# nginx下载地址

nginx第三方模块下载地址:

https://www.nginx.com/resources/wiki/modules/

nginx地址下载:

http://nginx.org/en/download.html

# 安装脚本

|  |
| --- |
| ./configure \  --user=nginx \  --group=nginx \  --prefix=/usr/share/nginx \  --sbin-path=/usr/sbin/nginx \  --conf-path=/etc/nginx/nginx.conf \  --error-log-path=/var/log/nginx/error.log \  --http-log-path=/var/log/nginx/access.log \  --pid-path=/var/run/nginx.pid \  --lock-path=/var/lock/subsys/nginx\  --with-http\_secure\_link\_module \  --with-http\_random\_index\_module \  --with-http\_ssl\_module \  --with-http\_realip\_module \  --with-http\_addition\_module \  --with-http\_sub\_module \  --with-http\_dav\_module \  --with-http\_flv\_module \  --with-http\_gzip\_static\_module \  --with-http\_stub\_status\_module \  --with-ipv6 \  --with-http\_perl\_module \  --with-http\_geoip\_module \  --with-mail \  --with-mail\_ssl\_module \  --add-module=/root/nginx/nginx-3rd/nginx-upstream-fair \  --add-module=/root/nginx/nginx-3rd/nginx\_mod\_h264\_streaming \  --add-module=/root/nginx/nginx-3rd/ngx\_cache\_purge  #--with-ld-opt='-L -Wl,-rpath,/usr/local/luajit/lib'  #--add-module=/root/rpmbuild/BUILD/nginx-1.8.1/lua-nginx-module-0.9.5rc2  #--add-module=/root/nginx/nginx-3rd/mod\_zip \  #--add-module=/root/nginx/nginx-3rd/ngx\_devel\_kit \  #--http-client-body-temp-path=/var/lib/nginx/tmp/client\_body \  #--http-proxy-temp-path=/var/lib/nginx/tmp/proxy \  #--http-fastcgi-temp-path=/var/lib/nginx/tmp/fastcgi \  #nginx-upstream-fair 比内建负载均衡模块更加好的负载均衡模块  #nginx\_mod\_h264\_streaming map4等流媒体  #ngx\_cache\_purge清楚缓存模块,不需要任何配置  #--prefix 指示web目录的路径 相当于 httpd的/var/www/html目录 |

# 3.安装报错的问题

perl module ExtUtils::Embed is required

解决: yum -y install perl-devel perl-ExtUtils-Embed

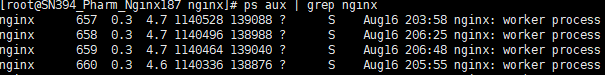
he GeoIP module requires the GeoIP library

解决: geoip-devel

pcre library找不到,下载pcre-devel就可以了

# 4.nginx.conf

worker\_processes auto; #工作进程个数, ps aux | grep nginx,设置为auto,它会根据cpu逻辑核心数目来定,查看核心数目 cat /proc/cpuinfo |grep processor



worker\_rlimit\_nofile 更改worker进程的最大打开文件数限制。如果没设置的话，这个值为操作系统的限制。设置后你的操作系统和Nginx可以处理比“ulimit -a”更多的文件，所以把这个值设高，这样nginx就不会有“too many open files”问题了

worker\_connections 1024; worker的最大连接数目

sendfile on; #打开系统sendfile()函数的支持,普通的read(),write()函数读写上太慢,需要切换很多次上下文环境,而sendfile()函数是底层的函数,效率高,不用切换很多次上下文环境即用户空间和内核空间

tcp\_nopush      on;

会设置调用tcp\_cork方法，这个也是默认的，结果就是数据包不会马上传送出去，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。

tcp\_nodelay on; #不需要等待0.2s后才能发送数据包,即立即发送和tcp\_nopush互斥,tcp\_nopush 设置为no,则这个设置为off

proxy\_connect\_timeout 60;#nginx跟后端服务器连接超时时间（代理连接超时）

proxy\_read\_timeout 120;#连接成功后，后端服务器响应时间

proxy\_send\_timeout 120;#后端服务器数据回传时间

proxy\_buffers 8 4k

该指令设置缓冲区的大小和数量,从被代理的后端服务器取得的响应内容,(数据)会放置到这里. 默认情况下,一个缓冲区的大小等于内存页面大小,可能是4K也可能是8K,这取决于平台,每个请求对应于所有的proxy\_buffers,而不是全部proxy\_buffer才8个,

