nginx配置学习资料

序言

Nginx是Igor Sysoev为俄罗斯访问量第二的rambler.ru站点设计开发的。从2004年发布至今，凭借开源的力量，已经接近成熟与完善。

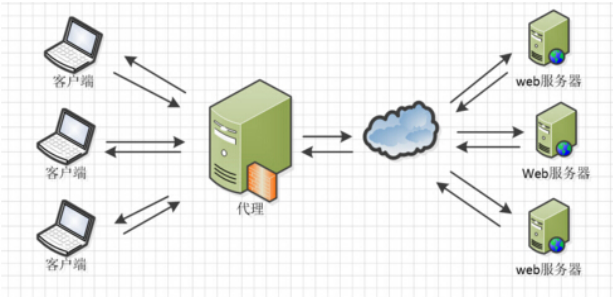
Nginx功能丰富，可作为HTTP服务器，也可作为反向代理服务器，邮件服务器。支持FastCGI、SSL、Virtual Host、URL Rewrite、Gzip等功能。并且支持很多第三方的模块扩展。

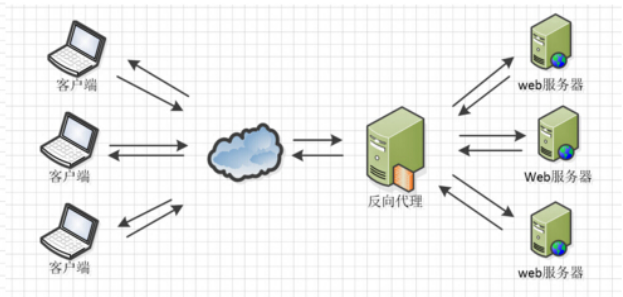
Nginx的稳定性、功能集、示例配置文件和低系统资源的消耗让他后来居上，在全球活跃的网站中有12.18%的使用比率，大约为2220万个网站。

Nginx常用功能

1、Http代理，反向代理：作为web服务器最常用的功能之一，尤其是反向代理。

这里我给来2张图，对正向代理与反响代理做个诠释，具体细节，大家可以翻阅下资料。



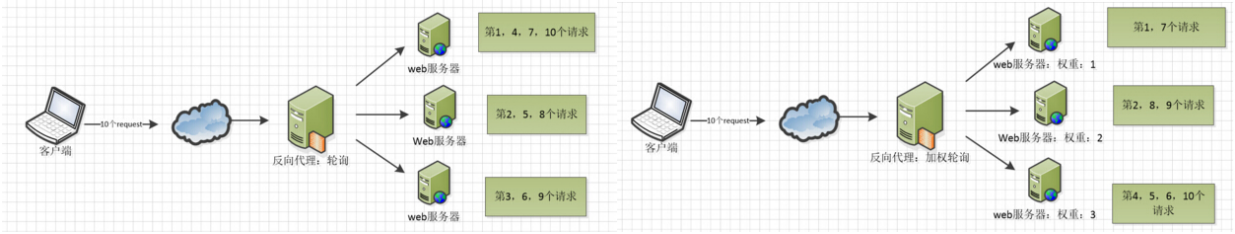


Nginx在做反向代理时，提供性能稳定，并且能够提供配置灵活的转发功能。Nginx可以根据不同的正则匹配，采取不同的转发策略，比如图片文件结尾的走文件服务器，动态页面走web服务器，只要你正则写的没问题，又有相对应的服务器解决方案，你就可以随心所欲的玩。并且Nginx对返回结果进行错误页跳转，异常判断等。如果被分发的服务器存在异常，他可以将请求重新转发给另外一台服务器，然后自动去除异常服务器。

