nginx基本配置与参数说明以及Nginx中的upstream轮询机制介绍

[收藏](javascript:void(0);)

[谢思华](https://my.oschina.net/xsh1208/home)

* 发表于 1年前

* 阅读 876

* 收藏 11

* 点赞 0

* [评论 0](https://my.oschina.net/xsh1208/blog/492374#comment-list)

[**开程序员的淘宝店！寻找开源技术服务伙伴！>>>**](https://zb.oschina.net/shop/join)  https://my.oschina.net/img/hot3.png

一.nginx简介

        Nginx (发音为[engine x])专为性能优化而开发，其最知名的优点是它的稳定性和低系统资源消耗，以及对并发连接的高处理能力(单台物理服务器可支持30000～50000个并发连接)， 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。

        实际使用中，nginx主要：反向代理、负载均衡

二.nginx基本配置与参数说明

#运行用户

user nobody;

#**启动进程,**通常设置成和**cpu的数量相等**

worker\_processes  1;

#全局错误日志及PID文件

#error\_log  logs/error.log;

#error\_log  logs/error.log  notice;

#error\_log  logs/error.log  info;

#pid        logs/nginx.pid;

#**工作模式**及连接数上限

events {

    #epoll是多路复用IO(I/O Multiplexing)中的一种方式,

    #仅用于linux2.6以上内核,可以大大提高nginx的性能

use   epoll;

    #**单个后台worker process进程**的**最大并发链接数**

worker\_connections  1024;

    # 并发总数是 worker\_processes 和 worker\_connections 的乘积

    # 即 max\_clients = worker\_processes \* worker\_connections

    # 在设置了反向代理的情况下，max\_clients = worker\_processes \* worker\_connections / 4  为什么

    # 为什么上面反向代理要除以4，应该说是一个经验值

    # 根据以上条件，正常情况下的Nginx Server可以应付的最大连接数为：4 \* 8000 = 32000

    # worker\_connections 值的设置跟物理内存大小有关

    # 因为并发受IO约束，max\_clients的值须小于系统可以打开的最大文件数

    # 而系统可以打开的最大文件数和内存大小成正比，一般1GB内存的机器上可以打开的文件数大约是10万左右

    # 我们来看看360M内存的VPS可以打开的文件句柄数是多少：

    # $ cat /proc/sys/fs/file-max

    # 输出 34336

    # 32000 < 34336，即并发连接总数小于系统可以打开的文件句柄总数，这样就在操作系统可以承受的范围之内

    # 所以，worker\_connections 的值需根据 worker\_processes 进程数目和系统可以打开的最大文件总数进行适当地进行设置

    # 使得并发总数小于操作系统可以打开的最大文件数目

    # 其实质也就是根据主机的物理CPU和内存进行配置

    # 当然，理论上的并发总数可能会和实际有所偏差，因为主机还有其他的工作进程需要消耗系统资源。

    # ulimit -SHn 65535

}

#**设定http服务器**

http {

    #**设定mime类型,类型由mime.type文件定义**

include    mime.types;

default\_type  application/octet-stream;

    #**设定日志格式**

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

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

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

access\_log  logs/access.log  main;

    #**sendfile 指令指定 nginx 是否调用** sendfile 函数（zero copy 方式）来输出文件，

    #**对于普通应用，必须设为 on**,

    #如果用来进行下载等应用磁盘IO重负载应用，可设置为 off，

    #以平衡磁盘与网络I/O处理速度，降低系统的uptime.

sendfile     on;

    #tcp\_nopush     on;

    #**连接超时时间**

    #keepalive\_timeout  0;

keepalive\_timeout  65;

tcp\_nodelay     on;

 #**FastCGI相关参数**是为了**改善网站的性能：减少资源占用，提高访问速度**。下面参数看字面意思都能理解。

fastcgi\_connect\_timeout 300;

fastcgi\_send\_timeout 300;

fastcgi\_read\_timeout 300;

fastcgi\_buffer\_size 64k;

fastcgi\_buffers 4 64k;

fastcgi\_busy\_buffers\_size 128k;

fastcgi\_temp\_file\_write\_size 128k;

      #**upstream的负载均衡，（以权重方式分发），weight是权重**，可以根据机器配置定义权重。weigth参数表示权值，**权值越高被分配到的几率越大**。

upstream blog.nginx.com {

    server 192.168.80.121:80 weight=3;

    server 192.168.80.122:80 weight=2;

    server 192.168.80.123:80 weight=3;

}

      #**upstream的负载均衡，（以nginx热备方式分发），**其它所有的非backup Server down或者忙的时候，请求backup机器。所以**这台机器压力会最轻**。

upstream blog.nginx.com {

    server 192.168.80.121:80;

    server 192.168.80.122:80;

    server 192.168.80.123:80 backup;

}

    #**开启gzip压缩**

gzip  on;

gzip\_disable "MSIE [1-6].";

    #**设定请求缓冲**

client\_header\_buffer\_size    128k;

large\_client\_header\_buffers  4 128k;

    #**设定虚拟主机配置**

    server {

        #**侦听80端口**

listen    80;

        #定义**访问地址**，域名可以有多个，用空格隔开

server\_name www.nginx.cn nginx.cn ;