proxy\_buffer\_size 4k

该指令设置缓冲区大小,主要缓冲从后端服务器发来的响应报文的头部信息,每个请求对应一个默认来说,该缓冲区大小等于指令 proxy\_buffers所设置的;但是,你可以把它设置得更小

proxy\_busy\_buffers\_size 8k;为proxy\_buffer\_size的2倍

proxy\_buffering on/off

为on,则开启proxy\_buffers和proxy\_busy\_buffers\_size,不管proxy\_buffering是on还是off,proxy\_buffer\_size都是开启的,proxy\_buffering on表示缓冲区开启,每个请求所获得的响应报文会会先存到nginx的缓冲区中,如果存不下,则写到硬盘上(即文件里),此时buffer一直处于busy直到buffer被写满,或数据接受完了,buffer开始从缓冲区发送数据到客户端.如果为off则不缓冲.只有proxy\_buffer\_size接受响应头部信息.

proxy\_max\_temp\_file\_size和proxy\_temp\_file\_write\_size

临时文件由proxy\_max\_temp\_file\_size和proxy\_temp\_file\_write\_size这两个指令决定。 proxy\_temp\_file\_write\_size是一次访问能写入的临时文件的大小，默认是proxy\_buffer\_size和proxy\_buffers中设置的缓冲区大小的2倍，Linux下一般是8k。

proxy\_max\_temp\_file\_size指定当响应内容大于proxy\_buffers指定的缓冲区时, 写入硬盘的临时文件的大小. 如果超过了这个值, Nginx将与Proxy服务器同步的传递内容, 而不再缓冲到硬盘. 设置为0时, 则直接关闭硬盘缓冲.

proxy\_temp\_path /data/nginx/proxy\_temp;设置临时文件路径

proxy\_cache\_path /data/nginx/proxy\_cache levels=2:2 keys\_zone=cache\_one:1024m max\_size=40g inactive=3d

设定缓冲区名称cache\_one ,levels=2:2 表示一级目录时2个字符,二级目录时2个字符,内存缓冲空间大小为1024MB,cache\_one的硬盘容量为40g,缓存文件3天没被访问就自动被删掉.

proxy\_temp\_path和proxy\_cache\_path指定的路径必须在同一分区

proxy\_cache\_valid 200 302 3d

说明对于状态为200和304的缓存文件的缓存时间是3天,3天后再访问该缓存文件,该文件会过期,从而去上游服务器去找

# 5.location中url匹配

规律: = none(location后没有符号),~^ 匹配的都是普通的字符串,~和~\*(忽略大小写)匹配的是正则表达式

优先级:

先比较符号

= > ^~ > ~/~\* > none

再比较字符串匹配的长度

= 固定字符串匹配

~^和none匹配字符串开头的,例如location ^~ /abc 则匹配 /abc,/abcd,/abc/d/e

~/~\*匹配正则表达式

如果配置文件中同时存在 location /a 和 location ^~ 则会报错” nginx: [emerg] duplicate”

# 6.负载均衡的5中方法

1.轮询

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.

uptream backend{

server 192.168.200.156:8080;

server 192.168.200.157:8080;

server 192.168.200.158:8080;

}

2.ip\_hash

nginx中的ip\_hash技术能够将某个ip的请求定向到固定的后端服务器,这样客户端的

ip能够建立起稳定的会话(session)

upstream backend{

ip\_hash;

server 192.168.200.156:8080;

server 192.168.200.157:8080;

server 192.168.200.158:8080;

}

注意:ip\_hash在以下情况下不能使用

1)nginx不是最前端的服务器,ip\_hash要求nginx一定是最前端的服务器,否则nginx得不到

正确的客户端ip,就不能通过ip作hash即键值对中的键值,譬如squid是最前端,后面接着

nginx,nginx使用ip\_hash作负载均衡的话,得到的全都是squid的ip地址

2)nginx的后端还有其他方式的负载均衡,假如nginx后端有其他nginx作为负载均衡

那么最前端的nginx使用ip\_hash的话,将客户端ip只能定向到固定的后面的nginx

服务器,而不是真正的服务器.