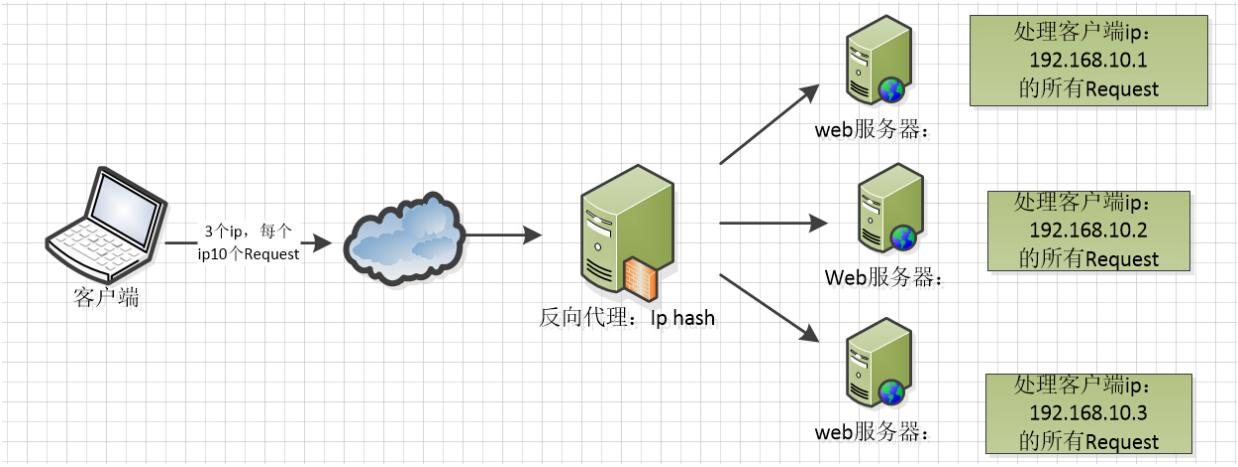
2、负载均衡

Nginx提供的负载均衡策略有2种：内置策略和扩展策略。内置策略为轮询，加权轮询，Ip hash。扩展策略，就天马行空，只有你想不到的没有他做不到的啦，你可以参照所有的负载均衡算法，给他一一找出来做下实现。

上3个图，理解这三种负载均衡算法的实现



Ip hash算法，对客户端请求的ip进行hash操作，然后根据hash结果将同一个客户端ip的请求分发给同一台服务器进行处理，可以解决session不共享的问题。



3、web缓存

Nginx可以对不同的文件做不同的缓存处理，配置灵活，并且支持FastCGI_Cache，主要用于对FastCGI的动态程序进行缓存。配合着第三方的ngx_cache_purge，对制定的URL缓存内容可以的进行增删管理。

4、Nginx相关地址

源码：https://trac.nginx.org/nginx/browser

官网：http://www.nginx.org/

Nginx配置文件结构

如果你下载好啦，你的安装文件，不妨打开conf文件夹的nginx.conf文件，Nginx服务器的基础配置，默认的配置也存放在此。

在nginx.conf的注释符号位#

nginx文件的结构，这个对刚入门的同学，可以多看两眼。

默认的config

```
#user  nobody;
worker_processes  1;
```

```
3
4   #error_log  logs/error.log;
5   #error_log  logs/error.log  notice;
6   #error_log  logs/error.log  info;
7
8   #pid        logs/nginx.pid;
9
10
11  events {
12      worker_connections  1024;
13  }
14
15
16  http {
17      include       mime.types;
18      default_type  application/octet-stream;
19
20      #log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
21      #                  '$status $body_bytes_sent "$http_referer" '
22      #                  '"$http_user_agent" "$http_x_forwarded_for"';
23
24      #access_log  logs/access.log  main;
25
26      sendfile        on;
27      #tcp_nopush     on;
28
29      #keepalive_timeout  0;
30      keepalive_timeout  65;
31
32      #gzip  on;
33
34      server {
35          listen       80;
36          server_name  localhost;
37
38          #charset koi8-r;
39
40          #access_log  logs/host.access.log  main;
41
42          location / {
43              root   html;
44              index  index.html index.htm;
45          }
46
47          #error_page  404              /404.html;
48
49          # redirect server error pages to the static page /50x.html
50          #
51          error_page   500 502 503 504  /50x.html;
```

```
        location = /50x.html {
            root   html;
        }

        # proxy the PHP scripts to Apache listening on 127.0.0.1:80
        #
        #location ~ \.php$ {
        #    proxy_pass   http://127.0.0.1;
        #}

        # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
        #
        #location ~ \.php$ {
        #    root           html;
        #    fastcgi_pass   127.0.0.1:9000;
        #    fastcgi_index  index.php;
        #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
        #    include        fastcgi_params;
        #}

        # deny access to .htaccess files, if Apache's document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #    deny  all;
        #}
    }


    # another virtual host using mix of IP-, name-, and port-based
configuration
    #
    #server {
    #    listen       8000;
    #    listen       somename:8080;
    #    server_name  somename  alias  another.alias;

    #    location / {
    #        root   html;
    #        index  index.html index.htm;
    #    }
    #}


    # HTTPS server
    #
    #server {
    #    listen       443 ssl;
    #    server_name  localhost;
```

```
100
101     #    ssl_certificate      cert.pem;
102     #    ssl_certificate_key  cert.key;
103
104     #    ssl_session_cache    shared:SSL:1m;
105     #    ssl_session_timeout  5m;
106
107     #    ssl_ciphers  HIGH:!aNULL:!MD5;
108     #    ssl_prefer_server_ciphers  on;
109
110     #    location / {
111     #        root   html;
112     #        index  index.html index.htm;
113     #    }
114     #}
115
116 }
```

