# Nginx 高级应用

Nginx除了可以作为后端HTTP服务器外，它还是一个高性能的反响代理服务器。在负载均衡的架构中，Nginx可以为我们提供非常稳定且高效的基于七层的负载解决方案。此外，Nginx可以根据规则以轮询、IP哈希、URL哈希等方式调度后端真实服务器，如同HAProxy一样，Nginx也支持对后患服务器的健康检查功能。

## 一，Nginx负载均衡

为了实现Nginx的反向代理以及负载均衡功能，我们需要用到HttpProxyModule 和 HttpUpstreamModule 模块。其中，HttpProxyModule 模块的作用是将用户的数据请求转发至其他服务器，HttpUpstreamModule 模块的作用是提供简单的负载均衡技术（轮询、最少连接、客户端IP）。这两个模块为Nginx默认的自动编译模块，如果不需要编译这些模块，可以用--without-http\_proxy\_module 与 --without-http\_upstream\_ip\_hash\_module两个编译参数禁用这些功能。

下面通过一个简单的例子说明Nginx作为反向代理服务器的配置方法。其中，location指令可以根据URI定义不同的配置，比如，当用户访问http://www.example.com/a时，我们可以定义将数据请求转发给后端A服务器；当用户访问http://www.example.com/b时，我们可以定义将数据请求转发给后端B服务器。location指令可以直接匹配字串，也可以匹配正则表达式，使用正则表达式时必须指定前缀~或~\*，~表示区分大小写，~\*代表不区分大小写，如果在匹配时希望达到精确匹配，则可以使用=前缀符号。

使用proxy\_pass指令可以根据location匹配的情况建立被代理的服务器或URI与代理服务器直接的映射关系。由于用户直接访问的是代理服务器，然后代理服务器负责转发数据请求给真实服务器。因此，后端真实服务器将无法获得客户端的IP地址，我们可以使用X-Forwarded-For实现重新定义数据包包头，记录用户真实的IP地址。

**location ~\* \.(mp3|mp4)$ { # 匹配URL以mp3或者mp4结尾的请求**

**proxy\_pass <http://localhost:8080>;**

**# location条件满足时将请求转发给本机8080端口**

**}**

**location / { # 匹配任意URL**

**proxy\_pass [http://localhost:8000](http://localhost:8080);**

**# location条件满足时将请求转发给本机8000端口**

**proxy\_set\_header X-Forwarded-For $remote\_addr;**

**# 保留用户真实的IP地址**

**}**

在负载均衡模块的指令中，upstream负责定义后端真实服务器集合，这样就可以通过proxy\_pass 和 fastcgi\_pass 将请求转发给一组服务器，在upstream中嵌套使用server指令可以设置具体的服务器及相关参数。

ip\_hast 指令可以根据用户IP地址的Hash值分配固定的后端服务器。

weight 的值代表服务器的权重。

down 指令可以设置某台服务器暂时处于宕机不可用状态。

简单的案例是：定义一个名为backend的服务器组，使用server定义组内所有的真实服务器成员，并将www2.example.com设置为宕机状态，当用户使用域名http://www.example.com访问代理服务器时，Nginx代理服务器会将请求转发给backend服务器组。

