Nginx 40 问！

mp.weixin.qq.com/s/TvxkuTu5HFOh-tLLS5Ypbw

> 来自：入门小站

什么是Nginx？

Nginx是一个 轻量级/高性能的反向代理Web服务器，用于 HTTP、HTTPS、SMTP、POP3 和 IMAP 协议。他实现非常高效的反向代理、负载平衡，他可以处理2-3万并发连接数，官方监测能支持5万并发，现在中国使用nginx网站用户有很多，例如：新浪、网易、腾讯等。

Nginx 有哪些优点？

- 跨平台、配置简单。
- 非阻塞、高并发连接：处理 2-3 万并发连接数，官方监测能支持 5 万并发。
- 内存消耗小：开启 10 个 Nginx 才占 150M 内存。

- 成本低廉，且开源。
- 稳定性高，宕机的概率非常小。
- 内置的健康检查功能：如果有一个服务器宕机，会做一个健康检查，再发送的请求就不会发送到宕机的服务器了。重新将请求提交到其他的节点上

Nginx应用场景？

- http服务器。Nginx是一个http服务可以独立提供http服务。可以做网页静态服务器。
- 虚拟主机。可以实现在一台服务器虚拟出多个网站，例如个人网站使用的虚拟机。
- 反向代理，负载均衡。当网站的访问量达到一定程度后，单台服务器不能满足用户的请求时，需要用多台服务器集群可以使用nginx做反向代理。并且多台服务器可以平均分担负载，不会应为某台服务器负载高宕机而某台服务器闲置的情况。
- nginz 中也可以配置安全管理、比如可以使用Nginx搭建API接口网关,对每个接口服务进行拦截。

Nginx怎么处理请求的？

```
server {                                        # 第一个Server区块开始，表示一个独立的虚拟主机站点
    listen       80;                            # 提供服务的端口，默认80
    server_name  localhost;                     # 提供服务的域名主机名
    location / {                                # 第一个location区块开始
        root   html;                            # 站点的根目录，相当于Nginx的安装目录
        index  index.html index.html;           # 默认的首页文件，多个用空格分开
    }                                           # 第一个location区块结果
```

首先，Nginx 在启动时，会解析配置文件，得到需要监听的端口与 IP 地址，然后在 Nginx 的 Master 进程里面先初始化好这个监控的Socket(创建 S ocket，设置 addr、reuse 等选项，绑定到指定的 ip 地址端口，再 listen 监听)。

然后，再 fork(一个现有进程可以调用 fork 函数创建一个新进程。由 fork 创建的新进程被称为子进程)出多个子进程出来。

之后，子进程会竞争 accept 新的连接。此时，客户端就可以向 nginx 发起连接了。当客户端与nginx进行三次握手，与 nginx 建立好一个连接后。此时，某一个子进程会 accept 成功，得到这个建立好的连接的 Socket ，然后创建 nginx 对连接的封装，即 ngx_connection_t 结构体。

接着，设置读写事件处理函数，并添加读写事件来与客户端进行数据的交换。

最后，Nginx 或客户端来主动关掉连接，到此，一个连接就寿终正寝了。

Nginx 是如何实现高并发的？

如果一个 server 采用一个进程(或者线程)负责一个request的方式，那么进程数就是并发数。那么显而易见的，就是会有很多进程在等待中。等什么？最多的应该是等待网络传输。

而 Nginx 的异步非阻塞工作方式正是利用了这点等待的时间。在需要等待的时候，这些进程就空闲出来待命了。因此表现为少数几个进程就解决了大量的并发问题。

Nginx是如何利用的呢，简单来说：同样的 4 个进程，如果采用一个进程负责一个 request 的方式，那么，同时进来 4 个 request 之后，每个进程就负责其中一个，直至会话关闭。期间，如果有第 5 个request进来了。就无法及时反应了，因为 4 个进程都没干完活呢，因此，一般有个调度进程，每当新进来了一个 request ，就新开个进程来处理。