nginx文件结构

```
1  ...              #全局块
2
3  events {         #events块
4     ...
5  }
6
7  http      #http块
8  {
9      ...   #http全局块
10     server        #server块
11     {
12         ...       #server全局块
13         location [PATTERN]   #location块
14         {
15             ...
16         }
17         location [PATTERN]
18         {
19             ...
20         }
21     }
22     server
23     {
24       ...
25     }
26     ...     #http全局块
27 }
```

1、全局块：配置影响nginx全局的指令。一般有运行nginx服务器的用户组，nginx进程pid存放路径，日志存放路径，配置文件引入，允许生成worker process数等。

2、events块：配置影响nginx服务器或与用户的网络连接。有每个进程的最大连接数，选取哪种事件驱动模型处理连接请求，是否允许同时接受多个网路连接，开启多个网络连接序列化等。

3、http块：可以嵌套多个server，配置代理，缓存，日志定义等绝大多数功能和第三方模块的配置。如文件引入，mime-type定义，日志自定义，是否使用sendfile传输文件，连接超时时间，单连接请求数等。

4、server块：配置虚拟主机的相关参数，一个http中可以有多个server。

5、location块：配置请求的路由，以及各种页面的处理情况。

下面给大家上一个配置文件，作为理解，同时也配入我搭建的一台测试机中，给大家示例。

```
############ 每个指令必须有分号结束。#################
#user administrator administrators;  #配置用户或者组，默认为nobody nobody。
#worker_processes 2;  #允许生成的进程数，默认为1
#pid /nginx/pid/nginx.pid;   #指定nginx进程运行文件存放地址
error_log log/error.log debug;  #制定日志路径，级别。这个设置可以放入全局块，http块，server块，级别以此为：debug|info|notice|warn|error|crit|alert|emerg
events {
    accept_mutex on;   #设置网路连接序列化，防止惊群现象发生，默认为on
    multi_accept on;  #设置一个进程是否同时接受多个网络连接，默认为off
    #use epoll;      #事件驱动模型，select|poll|kqueue|epoll|resig|/dev/poll|eventport
    worker_connections  1024;    #最大连接数，默认为512
}
http {
    include       mime.types;   #文件扩展名与文件类型映射表
    default_type  application/octet-stream; #默认文件类型，默认为text/plain
    #access_log off; #取消服务日志
    log_format myFormat '$remote_addr–$remote_user [$time_local] $request $status $body_bytes_sent $http_referer $http_user_agent $http_x_forwarded_for'; #自定义格式
    access_log log/access.log myFormat;  #combined为日志格式的默认值
    sendfile on;   #允许sendfile方式传输文件，默认为off，可以在http块，server块，location块。
    sendfile_max_chunk 100k;  #每个进程每次调用传输数量不能大于设定的值，默认为0，即不设上限。
    keepalive_timeout 65;  #连接超时时间，默认为75s，可以在http，server，location块。

    upstream mysvr {
      server 127.0.0.1:7878;
      server 192.168.10.121:3333 backup;  #热备
    }
```

```
    error_page 404 https://www.baidu.com; #错误页
    server {
        keepalive_requests 120; #单连接请求上限次数。
        listen       4545;   #监听端口
        server_name  127.0.0.1;   #监听地址
        location  ~*^.+$ {       #请求的url过滤，正则匹配，~为区分大小写，~*为不区分大小写。
           #root path;  #根目录
           #index vv.txt;  #设置默认页
           proxy_pass  http://mysvr;  #请求转向mysvr 定义的服务器列表
           deny 127.0.0.1;  #拒绝的ip
           allow 172.18.5.54; #允许的ip
        }
    }
}
```