**http {**

**upstream backend {**

**ip\_hash;**

**server www1.example.com weight=2;**

**server www2.example.com down;**

**server www3.example.com ;**

**}**

**server {**

**listen 80;**

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

**location / {**

**proxy\_pass http://backend;**

**}**

**}**

**}**

## 二，Nginx负载均衡案例

Nginx反向代理负载均衡结构如下：

**|-------------------------------------------------- |**

**| |-----web1 |**

**| | |**

**client------------- nginx ----- <\\\\> ------|-----web2 |**

**| | |**

**| |-----web3 |  
 |-------------------------------------------------- |**

案例中Nginx使用upstream定义一组服务器，使用proxy\_pass进行代理转发

服务器名称 网络配置

nginx.example.cpm（代理服务器Nginx） eno16777736:10.10.10 外网IP

eno33554984:192.168.0.254 内网IP

web1.example.com eno16777736:192.168.0.1 内网IP

web2.example.com eno16777736:192.168.0.1 内网IP

web3.example.com eno16777736:192.168.0.1 内网IP

client.example.com （客户端client） eno16777736:10.10.10.100 客户端IP

**web服务器配置：**

三台Web服务器基本采用相同的配置，注意，IP地址需要改变。

**[root@web1 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno16777736**

DEVICE="eno16777736" # 物理设置的名字

BOOTPROTO="static" # 静态ip状态设置

ONBOOT="yes" # 开启激活该网卡

IPADDR=192.168.0.1 # 赋予该网卡的IP地址

PREFIX=24 # 网络位掩码个数

GATEWAY=192.168.0.254 # 默认网关

TYPE="Ethernet" # 网络类型为：Ethernet

**[root@web2 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno16777736**

DEVICE="eno16777736" # 物理设置的名字

BOOTPROTO="static" # 静态ip状态设置

ONBOOT="yes" # 开启激活该网卡

IPADDR=192.168.0.1 # 赋予该网卡的IP地址

PREFIX=24 # 网络位掩码个数

GATEWAY=192.168.0.254 # 默认网关

TYPE="Ethernet" # 网络类型为：Ethernet

**[root@web3 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno16777736**

DEVICE="eno16777736" # 物理设置的名字

BOOTPROTO="static" # 静态ip状态设置

ONBOOT="yes" # 开启激活该网卡

IPADDR=192.168.0.1 # 赋予该网卡的IP地址

PREFIX=24 # 网络位掩码个数

GATEWAY=192.168.0.254 # 默认网关

TYPE="Ethernet" # 网络类型为：Ethernet

以上三台web服务器需要安装httpd:

[root@web[123] ~]# systemctl restart network

[root@web[123] ~]# yum -y install httpd

[root@web1 ~]# echo "192.168.0.1" > /var/www/html/index.html

[root@web2 ~]# echo "192.168.0.2" > /var/www/html/index.html

[root@web3 ~]# echo "192.168.0.3" > /var/www/html/index.html

[root@web[123] ~]# firewall-cmd --permanent --add-port=80/tcp

[root@web[123] ~]# firewall-cmd --reload

**Nginx代理服务器设置：**

**[root@nginx ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno16777736**

DEVICE="eno16777736" # 物理设置的名字

BOOTPROTO="static" # 静态ip状态设置

ONBOOT="yes" # 开启激活该网卡

IPADDR=192.168.0.1 # 赋予该网卡的IP地址

PREFIX=8 # 网络位掩码个数

TYPE="Ethernet" # 网络类型为：Ethernet

**[root@nginx ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno33554984**

DEVICE="eno16777736" # 物理设置的名字

BOOTPROTO="static" # 静态ip状态设置

ONBOOT="yes" # 开启激活该网卡

IPADDR=192.168.0.254 # 赋予该网卡的IP地址

PREFIX=24 # 网络位掩码个数

TYPE="Ethernet" # 网络类型为：Ethernet

[root@nginx ~]# systemctl restart network

[root@nginx ~]# firewall-cmd --permanent --add-port=80/tcp

[root@nginx ~]# firewall-cmd --reload

[root@nginx ~]# wget <http://nginx.org/download/nginx-1.13.6.tar.gz>

[root@nginx ~]# tar -zxvf nginx-1.13.6.tar.gz -C /usr/src

[root@nginx ~]# yum -y install gcc pcre pcre-devel openssl openssl-devel \

> zlib-devel gd gd-devel perl perl-ExtUtils-Embed

[root@nginx ~]# cd /usr/src/nginx-1.13.6

[root@nginx nginx-1.13.6]# ./configure --prefix=/usr/local/nginx \

> --with-ipv6 \

> --with-http\_ssl\_module \

> --with-http\_realip\_module \

> --with-http\_addition\_module \

> --with-http\_dav\_module \

> --with-http\_flv\_module \

> --with-http\_mp4\_module \

> --with-http\_gzip\_static\_module \

> --with-http\_perl\_module \

> --with-mail \

> --with-mail\_ssl\_module

[root@nginx nginx-1.13.6]# make && make install

[root@nginx nginx-1.13.6]# cp /usr/local/nginx/conf/nginx.conf \

> /usr/local/nginx/conf/nginx.conf.bak

[root@nginx nginx-1.13.6]# vim /usr/local/nginx/conf/nginx.conf

**#设置用户组**

**user nginx;**

**#启动子进程数，可以通过ps aux | grep nginx 查看**

**worker\_processes 1;**

**#错误日志文件，以及日志级别**

**#error\_log logs/error.log;**

**#error\_log logs/error.log notice;**

**error\_log logs/error.log info;**

**#进程号保存文件**

**pid logs/nginx.pid;**

**events {**

**#每个进程可以处理的连接数，受系统文件句柄的限制**

**worker\_connections 1024;**

**}**

**http {**

**#mime.types 为文件类型定义文件**

**include mime.types;**

**#默认文件类型**

**default\_type application/octet-stream;**

**#使用log\_format可以自定义日志格式，名称为main**

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

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

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

**#创建访问日志，格式采用main定义的格式**

**#access\_log logs/access.log main;**

**#是否调用sendfile()进行数据复制，sendfile()复制数据是在内和级别完成的，所以**

**#会比一般的read、write更高效**

**sendfile on;**

**#开启后服务器的响应头部信息产生独立的数据包发送，即一个响应头信息一个包**

**tcp\_nopush on;**

**#保持连接的超时时间**

**#keepalive\_timeout 0;**

**keepalive\_timeout 65;**

**#是否采用压缩功能，将页面压缩后传输更节省流量**

**gzip on;**

**#容量小于1000B的内容不压缩**

**gzip\_min\_length 1000;**

**gzip\_buffers 16 32k;**

**#指定压缩文件类型**

**gzip\_types text/plain application/xml;**

**#压缩级别，可以是1至9之间的数字，数字越大，压缩效果越好，速度也就越满**

**gzip\_comp\_level 2;**

**#客户端请求的缓存大小**

**client\_body\_buffer\_size 128k;**

**#客户端请求所允许的最大文件容量**

**client\_max\_body\_size 100m;**

**#客户端请求的Header缓存大小**

**large\_client\_header\_buffers 4 8k;**

**#启用代理缓冲功能**

**proxy\_buffering on;**

**#从被代理服务器（真实服务器）所获取的第一部分响应数据的缓存大小**

**proxy\_buffer\_size 8k;**

**#从被代理服务器读取响应数据的缓存个数与容量**

**proxy\_buffers 8 128k;**

**#设置缓存目录及其他缓存参数，缓存路径为/usr/local/nginx/cache**

**#levels 用来设置子目录的个数**

**#kes\_zone 定义缓冲名称及容量（名称为one，容量为100m）**

**#inactive 定义缓存的存活时间为1天**

**#max\_size 设置硬盘缓存容量为2GB。