回想下，BIO 是不是存在酱紫的问题？

Nginx 不这样，每进来一个 request ，会有一个 worker 进程去处理。但不是全程的处理，处理到什么程度呢？处理到可能发生阻塞的地方，比如向上游（后端）服务器转发 request ，并等待请求返回。那么，这个处理的 worker 不会这么傻等着，他会在发送完请求后，注册一个事件：如果 upstream 返回了，告诉我一声，我再接着干。于是他就休息去了。此时，如果再有 request 进来，他就可以很快再按这种方式处理。而一旦上游服务器返回了，就会触发这个事件，worker 才会来接手，这个 request 才会接着往下走。

这就是为什么说，Nginx 基于事件模型。

由于 web server 的工作性质决定了每个 request 的大部份生命都是在网络传输中，实际上花费在 server 机器上的时间片不多。这是几个进程就解决高并发的秘密所在。即：

webserver 刚好属于网络 IO 密集型应用，不算是计算密集型。

异步，非阻塞，使用 epoll ，和大量细节处的优化。也正是 Nginx 之所以然的技术基石。

什么是正向代理？

一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。

客户端才能使用正向代理。正向代理总结就一句话：代理端代理的是客户端。例如说：我们使用的OpenVPN 等等。

什么是反向代理？

反向代理（Reverse Proxy）方式，是指以代理服务器来接受 Internet上的连接请求，然后将请求，发给内部网络上的服务器并将从服务器上得到的结果返回给 Internet 上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器。

反向代理总结就一句话：代理端代理的是服务端。

反向代理服务器的优点是什么?

反向代理服务器可以隐藏源服务器的存在和特征。它充当互联网云和web服务器之间的中间层。这对于安全方面来说是很好的，特别是当您使用web托管服务时。

Nginx目录结构有哪些？

```
[root@localhost ~]# tree /usr/local/nginx
/usr/local/nginx
├── client_body_temp
├── conf                                # Nginx所有配置文件的目录
│   ├── fastcgi.conf                    # fastcgi相关参数的配置文件
│   ├── fastcgi.conf.default            # fastcgi.conf的原始备份文件
│   ├── fastcgi_params                  # fastcgi的参数文件
│   ├── fastcgi_params.default
│   ├── koi-utf
│   ├── koi-win
│   ├── mime.types                      # 媒体类型
│   ├── mime.types.default
│   ├── nginx.conf                      # Nginx主配置文件
│   ├── nginx.conf.default
│   ├── scgi_params                     # scgi相关参数文件
│   ├── scgi_params.default
│   ├── uwsgi_params                    # uwsgi相关参数文件
│   ├── uwsgi_params.default
│   └── win-utf
├── fastcgi_temp                        # fastcgi临时数据目录
├── html                                # Nginx默认站点目录
│   ├── 50x.html                        # 错误页面优雅替代显示文件，例如当出现502错误时会调用此页面
│   └── index.html                      # 默认的首页文件
├── logs                                # Nginx日志目录
│   ├── access.log                      # 访问日志文件
│   ├── error.log                       # 错误日志文件
│   └── nginx.pid                       # pid文件，Nginx进程启动后，会把所有进程的ID号写到此文件
├── proxy_temp                          # 临时目录
├── sbin                                # Nginx命令目录
│   └── nginx                           # Nginx的启动命令
├── scgi_temp                           # 临时目录
└── uwsgi_temp                          # 临时目录
```