上面是nginx的基本配置，需要注意的有以下几点：

1、配置项

- 1.$remote_addr 与$http_x_forwarded_for 用以记录客户端的ip地址；
- 2.$remote_user： 用来记录客户端用户名称；
- 3.$time_local： 用来记录访问时间与时区；
- 4.$request： 用来记录请求的url与http协议；
- 5.$status： 用来记录请求状态；成功是200,
- 6.$body_bytes_s ent： 记录发送给客户端文件主体内容大小；
- 7.$http_referer： 用来记录从那个页面链接访问过来的；
- 8.$http_user_agent： 记录客户端浏览器的相关信息；

2、惊群现象：一个网路连接到来，多个睡眠的进程被同时叫醒，但只有一个进程能获得链接，这样会影响系统性能。

3、每个指令必须有分号结束。

Nginx代理服务的配置说明

1、上一篇中我们在http模块中有下面的配置，当代理遇到状态码为404时，我们把404页面导向百度。

```
error_page 404 https://www.baidu.com; #错误页
```

然而这个配置，细心的朋友可以发现他并没有起作用。

如果我们想让他起作用，我们必须配合着下面的配置一起使用

```
proxy_intercept_errors on;     #如果被代理服务器返回的状态码为400或者大于400，设置的error_page配置起作用。默认为off。
```

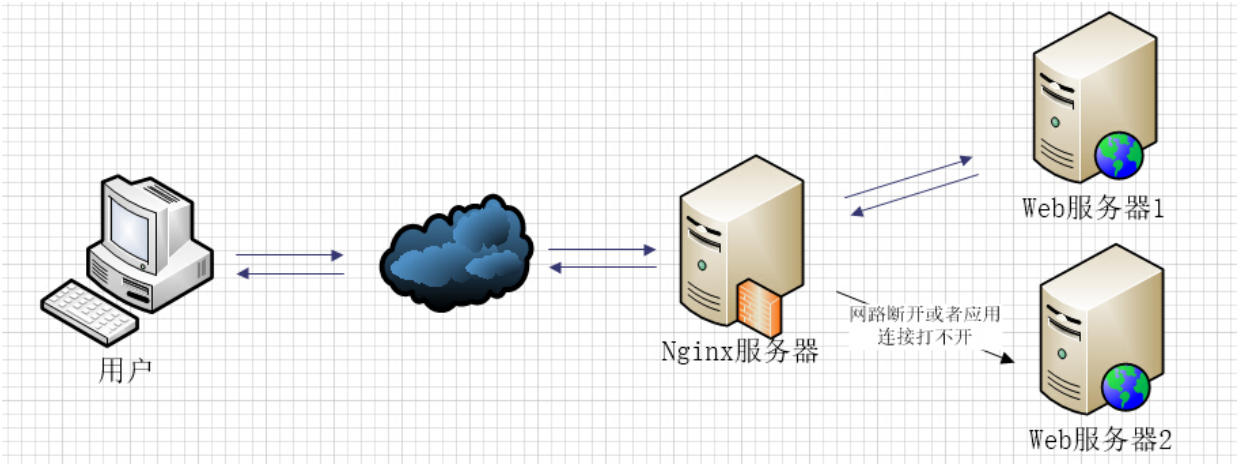
2、如果我们的代理只允许接受get，post请求方法的一种

```
proxy_method get;     #支持客户端的请求方法。post/get;
```

3、设置支持的http协议版本

```
proxy_http_version 1.0 ; #Nginx服务器提供代理服务的http协议版本1.0，1.1，默认设置为1.0版本
```

4、如果你的nginx服务器给2台web服务器做代理，负载均衡算法采用轮询，那么当你的一台机器web程序iis关闭，也就是说web不能访问，那么nginx服务器分发请求还是会给这台不能访问的web服务器，如果这里的响应连接时间过长，就会导致客户端的页面一直在等待响应，对用户来说体验就打打折扣，这里我们怎么避免这样的情况发生呢。这里我配张图来说明下问题。