3.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。

upstream backend{

server 192.168.200.156:8080 weight=10;

server 192.168.200.157:8080 weight=20;

server 192.168.200.158:8080 weight=30;

}

4.fair(第三方模块:nginx-upstream-fair)

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

upstream backend{

fair;

server 192.168.200.156:8080;

server 192.168.200.157:8080;

server 192.168.200.158:8080;

}

5.url\_hash(第三方模块:nginx-module-url)

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

upstream backend{

hash $request\_uri;

server 192.168.200.156:8080;

server 192.168.200.157:8080;

server 192.168.200.158:8080;

}

每个设备的状态设置为:  1.down 表示单前的server暂时不参与负载  2.weight 默认为1.weight越大，负载的权重就越大。  3.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  4.fail\_timeout:max\_fails次失败后，暂停的时间。  5.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，用来给不用的server来使用。

# 7.负载均衡例子

1.nginx主机名:www.b.com

2.后台两个web服务器: 192.168.200.149和192.168.200.146

VirtualHost配置:

<VirtualHost \*:80>

ServerAdmin webmaster@dummy-host.example.com

DocumentRoot /var/www/html/b

ServerName www.b.com

#ErrorLog

CustomLog logs/b/b-access.log vcombined #b目录要存在,不然报错

#设置主页

DirectoryIndex b\_index.html

</VirtualHost>

3.nginx.conf

http{

upstream test1{

server 192.168.200.146:80;

server 192.168.200.149:80;

}

location / {

proxy\_set\_header Host $host; #作用是如果不写这一行,则会导致上游web服务器中的host字段为test1,而不是客户要访问的主机名称(nginx的主机

名)

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_pass <http://test1>;

}

}

# 8.nginx内建变量

$args #请求中的参数值

$query\_string #同 $args

$arg\_NAME #GET请求中NAME的值

$is\_args #如果请求中有参数，值为"?"，否则为空字符串

$uri #请求中的当前URI(不带请求参数，参数位于$args)，可以不同于浏览器传递的$request\_uri的值，它可以通过内部重定向，或者使用index指令进行修改，$uri不包含主机名，如"/foo/bar.html"。

$document\_uri #同 $uri

$document\_root #当前请求的文档根目录或别名

$host #优先级：HTTP请求行的主机名>"HOST"请求头字段>符合请求的服务器名

$hostname #主机名

$https #如果开启了SSL安全模式，值为"on"，否则为空字符串。

$binary\_remote\_addr #客户端地址的二进制形式，固定长度为4个字节

$body\_bytes\_sent #传输给客户端的字节数，响应头不计算在内；这个变量和Apache的mod\_log\_config模块中的"%B"参数保持兼容

$bytes\_sent #传输给客户端的字节数

$connection #TCP连接的序列号

$connection\_requests #TCP连接当前的请求数量

$content\_length #"Content-Length" 请求头字段

$content\_type #"Content-Type" 请求头字段

$cookie\_name #cookie名称

$limit\_rate #用于设置响应的速度限制

$msec #当前的Unix时间戳

$nginx\_version #nginx版本

$pid #工作进程的PID

$pipe #如果请求来自管道通信，值为"p"，否则为"."

$proxy\_protocol\_addr #获取代理访问服务器的客户端地址，如果是直接访问，该值为空字符串

$realpath\_root #当前请求的文档根目录或别名的真实路径，会将所有符号连接转换为真实路径

$remote\_addr #客户端地址

$remote\_port #客户端端口

$remote\_user #用于HTTP基础认证服务的用户名

$request #代表客户端的请求地址

$request\_body #客户端的请求主体：此变量可在location中使用，将请求主体通过proxy\_pass，fastcgi\_pass，uwsgi\_pass和scgi\_pass传递给下一级的代理服务器

$request\_body\_file #将客户端请求主体保存在临时文件中。文件处理结束后，此文件需删除。如果需要之一开启此功能，需要设置client\_body\_in\_file\_only。如果将次文件传递给后端的代理服务器，需要禁用request body，即设置proxy\_pass\_request\_body off，fastcgi\_pass\_request\_body off，uwsgi\_pass\_request\_body off，or scgi\_pass\_request\_body off