Nginx配置文件nginx.conf有哪些属性模块?

```
worker_processes1；                                    # worker进程的数量


  events {                                                 # 事件区块开始

worker_connections1024；                               # 每个worker进程支持的最大连接数


  }                                                        # 事件区块结束

  http {                                                   # HTTP区块开始

include       mime.types；                               # Nginx支持的媒体类型库文件
default_type  application/octet-stream；                 # 默认的媒体类型
sendfile        on；                                     # 开启高效传输模式
keepalive_timeout  65；                                  # 连接超时
server {                                                 # 第一个Server区块开始，表示一个独
立的虚拟主机站点
listen80；                                          # 提供服务的端口，默认80
    server_name  localhost；                             # 提供服务的域名主机名
    location / {                                         # 第一个location区块开始
root   html；                                       # 站点的根目录，相当于Nginx的安装目录
        index  index.html index.htm；                    # 默认的首页文件，多个用空格分开
    }                                                    # 第一个location区块结果


    error_page   500502503504  /50x.html；               # 出现对应的http状态码时，使用
50x.html回应客户
    location = /50x.html {                               # location区块开始，访问50x.html
root   html；                                       # 指定对应的站点目录为html
    }
}

        ......
```

cookie和session区别？

共同:
存放用户信息。存放的形式：key-value格式 变量和变量内容键值对。

区别:
cookie

- 存放在客户端浏览器
- 每个域名对应一个cookie，不能跨跃域名访问其他cookie
- 用户可以查看或修改cookie
- http响应报文里面给你浏览器设置
- 钥匙（用于打开浏览器上锁头）

session:

- 存放在服务器（文件，数据库，redis）
- 存放敏感信息
- 锁头

为什么 Nginx 不使用多线程？

Apache: 创建多个进程或线程，而每个进程或线程都会为其分配 cpu 和内存（线程要比进程小的多，所以 worker 支持比 perfork 高的并发），并发过大会榨干服务器资源。

Nginx: 采用单线程来异步非阻塞处理请求（管理员可以配置 Nginx 主进程的工作进程的数量）(epoll)，不会为每个请求分配 cpu 和内存资源，节省了大量资源，同时也减少了大量的 CPU 的上下文切换。所以才使得 Nginx 支持更高的并发。

nginx和apache的区别

轻量级，同样起web服务，比apache占用更少的内存和资源。

抗并发，nginx处理请求是异步非阻塞的，而apache则是阻塞性的，在高并发下nginx能保持低资源，低消耗高性能。

高度模块化的设计，编写模块相对简单。

最核心的区别在于apache是同步多进程模型，一个连接对应一个进程，nginx是异步的，多个连接可以对应一个进程。

Nginx	Apache
nginx是一个基于事件的web服务器	apache是一个基于流程的服务器
所有请求都由一个线程处理	单个线程处理单个请求
nginx避免子进程的概念	apache是基于子进程的
nginx类似于速度	apache类似于功率
nginx在内存消耗和连接方面比较好	apache在内存消耗和连接上没有提高
nginx在负载均衡方面表现较好	当流量到达进程极限时，apache将拒绝新的连接。
对于php来说，nginx可能更可取，因为它支持php	apache支持php，python，perl和其他语言使用插件，当应用程序基于python或ruby时，它非常有用。
nginx不支持IBMI和openvms一样的os	apache支持更多的os
nginx只具有核心功能	apache提供了比nginx更多的功能

Nginx	Apache
nginx的性能和可伸缩性不依赖于硬件	apache依赖于cpu和内存等硬件组件。

什么是动态资源、静态资源分离？

动态资源、静态资源分离，是让动态网站里的动态网页根据一定规则把不变的资源和经常变的资源区分开来，动静资源做好了拆分以后我们就可以根据静态资源的特点将其做缓存操作，这就是网站静态化处理的核心思路。

动态资源、静态资源分离简单的概括是：动态文件与静态文件的分离。

为什么要做动、静分离？

在我们的软件开发中，有些请求是需要后台处理的（如：.jsp,.do 等等），有些请求是不需要经过后台处理的（如：css、html、jpg、js 等等文件），这些不需要经过后台处理的文件称为静态文件，否则动态文件。