如果负载均衡中其中web2发生这样的情况，nginx首先会去web1请求，但是nginx在配置不当的情况下会继续分发请求道web2，然后等待web2响应，直到我们的响应时间超时，才会把请求重新分发给web1，这里的响应时间如果过长，用户等待的时间就会越长。

下面的配置是解决方案之一。

```
proxy_connect_timeout 1;    #nginx服务器与被代理的服务器建立连接的超时时间，默认60秒
proxy_read_timeout 1; #nginx服务器想被代理服务器组发出read请求后，等待响应的超时时间，默认为60秒。
proxy_send_timeout 1; #nginx服务器想被代理服务器组发出write请求后，等待响应的超时时间，默认为60秒。
proxy_ignore_client_abort on;  #客户端断网时，nginx服务器是否终端对被代理服务器的请求。默认为off。
```

5、如果使用upstream指令配置啦一组服务器作为被代理服务器，服务器中的访问算法遵循配置的负载均衡规则，同时可以使用该指令配置在发生哪些异常情况时，将请求顺次交由下一组服务器处理。

```
proxy_next_upstream timeout;  #反向代理upstream中设置的服务器组，出现故障时，被代理服务器返回的状态值。
error|timeout|invalid_header|http_500|http_502|http_503|http_504|http_404|off
```

error：建立连接或向被代理的服务器发送请求或读取响应信息时服务器发生错误。

timeout：建立连接，想被代理服务器发送请求或读取响应信息时服务器发生超时。

invalid_header:被代理服务器返回的响应头异常。

off:无法将请求分发给被代理的服务器。

http_400，....:被代理服务器返回的状态码为400，500，502，等。

6、如果你想通过http获取客户的真是ip而不是获取代理服务器的ip地址，那么要做如下的设置。

```
proxy_set_header Host $host; #只要用户在浏览器中访问的域名绑定了 VIP VIP 下面有RS；则就用$host ； host是访问URL中的域名和端口  www.taobao.com:80
proxy_set_header X-Real-IP $remote_addr;  #把源IP 【$remote_addr,建立HTTP连接header里面的信息】赋值给X-Real-IP;这样在代码中 $X-Real-IP来获取 源IP
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;#在nginx 作为代理服务器时，设置的IP列表，会把经过的机器ip，代理机器ip都记录下来，用 【，】隔开；代码中用 echo $x-forwarded-for |awk -F, '{print $1}' 来作为源IP
```

关于X-Forwarded-For与X-Real-IP的一些相关文章我推荐一位博友的：HTTP 请求头中的 X-Forwarded-For ，这位博友对http协议有一系列的文章阐述，推荐大家去关注下。

7、下面是我的一个关于代理配置的配置文件部分，仅供参考。

```
    include       mime.types;   #文件扩展名与文件类型映射表
    default_type  application/octet-stream; #默认文件类型，默认为text/plain
    #access_log off; #取消服务日志
    log_format myFormat ' $remote_addr–$remote_user [$time_local] $request $status $body_bytes_sent $http_referer $http_user_agent $http_x_forwarded_for'; #自定义格式
    access_log log/access.log myFormat;  #combined为日志格式的默认值
    sendfile on;   #允许sendfile方式传输文件，默认为off，可以在http块，server块，location块。
    sendfile_max_chunk 100k;  #每个进程每次调用传输数量不能大于设定的值，默认为0，即不设上限。
    keepalive_timeout 65;  #连接超时时间，默认为75s，可以在http，server，location块。
    proxy_connect_timeout 1;   #nginx服务器与被代理的服务器建立连接的超时时间，默认60秒
    proxy_read_timeout 1; #nginx服务器想被代理服务器组发出read请求后，等待响应的超时间，默认为60秒。
    proxy_send_timeout 1; #nginx服务器想被代理服务器组发出write请求后，等待响应的超时间，默认为60秒。
```

```
12      proxy_http_version 1.0 ; #Nginx服务器提供代理服务的http协议版本1.0，1.1，默认设置为1.0版本。
13      #proxy_method get;     #支持客户端的请求方法。post/get；
14      proxy_ignore_client_abort on;  #客户端断网时，nginx服务器是否终端对被代理服务器的请求。默认为off。
15      proxy_ignore_headers "Expires" "Set-Cookie";  #Nginx服务器不处理设置的http相应投中的头域，这里空格隔开可以设置多个。
16      proxy_intercept_errors on;     #如果被代理服务器返回的状态码为400或者大于400，设置的error_page配置起作用。默认为off。
17      proxy_headers_hash_max_size 1024; #存放http报文头的哈希表容量上限，默认为512个字符。
18      proxy_headers_hash_bucket_size 128; #nginx服务器申请存放http报文头的哈希表容量大小。默认为64个字符。
19      proxy_next_upstream timeout;  #反向代理upstream中设置的服务器组，出现故障时，被代理服务器返回的状态值。error|timeout|invalid_header|http_500|http_502|http_503|http_504|http_404|off
20      #proxy_ssl_session_reuse on; 默认为on，如果我们在错误日志中发现“SSL3_GET_FINSHED:digest check failed”的情况时，可以将该指令设置为off。
```