$request\_completion #如果请求成功，值为"OK"，如果请求未完成或者请求不是一个范围请求的最后一部分，则为空

$request\_filename #当前连接请求的文件路径，由root或alias指令与URI请求生成

$request\_length #请求的长度 (包括请求的地址，http请求头和请求主体)

$request\_method #HTTP请求方法，通常为"GET"或"POST"

$request\_time #处理客户端请求使用的时间; 从读取客户端的第一个字节开始计时

$request\_uri #这个变量等于包含一些客户端请求参数的原始URI，它无法修改，请查看$uri更改或重写URI，不包含主机名，例如："/cnphp/test.php?arg=freemouse"

$scheme #请求使用的Web协议，"http" 或 "https"

$server\_addr #服务器端地址，需要注意的是：为了避免访问linux系统内核，应将ip地址提前设置在配置文件中

$server\_name #服务器名

$server\_port #服务器端口

$server\_protocol #服务器的HTTP版本，通常为 "HTTP/1.0" 或 "HTTP/1.1"

$status #HTTP响应代码

$time\_iso8601 #服务器时间的ISO 8610格式

$time\_local #服务器时间（LOG Format 格式）

$cookie\_NAME #客户端请求Header头中的cookie变量，前缀"$cookie\_"加上cookie名称的变量，该变量的值即为cookie名称的值

$http\_NAME #匹配任意请求头字段；变量名中的后半部分NAME可以替换成任意请求头字段，如在配置文件中需要获取http请求头："Accept-Language"，$http\_accept\_language即可

$http\_cookie

$http\_post

$http\_referer

$http\_user\_agent

$http\_x\_forwarded\_for

$sent\_http\_NAME #可以设置任意http响应头字段；变量名中的后半部分NAME可以替换成任意响应头字段，如需要设置响应头Content-length，$sent\_http\_content\_length即可

$sent\_http\_cache\_control

$sent\_http\_connection

$sent\_http\_content\_type

$sent\_http\_keep\_alive

$sent\_http\_last\_modified

$sent\_http\_location

$sent\_http\_transfer\_encoding

# 9.缓存配置

http{

#注意/data/nginx目录一定要存在,不然报错,proxy\_cache会自动生成

proxy\_cache\_path /data/nginx/proxy\_cache levels=2:2 keys\_zone=cache\_one:200m max\_size=40g inactive=3d;

#注意这个purge要放在 下一个location的上面,不然会被下面的缓存location抢先匹配

location ~ /purge(/.\*)

{

#加密

#auth\_basic "input pharmnet purge name ";

#auth\_basic\_user\_file conf.d/nginx\_passwd;

# allow 202.107.194.0/24;

# allow 10.103.33.0/24;

# deny all;

proxy\_cache\_purge cache\_one $host$1$is\_args$args;

}

location ~ .\*\.(js|css|ico|png|jpg|eot|svg|ttf|woff|html) {

#添加一个header显示缓存是否命中,

#MISS 未命中，请求被传送到后端

#·HIT 缓存命中

#·EXPIRED 缓存已经过期请求被传送到后端,这个在超过proxy\_cache\_valid 设置的过期时间时,再次返回就会返回EXPIRED(200)

#·UPDATING 正在更新缓存，将使用旧的应答

#·STALE 后端将得到过期的应答

#在上游httpd日志中可以获取其值

add\_header Nginx-Cache "$upstream\_cache\_status";

#cache\_one是在http里定义的缓存名称

proxy\_cache cache\_one;

#说明对于状态为200和304的缓存文件的缓存时间是1分钟，1分钟之后再访问该缓存文件时，文件会过期，从而去源服务器重新取数据。

proxy\_cache\_valid 200 304 1m;

#uri都对应一个key,这样客户端访问一个url时,根据根据url得到key,去nginx里找看有没有这个key,有的话,就回去key所对应的缓存内容,没有的话就转到上游web服务器

#proxy\_cache\_key就是用来设置key的格式

proxy\_cache\_key $host$uri$is\_args$args;

proxy\_set\_header Host $host; #作用是如果不写这一行,则会导致上游web服务器中的host字段为test1,而不是客户要访问的主机名称(nginx的主机名)

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_pass http://test1;