因此我们后台处理忽略静态文件。这会有人又说那我后台忽略静态文件不就完了吗？当然这是可以的，但是这样后台的请求次数就明显增多了。在我们对资源的响应速度有要求的时候，我们应该使用这种动静分离的策略去解决动、静分离将网站静态资源（HTML，JavaScript，CSS，img等文件）与后台应用分开部署，提高用户访问静态代码的速度，降低对后台应用访问

这里我们将静态资源放到 Nginx 中，动态资源转发到 Tomcat 服务器中去。

当然，因为现在七牛、阿里云等 CDN 服务已经很成熟，主流的做法，是把静态资源缓存到 CDN 服务中，从而提升访问速度。

相比本地的 Nginx 来说，CDN 服务器由于在国内有更多的节点，可以实现用户的就近访问。并且，CDN 服务可以提供更大的带宽，不像我们自己的应用服务，提供的带宽是有限的。

什么叫 CDN 服务？

CDN ，即内容分发网络。

其目的是，通过在现有的 Internet中 增加一层新的网络架构，将网站的内容发布到最接近用户的网络边缘，使用户可就近取得所需的内容，提高用户访问网站的速度。

一般来说，因为现在 CDN 服务比较大众，所以基本所有公司都会使用 CDN 服务。

Nginx怎么做的动静分离？

只需要指定路径对应的目录。location/可以使用正则表达式匹配。并指定对应的硬盘中的目录。如下：（操作都是在Linux上）

```
location /image/ {
        root   /usr/local/static/;
        autoindex on;
}
```

步骤：

```
# 创建目录
mkdir /usr/local/static/image
# 进入目录
cd  /usr/local/static/image
# 上传照片
photo.jpg
# 重启nginx
sudo nginx -s reload
```

打开浏览器 输入 server_name/image/1.jpg 就可以访问该静态图片了

Nginx负载均衡的算法怎么实现的?策略有哪些?

为了避免服务器崩溃，大家会通过负载均衡的方式来分担服务器压力。将对台服务器组成一个集群，当用户访问时，先访问到一个转发服务器，再由转发服务器将访问分发到压力更小的服务器。

Nginx负载均衡实现的策略有以下五种：

> 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，如果后端某个服务器宕机，能自动剔除故障系统。

```
upstream backserver {
server1112;
server1113;

          }
```

> 权重 weight的值越大，分配到的访问概率越高，主要用于后端每台服务器性能不均衡的情况下。其次是为在主从的情况下设置不同的权值，达到合理有效的地利用主机资源。

```
# 权重越高，在被访问的概率越大，如上例，分别是20%，80%。

upstream backserver {
 server 1112 weight=2;
 server 1113 weight=8;
}
```

-ip_hash(IP绑定) 每个请求按访问IP的哈希结果分配，使来自同一个IP的访客固定访问一台后端服务器，并且可以有效解决动态网页存在的session共享问题

```
upstream backserver {

   ip_hash;
   server 1112:88;
   server 1113:80;
  }
```

fair(第三方插件)

必须安装upstream_fair模块。

对比 weight、ip_hash更加智能的负载均衡算法，fair算法可以根据页面大小和加载时间长短智能地进行负载均衡，响应时间短的优先分配。

```
# 哪个服务器的响应速度快，就将请求分配到那个服务器上。

upstream backserver {
 server server1;
 server server2;
 fair;
}
```

url_hash(第三方插件)

必须安装Nginx的hash软件包

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，可以进一步提高后端缓存服务器的效率。

```
upstream backserver {
   server squid1:3128;
   server squid2:3128;
   hash $request_uri;
   hash_method crc32;
}
```