**

**#这里定义的缓存为one,将在下面使用proxy\_cache指令调用该缓存**

**proxy\_cache\_path /usr/local/nginx/cache levels=1:2 keys\_zone=one:100m inactive=1d max\_size=2G;**

**#与后端服务器建立TCP握手的超时时间**

**proxy\_connect\_timeout 60s;**

**#设置后端服务器组，名称为servers**

**upstream servers {**

**#ip\_hash 使用IP哈希值确保相同IP的客户端使用相同的后端服务器，**

**#不使用该参数调度，其将采用轮询的方式进行数据转发**

**#ip\_hash;**

**#max\_fails 设置连接后端服务器3次失败则认为服务器处于无效状态**

**#fail\_timeout 设置连接后端服务器超时时间为30s**

**#weight 设置权重为2**

**server 192.168.0.1:80 max\_fails=3 fail\_timeout=30s weight=2;**

**server 192.168.0.2:80 max\_fails=3 fail\_timeout=30s weight=2;**

**server 192.168.0.3:80 max\_fails=3 fail\_timeout=30s weight=2;**

**}**

**server {**

**#监听80端口**

**listen 80;**

**#设置服务器域名**

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

**#记录日志**

**access\_log logs/host.access.log main;**

**#匹配任何网址**

**location / {**

**#条件满足时，将请求转发给servers服务器组**

**proxy\_pass http://servers;**

**#使用代理缓存one**

**proxy\_cache one;**

**#保留用户的真实IP地址**

**proxy\_set\_header X-Forwarded-For $remote\_addr;**

**}**

**}**

**}**

[root@nginx ~]# /usr/local/nginx/sbin/nginx #启动nginx

[root@nginx ~]# echo "/usr/local/nginx/sbin/nginx" >> /etc/rc.local #写入脚本开机自启动

**客户端设置：**

**[root@client ~]# vim /etc/sysconfig/network-scripts/ifcfg-eno16777736**

DEVICE="eno16777736" # 物理设置的名字

BOOTPROTO="static" # 静态ip状态设置

ONBOOT="yes" # 开启激活该网卡

IPADDR=10.10.10.100 # 赋予该网卡的IP地址

PREFIX=8 # 网络位掩码个数

TYPE="Ethernet" # 网络类型为：Ethernet

[root@client ~]# systemctl restart network

最后，通过客户端访问http://10.10.10.10，刷新将分别得到web1、web2以及web3三台主机返回的页面信息。

## 三，Nginx rewrite规则

由于互联网资源分布在世界各地，因此，如果用户希望查看某个资源，就需要使用URL来唯一定位该资源（如：<https://goole.com）。而URL重写则可以在Web服务器上面定制规则将用户访问资源的URL进行重新改写，改写后再交给Web服务器进程处理。>

Nginx的HttpRewirteModule模块支持基于PCRE（Perl Compatible Regular Expressions）的地址重写功能。

Nginx的rewrite语法格式与Apache非常类似：rewrite regex replacement[flag]，其中，flag可以被设置为last、break、redirect、permanent。

last 标记将结束当前rewrite指导并重新搜索location匹配；

break 标记也将结束当前rewrite指令，并且不在处理location块的其他匹配；

redirect 标记将返回临时重定向代码302；

permanent 将返回永久重定向代码301

在进行地址重定向的时候，一般会结合location或if语句进行。if语句的语法结构：

**if (条件) {...}**

其中对的if语句可以放置在配置文件中的server块或location块，可以使用一下测试操作符。

操作符 描述

~ 区分大小写匹配

~\* 不区分大小写匹配

= 精确匹配

-f 测试文件是否存在，存在且为文件则结果为真

-d 测试目录是否存在，存在则结果为真

-e 测试文件或目录是否存在，存在则结果为真

-x 测试文件是否可执行

! 区分，可以与以上操作符相结合

rewrite地址重写及return应用的语法说明：

#测试哭护短浏览器是否使用的是Microsoft IE浏览器，如果是，则用户访问的任何资源都将被重定向到/msie/目录下的相应资源，$http\_user\_agent 与 $1 是Nginx内置的变量。

# $http\_user\_agent 会返回客户端所使用的浏览器标识

# $1 为位置变量，即将用户原本需要访问的资源重定向到/msie/目录下的相应资源

if ($http\_user\_agent ~ MSIE) {

rewrite ^(.\*)$ /msie/$1 break;

}

#如果用户使用的是iPhone 或 iPod访问服务器，则将请求重定向至其他服务器

#这里使用了一个URI概念，URL是统一资源定位符，而URL是通用资源标示符

#如果一个URL为http://www.example.com/test/index.html，相对路径URI为

#/test/index.html

if ($http\_user\_agent ~\* '(iPhone|iPod)') {

rewrite ^.+ http://mobile.site.com$uri;