}

}

# 10.nginx由service控制

vi /etc/init.d/nginx

|  |
| --- |
| #!/bin/sh  #  # nginx - this script starts and stops the nginx daemon  #  # chkconfig: - 85 15  # description: NGINX is an HTTP(S) server, HTTP(S) reverse \  # proxy and IMAP/POP3 proxy server  # processname: nginx  # config: /etc/nginx/nginx.conf  # config: /etc/sysconfig/nginx  # pidfile: /var/run/nginx.pid  # Source function library.  . /etc/rc.d/init.d/functions  # Source networking configuration.  . /etc/sysconfig/network  # Check that networking is up.  [ "$NETWORKING" = "no" ] && exit 0  nginx="/usr/sbin/nginx"  prog=$(basename $nginx)  NGINX\_CONF\_FILE="/etc/nginx/nginx.conf"  [ -f /etc/sysconfig/nginx ] && . /etc/sysconfig/nginx  lockfile=/var/lock/subsys/nginx  make\_dirs() {  # make required directories  user=`$nginx -V 2>&1 | grep "configure arguments:" | sed 's/[^\*]\*--user=\([^ ]\*\).\*/\1/g' -`  if [ -z "`grep $user /etc/passwd`" ]; then  useradd -M -s /bin/nologin $user  fi  options=`$nginx -V 2>&1 | grep 'configure arguments:'`  for opt in $options; do  if [ `echo $opt | grep '.\*-temp-path'` ]; then  value=`echo $opt | cut -d "=" -f 2`  if [ ! -d "$value" ]; then  # echo "creating" $value  mkdir -p $value && chown -R $user $value  fi  fi  done  }  start() {  [ -x $nginx ] || exit 5  [ -f $NGINX\_CONF\_FILE ] || exit 6  make\_dirs  echo -n $"Starting $prog: "  daemon $nginx -c $NGINX\_CONF\_FILE  retval=$?  echo  [ $retval -eq 0 ] && touch $lockfile  return $retval  }  stop() {  echo -n $"Stopping $prog: "  killproc $prog -QUIT  retval=$?  echo  [ $retval -eq 0 ] && rm -f $lockfile  return $retval  }  restart() {  configtest || return $?  stop  sleep 1  start  }  reload() {  configtest || return $?  echo -n $"Reloading $prog: "  killproc $nginx -HUP  RETVAL=$?  echo  }  force\_reload() {  restart  }  configtest() {  $nginx -t -c $NGINX\_CONF\_FILE  }  rh\_status() {  status $prog  }  rh\_status\_q() {  rh\_status >/dev/null 2>&1  }  case "$1" in  start)  rh\_status\_q && exit 0  $1  ;;  stop)  rh\_status\_q || exit 0  $1  ;;  restart|configtest)  $1  ;;  reload)  rh\_status\_q || exit 7  $1  ;;  force-reload)  force\_reload  ;;  status)  rh\_status  ;;  condrestart|try-restart)  rh\_status\_q || exit 0  ;;  \*)  echo $"Usage: $0 {start|stop|status|restart|condrestart|try-restart|reload|force-reload|configtest}"  exit 2  esac |

chmod 755 /etc/init.d/nginx

chkconfig --add /etc/init.d/nginx

# 11.日志格式

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

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

'"$http\_user\_agent" "$http\_host" up\_ip: $upstream\_addr $upstream\_http\_host '

'up\_type $upstream\_http\_content\_type' ' gzip "$gzip\_ratio" '

' "$upstream\_cache\_status" "$http\_x\_forwarded\_for" $cookie\_nsunid';

# 12.logrotate日志轮询

在/etc/logrotate.d下新建nginx文件:

|  |
| --- |
| /data/nginx/logs/access.log{  daily  rotate 3  missingok  notifempty  compress  delaycompress  sharedscripts  postrotate  [ ! -f /var/run/nginx.pid ] || kill -USR1 `cat /var/run/nginx.pid`  endscript  } |

postrotate里表示让nginx重新生成日志,不然nginx还是会把日志发送到生成的日志,例如日志文件是access.log,如果mv access.log access.log,此时nginx还是会把日志输入到access.log.1中,而不会新生成access.log,而postrotate中的脚本就是用来生成新的access.log的