如何用Nginx解决前端跨域问题？

使用Nginx转发请求。把跨域的接口写成调本域的接口，然后将这些接口转发到真正的请求地址。

Nginx虚拟主机怎么配置?`

1、基于域名的虚拟主机，通过域名来区分虚拟主机——应用：外部网站

2、基于端口的虚拟主机，通过端口来区分虚拟主机——应用：公司内部网站，外部网站的管理后台

3、基于ip的虚拟主机。

基于虚拟主机配置域名

需要建立/data/www /data/bbs目录，windows本地hosts添加虚拟机ip地址对应的域名解析；对应域名网站目录下新增index.html文件；

```
# 当客户端访问www.rumenz.com,监听端口号为80,直接跳转到data/www目录下文件
server {
listen80;
server_name  www.rumenz.com;
location / {
root   data/www;
index  index.html index.htm;
    }
}
```

```
                                        # 当客户端访问www.rumenz.com,监听端口号为80,直接跳转到data/bbs目录下文件
     server {
    listen       80;
    server_name  bbs.rumenz.com;
    location / {
        root   data/bbs;
        index  index.html index.htm;
    }
}
```

基于端口的虚拟主机

使用端口来区分，浏览器使用域名或ip地址:端口号 访问

```
# 当客户端访问www.rumenz.com,监听端口号为8080,直接跳转到data/www目录下文件
server {
listen8080;
server_name  rumenz.com;
location / {
root   data/www;
index  index.html index.htm;
    }
}
```

```
                            # 当客户端访问www.rumenz.com,监听端口号为80直接跳转到真实ip服务器地址 11:8080
server {
listen       80;
server_name  www.rumenz.com;
    location / {
                  proxy_pass http://11:8080;
         index  index.html index.htm;
    }
}
```

location的作用是什么？

location指令的作用是根据用户请求的URI来执行不同的应用，也就是根据用户请求的网站URL进行匹配，匹配成功即进行相关的操作。

注意：~ 代表自己输入的英文字母

匹配符	匹配规则	优先级
=	精确匹配	1
^~	以某个字符串开头	2
~	区分大小写的正则匹配	3
~*	不区分大小写的正则匹配	4
!~	区分大小写不匹配的正则	5
!~*	不区分大小写不匹配的正则	6
/	通用匹配，任何请求都会匹配到	7

Location正则案例

```
# 优先级1,精确匹配，根路径

location =/ {
return400;
}
# 优先级2,以某个字符串开头,以av开头的，优先匹配这里，区分大小写
location ^~ /av {
root /data/av/;
}



# 优先级3，区分大小写的正则匹配，匹配/media*****路径
location~ /media {
alias /data/static/;
}



# 优先级4 ，不区分大小写的正则匹配，所有的****.jpg|gif|png 都走这里
location~* .*\.(jpg|gif|png|js|css)$ {
root  /data/av/;
}



# 优先7，通用匹配
location / {
return403;

     }
```

限流怎么做的?

Nginx限流就是限制用户请求速度，防止服务器受不了

限流有3种

- 正常限制访问频率（正常流量）
- 突发限制访问频率（突发流量）
- 限制并发连接数

Nginx的限流都是基于漏桶流算法

实现三种限流算法

正常限制访问频率（正常流量）：

限制一个用户发送的请求，我Nginx多久接收一个请求。

Nginx中使用ngx_http_limit_req_module模块来限制的访问频率，限制的原理实质是基于漏桶算法原理来实现的。在nginx.conf配置文件中可以使用limit_req_zone命令及limit_req命令限制单个IP的请求处理频率。

```
# 定义限流维度，一个用户一分钟一个请求进来，多余的全部漏掉
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/m;
# 绑定限流维度
server{
    location /seckill.html{
        limit_req zone=zone;
        proxy_pass http://lj_seckill;
    }
}
```

1r/s代表1秒一个请求，1r/m一分钟接收一个请求， 如果Nginx这时还有别人的请求没有处理完，Nginx就会拒绝处理该用户请求。

突发限制访问频率（突发流量）：

限制一个用户发送的请求，我Nginx多久接收一个。

上面的配置一定程度可以限制访问频率，但是也存在着一个问题：如果突发流量超出请求被拒绝处理，无法处理活动时候的突发流量，这时候应该如何进一步处理呢？Nginx提供burst参数结合nodelay参数可以解决流量突发的问题，可以设置能处理的超过设置的请求数外能额外处理的请求数。我们可以将之前的例子添加burst参数以及nodelay参数：

```
# 定义限流维度，一个用户一分钟一个请求进来，多余的全部漏掉
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/m;
# 绑定限流维度
server{
    location/seckill.html{
        limit_req zone=zone burst=5 nodelay;
        proxy_pass http://lj_seckill;
    }

}
```