}

#用户使用POST方式请求数据时，则返回405错误

if ($request\_method = POST) {

return 405;

}

rewrite ^(/download/.\*)/media/(.\*)\..\*$ $1/mp3/$2.mp3 last;

rewrite ^(/download/.\*)/audio/(.\*)\..\*$ $1/mp3/$2.ra last;

## 四，Nginx rewrite重写案例

需求：当用户访问wp\_administrator目录下的资源时需要进行重定向admin目录。

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

**#设置用户组**

**user nginx;**

**#启动子进程数，可以通过ps aux | grep nginx 查看**

**worker\_processes 1;**

**#错误日志文件，以及日志级别**

**error\_log logs/error.log notice;**

**#进程号保存文件**

**pid logs/nginx.pid;**

**events {**

**#每个进程可以处理的连接数，受系统文件句柄的限制**

**worker\_connections 1024;**

**}**

**http {**

**#mime.types 为文件类型定义文件**

**include mime.types;**

**#默认文件类型**

**default\_type application/octet-stream;**

**#使用log\_format可以自定义日志格式，名称为main**

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

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

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

**#创建访问日志，格式采用main定义的格式**

**#access\_log logs/access.log main;**

**#是否调用sendfile()进行数据复制，sendfile()复制数据是在内和级别完成的，所以**

**#会比一般的read、write更高效**

**sendfile on;**

**#开启后服务器的响应头部信息产生独立的数据包发送，即一个响应头信息一个包**

**tcp\_nopush on;**

**server\_tokens off;**

**#保持连接的超时时间**

**keepalive\_timeout 65;**

**keepalive\_requests 100;**

**#是否采用压缩功能，将页面压缩后传输更节省流量**

**gzip on;**

**#容量小于1000B的内容不压缩**

**gzip\_min\_length 1000;**

**#压缩缓存的个数与容量**

**gzip\_buffers 16 32k;**

**#指定压缩文件类型**

**gzip\_types text/plain application/xml;**

**#压缩级别，可以是1至9之间的数字，数字越大，压缩效果越好，速度也就越满**

**gzip\_comp\_level 2;**

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

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

**#客户端请求的缓存大小**

**client\_body\_buffer\_size 128k;**

**#客户端请求所允许的最大文件容量**

**client\_max\_body\_size 100m;**

**#客户端请求的Header缓存大小**

**large\_client\_header\_buffers 4 8k;**

**server {**

**#监听80端口**

**listen 80;**

**root html;**

**index index.html;**

**#设置服务器域名**

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

**#记录日志**

**access\_log logs/host.access.log main;**

**if ($http\_user\_agent ~ MSIE) {**

**rewrite ^(.\*)$ /msie/$1 break;**

**}**

**if ($http\_user\_agent ~ Firefox) {**

**rewrite ^(.\*)$ /firefox/$1 break;**

**}**

**location /php\_admin {**

**rewrite ^/php\_admin/.\*$ /admin permanent;**

**}**

**}**

**}**

[root@centos7 ~]# mkdir /usr/local/nginx/html/{msie,firfox,admin}

[root@centos7 ~]# echo "firefox browser" > /usr/local/nginx/html/firefox/index.html

[root@centos7 ~]# echo "msie browser" > /usr/local/nginx/html/msie/index.html

[root@centos7 ~]# echo "Admin Page" > /usr/local/nginx/html/admin/index.html

[root@centos7 ~]# /usr/loca/nginx/sbin/nginx

客户端使用火狐浏览器访问http://192.168.0.108/index.html将获得页面内容firefox browser，使用IE浏览器访问http://192.168.0.108/index.html 将获得页面内容IE browser，其他浏览器访问http://192.168.0.108/php\_admin 将获得页面内容Admin Page。