Nginx负载均衡详解

上一篇中我说啦nginx有哪些中负载均衡算法。这一结我就给如果操作配置的给大家做详细说明下。

首先给大家说下upstream这个配置的，这个配置是写一组被代理的服务器地址，然后配置负载均衡的算法。这里的被代理服务器地址有2中写法。

```
upstream mysvr {
      server 192.168.10.121:3333;
      server 192.168.10.122:3333;
    }
 server {
        ....
        location  ~*^.+$ {
           proxy_pass  http://mysvr;  #请求转向mysvr 定义的服务器列表
        }
```

```
upstream mysvr {
      server  http://192.168.10.121:3333;
      server  http://192.168.10.122:3333;
    }
 server {
        ....
        location  ~*^.+$ {
           proxy_pass  mysvr;  #请求转向mysvr 定义的服务器列表
        }
```

然后，就来点实战的东西。

1、热备：如果你有2台服务器，当一台服务器发生事故时，才启用第二台服务器给提供服务。服务器处理请求的顺序：AAAAAA突然A挂啦，BBBBBBBBBBBBBB.....

```
upstream mysvr {
      server 127.0.0.1:7878;
      server 192.168.10.121:3333 backup;  #热备
    }
```

2、轮询：nginx默认就是轮询其权重都默认为1，服务器处理请求的顺序：ABABABABAB....

```
upstream mysvr {
      server 127.0.0.1:7878;
      server 192.168.10.121:3333;
    }
```

3、加权轮询：跟据配置的权重的大小而分发给不同服务器不同数量的请求。如果不设置，则默认为1。下面服务器的请求顺序为：ABBABBABBABBABB....

```
 upstream mysvr {
      server 127.0.0.1:7878 weight=1;
      server 192.168.10.121:3333 weight=2;
}
```

4、ip_hash:nginx会让相同的客户端ip请求相同的服务器。

```
upstream mysvr {
      server 127.0.0.1:7878;
      server 192.168.10.121:3333;
      ip_hash;
    }
```

5、如果你对上面4种均衡算法不是很理解，那么麻烦您去看下我上一篇配的图片，可能会更加容易理解点。

到这里你是不是感觉nginx的负载均衡配置特别简单与强大，那么还没完，咱们继续哈，这里扯下蛋。

关于nginx负载均衡配置的几个状态参数讲解。

- down，表示当前的server暂时不参与负载均衡。
- backup，预留的备份机器。当其他所有的非backup机器出现故障或者忙的时候，才会请求backup机器，因此这台机器的压力最轻。
- 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。
- fail_timeout，在经历了max_fails次失败后，暂停服务的时间。max_fails可以和fail_timeout一起使用。

```
upstream mysvr {
     server 127.0.0.1:7878 weight=2 max_fails=2 fail_timeout=2;
     server 192.168.10.121:3333 weight=1 max_fails=2 fail_timeout=1;
   }
```

到这里应该可以说nginx的内置负载均衡算法已经没有货啦。如果你想跟多更深入的了解nginx的负载均衡算法，nginx官方提供一些插件大家可以了解下。