为什么就多了一个 burst=5 nodelay; 呢，多了这个可以代表Nginx对于一个用户的请求会立即处理前五个，多余的就慢慢来落，没有其他用户的请求我就处理你的，有其他的请求的话我Nginx就漏掉不接受你的请求

限制并发连接数

Nginx中的ngx_http_limit_conn_module模块提供了限制并发连接数的功能，可以使用limit_conn_zone指令以及limit_conn执行进行配置。接下来我们可以通过一个简单的例子来看下：

```
http {
limit_conn_zone$binary_remote_addr zone=myip:10m;
limit_conn_zone$server_name zone=myServerName:10m;
}
server {
location / {
limit_conn myip 10;
limit_conn myServerName 100;
rewrite / http://www.rumenz.net permanent;
    }

                                                    }
```

上面配置了单个IP同时并发连接数最多只能10个连接，并且设置了整个虚拟服务器同时最大并发数最多只能100个链接。当然，只有当请求的header被服务器处理后，虚拟服务器的连接数才会计数。刚才有提到过Nginx是基于漏桶算法原理实现的，实际上限流一般都是基于漏桶算法和令牌桶算法实现的。

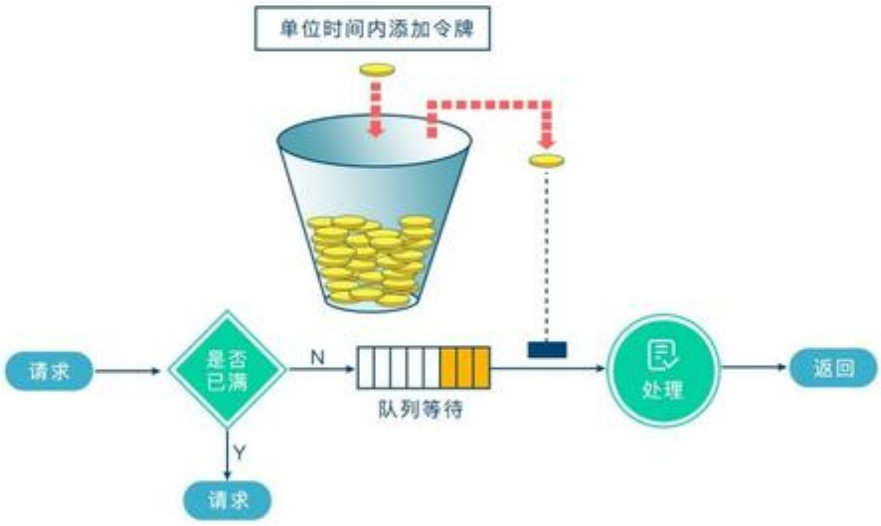
漏桶流算法和令牌桶算法知道？

一文搞定，手撸Springboot + aop + Lua分布式限流的最佳实践_vincent-CSDN博客_springboot如何限流

漏桶算法:漏桶算法思路很简单，我们把水比作是请求，漏桶比作是系统处理能力极限，水先进入到漏桶里，漏桶里的水按一定速率流出，当流出的速率小于流入的速率时，由于漏桶容量有限，后续进入的水直接溢出（拒绝请求），以此实现限流。

令牌桶算法:令牌桶算法的原理也比较简单，我们可以理解成医院的挂号看病，只有拿到号以后才可以进行诊病。

系统会维护一个令牌（token）桶，以一个恒定的速度往桶里放入令牌（token），这时如果有请求进来想要被处理，则需要先从桶里获取一个令牌（token），当桶里没有令牌（token）可取时，则该请求将被拒绝服务。令牌桶算法通过控制桶的容量、发放令牌的速率，来达到对请求的限制。



