果您在进行某一项即时的操作，可能需要 nginx 的超时参数调大点，例如设置成 60 秒：`send_timeout 60;`

只是调整了这两个参数，结果就是没有再显示那个超时，可以说效果不错，但是也可能是由于其他的原因，目前关于 nginx 的资料不是很多，很多事情都需要长期的经验累计才有结果，期待您的发现哈！

5、如何使用 Nginx Proxy

朋友一台服务器运行 tomcat 为 8080 端口,IP:192.168.1.2:8080,另一台机器 IP:192.168.1.8. 朋友想通过访问<http://192.168.1.8> 即可访问 tomcat 服务.配置如下:
在 192.168.1.8 的 nginx.conf 上配置如下:

```
server {
    listen 80;
    server_name java.linuxtone.org

    location / {
        proxy_pass http://192.168.1.2:8080;
        include /usr/local/nginx/conf/proxy.conf;
    }
}
```

6、如何关闭 Nginx 的 LOG

```
access_log /dev/null;  
error_log /dev/null;
```

十七、相关资源下载

1. nginx 配置示例及脚本下载:

wget <http://www.linuxtone.org/lemp/scripts.tar.gz> #此脚本范例定期更新.

【附】:

本文档定期更新,同时欢迎朋友多提宝贵意见, 丰富 nginx tips 内容.

最新的文档发布.请朋友们关注: <http://bbs.linuxtone.org>