        #定义**服务器的默认网站根目录位置**

root html;

        #设定本虚拟主机的访问日志

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

        #**默认请求**

location / {

   #定义首页索引文件的名称

   index index.php index.html index.htm;

}

         #对 “/” **启用反向代理**

location / {

    proxy\_pass http://127.0.0.1:88;

    proxy\_redirect off;

    proxy\_set\_header X-Real-IP $remote\_addr;

    #后端的Web服务器可以通过X-Forwarded-For获取用户真实IP

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

    #以下是一些反向代理的配置，可选。

    proxy\_set\_header Host $host;

    client\_max\_body\_size 10m; #允许客户端请求的最大单文件字节数

    client\_body\_buffer\_size 128k; #缓冲区代理缓冲用户端请求的最大字节数，

    proxy\_connect\_timeout 90; #nginx跟后端服务器连接超时时间(代理连接超时)

    proxy\_send\_timeout 90; #后端服务器数据回传时间(代理发送超时)

    proxy\_read\_timeout 90; #连接成功后，后端服务器响应时间(代理接收超时)

    proxy\_buffer\_size 4k; #设置代理服务器（nginx）保存用户头信息的缓冲区大小

    proxy\_buffers 4 32k; #proxy\_buffers缓冲区，网页平均在32k以下的设置

    proxy\_busy\_buffers\_size 64k; #高负荷下缓冲大小（proxy\_buffers\*2）

    proxy\_temp\_file\_write\_size 64k;

    #设定缓存文件夹大小，大于这个值，将从upstream服务器传

}

        #设定查看Nginx状态的地址

location /NginxStatus {

    stub\_status on;

    access\_log on;

    auth\_basic “NginxStatus”;

    auth\_basic\_user\_file conf/htpasswd;

    #htpasswd文件的内容可以用apache提供的htpasswd工具来产生。

}

        #本地动静分离反向代理配置

        #所有jsp的页面均交由tomcat或resin处理

location ~ .(jsp|jspx|do)?$ {

    proxy\_set\_header Host $host;

    proxy\_set\_header X-Real-IP $remote\_addr;

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

    proxy\_pass http://127.0.0.1:8080;

}

    #所有静态文件由nginx直接读取不经过tomcat或resin

location ~ .\*.(htm|html|gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt

                |pdf|xls|mp3|wma)$ {

    expires 15d;

}

location ~ .\*.(js|css)?$ {

    expires 1h;

}

        # 定义**错误提示页面**

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

location = /50x.html {

}

        #**静态文件缓存时间设置，nginx自己处理**

location ~ ^/(images|javascript|js|css|flash|media|static)/ {

   #过期30天，静态文件不怎么更新，过期可以设大一点，

   #如果频繁更新，则可以设置得小一点。

   expires 30d;

}

        #PHP 脚本**请求全部转发到** FastCGI处理. 使用FastCGI默认配置.

location ~ .php$ {

   fastcgi\_pass 127.0.0.1:9000;

   fastcgi\_index index.php;

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

   include fastcgi\_params;

}

        #禁止访问 .htxxx 文件

location ~ /.ht {

   deny all;

}

    }

}

三.nginx配置超时时间

<http://my.oschina.net/xsh1208/blog/199674>

四.Nginx中的upstream轮询机制介绍【负载均衡】

Nginx中upstream有以下几种方式：

1、轮询(weight=1)   
默认选项，当weight不指定时，各服务器weight相同， 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。

upstream bakend {

    server 192.168.1.10;

    server 192.168.1.11;

}

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。   
 如果后端服务器down掉，能自动剔除。   
 比如下面配置，则1.11服务器的访问量为1.10服务器的两倍。

upstream bakend {

    server 192.168.1.10 weight=1;

    server 192.168.1.11 weight=2;

}

3、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不能跨服务器的问题。   
如果后端服务器down掉，要手工down掉。

upstream resinserver{

    ip\_hash;

    server 192.168.1.10:8080;

    server 192.168.1.11:8080;

}

4、fair（第三方插件）   
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 resinserver{

    server 192.168.1.10:8080;

    server 192.168.1.11:8080;

    fair;

}

5、url\_hash（第三方插件）   
 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，后端服务器为缓存服务器时比较有效。   
 在upstream中加入hash语句，hash\_method是使用的hash算法

upstream resinserver{

server 192.168.1.10:8080;

server 192.168.1.11:8080;

hash $request\_uri;

hash\_method crc32;

}

设备的状态有:   
1.down 表示单前的server暂时不参与负载   
2.weight 权重,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 备用服务器, 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。

五.nginx启动、重启、关闭操作介绍

启动nginx:

nginx -c /path/to/nginx.conf

重启nginx：

nginx -s reload ：修改配置后重新加载生效

nginx -s reopen ：重新打开日志文件

nginx -t -c /path/to/nginx.conf 测试nginx配置文件是否正确

关闭nginx：

nginx -s stop :快速停止nginx

quit ：完整有序的停止nginx

其他的停止nginx 方式：

ps -ef | grep nginx

kill -QUIT 主进程号 ：从容停止Nginx

kill -TERM 主进程号 ：快速停止Nginx

pkill -9 nginx ：强制停止Nginx

平滑重启nginx：

kill -HUP 主进程号