Nginx配置高可用性怎么配置？

当上游服务器(真实访问服务器)，一旦出现故障或者是没有及时相应的话，应该直接轮训到下一台服务器，保证服务器的高可用

Nginx配置代码：

```
server {
listen80;
server_name  www.rumenz.com;
location / {
### 指定上游服务器负载均衡服务器
proxy_pass http://backServer;
###nginx与上游服务器(真实访问的服务器)超时时间 后端服务器连接的超时时间_发起握手等候响应超时时间
proxy_connect_timeout1s;
###nginx发送给上游服务器(真实访问的服务器)超时时间
proxy_send_timeout1s;
### nginx接受上游服务器(真实访问的服务器)超时时间
proxy_read_timeout1s;
index  index.html index.htm;


}
```

Nginx怎么判断别IP不可访问？

```
# 如果访问的ip地址为11115,则返回403
 if  ($remote_addr = 11115) {
     return 403;
 }
```

在nginx中，如何使用未定义的服务器名称来阻止处理请求？

只需将请求删除的服务器就可以定义为：

服务器名被保留一个空字符串，他在没有主机头字段的情况下匹配请求，而一个特殊的nginx的非标准代码被返回，从而终止连接。

怎么限制浏览器访问？

```
## 不允许谷歌浏览器访问 如果是谷歌浏览器返回500
if ($http_user_agent ~ Chrome) {
    return 500;
}
```

Rewrite全局变量是什么？

```
$remote_addr         //获取客户端ip
$binary_remote_addr //客户端ip（二进制)
$remote_port         //客户端port，如：50472
$remote_user         //已经经过Auth Basic Module验证的用户名
$host            //请求主机头字段，否则为服务器名称，如:blog.sakmon.com
$request         //用户请求信息，如：GET ?a=1&b=2 HTTP/1.1
$request_filename    //当前请求的文件的路径名，由root或alias和URI request组合而成，
如：/2013/81.html
$status          //请求的响应状态码,如:200
$body_bytes_sent         // 响应时送出的body字节数数量。即使连接中断，这个数据也是精确的,
如：40
$content_length         // 等于请求行的“Content_Length”的值
$content_type           // 等于请求行的“Content_Type”的值
$http_referer           // 引用地址
$http_user_agent       // 客户端agent信息,如：Mozilla/5.0 (Windows NT 5.1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/29.0.1547.76 Safari/537.36
$args             //与$query_string相同 等于当中URL的参数(GET)，如a=1&b=2
$document_uri         //与$uri相同  这个变量指当前的请求URI，不包括任何参数(见
$args) 如:/2013/81.html
$document_root        //针对当前请求的根路径设置值
$hostname         //如：centos53.localdomain
$http_cookie         //客户端cookie信息
$cookie_COOKIE       //cookie COOKIE变量的值
$is_args    //如果有$args参数，这个变量等于”?”，否则等于”"，空值，如?
$limit_rate //这个变量可以限制连接速率，0表示不限速
$query_string        // 与$args相同 等于当中URL的参数(GET)，如a=1&b=2
$request_body       // 记录POST过来的数据信息
$request_body_file  //客户端请求主体信息的临时文件名
$request_method        //客户端请求的动作，通常为GET或POST,如：GET
$request_uri           //包含请求参数的原始URI，不包含主机名，如：/2013/81.html?a=1&b=2
$scheme             //HTTP方法（如http，https）,如：http
$uri             //这个变量指当前的请求URI，不包括任何参数(见$args) 如:/2013/81.html
$request_completion //如果请求结束，设置为OK. 当请求未结束或如果该请求不是请求链串的最后一个
时，为空(Empty)，如：OK
$server_protocol    //请求使用的协议，通常是HTTP/1.0或HTTP/1.1，如：HTTP/1.1
$server_addr         //服务器IP地址，在完成一次系统调用后可以确定这个值
$server_name         //服务器名称，如：blog.sakmon.com
$server_port         //请求到达服务器的端口号,如：80
```

Nginx 如何实现后端服务的健康检查？

方式一，利用 nginx 自带模块 ngx_http_proxy_module 和 ngx_http_upstream_module 对后端节点做健康检查。

方式二(推荐)，利用 nginx_upstream_check_module 模块对后端节点做健康检查。

Nginx 如何开启压缩？

开启nginx gzip压缩后，网页、css、js等静态资源的大小会大大的减少，从而可以节约大量的带宽，提高传输效率，给用户快的体验。虽然会消耗cpu资源，但是为了给用户更好的体验是值得的。

开启的配置如下：

将以上配置放到nginx.conf的http{ ... }节点中。

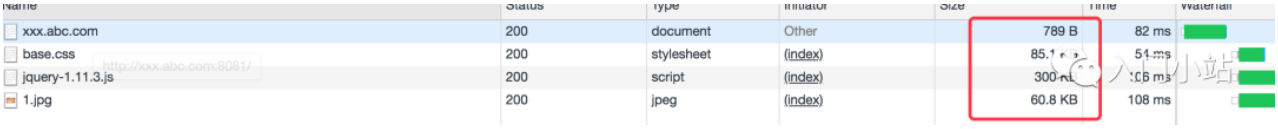
```
http {


# 开启gzip
gzipon;
# 启用gzip压缩的最小文件；小于设置值的文件将不会被压缩
gzip_min_length1k;
# gzip 压缩级别 1-10
gzip_comp_level2;
# 进行压缩的文件类型。
gzip_types text/plain application/javascript application/x-
javascript text/css application/xml text/javascript application/x-httpd-
php image/jpeg image/gif image/png;
# 是否在http header中添加Vary: Accept-Encoding，建议开启
gzip_varyon;

                                                    }
```

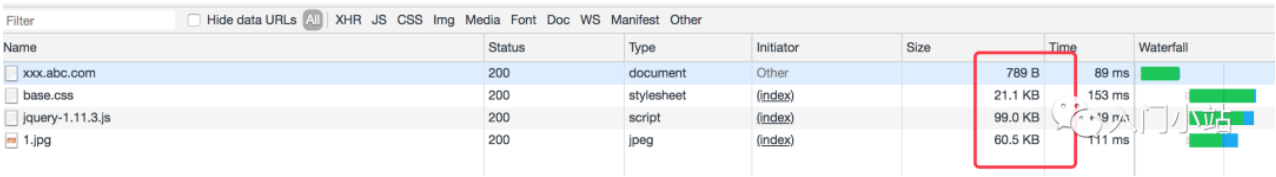
保存并重启nginx，刷新页面（为了避免缓存，请强制刷新）就能看到效果了。以谷歌浏览器为例，通过F12看请求的响应头部：

我们可以先来对比下，如果我们没有开启zip压缩之前，我们的对应的文件大小，如下所示：



Name	Status	Type	Initiator	Size	Time	Waterfall
xxx.abc.com	200	document	Other	789 B	82 ms	
base.css	200	stylesheet	(index)	85.1	51 ms	
jquery-1.11.3.js	200	script	(index)	300 KB	108 ms	
1.jpg	200	jpeg	(index)	60.8 KB	108 ms	

现在我们开启了gzip进行压缩后的文件的大小，可以看到如下所示：



Filter | Hide data URLs | All | XHR | JS | CSS | Img | Media | Font | Doc | WS | Manifest | Other

Name	Status	Type	Initiator	Size	Time	Waterfall
xxx.abc.com	200	document	Other	789 B	89 ms	
base.css	200	stylesheet	(index)	21.1 KB	153 ms	
jquery-1.11.3.js	200	script	(index)	99.0 KB	99 ms	
1.jpg	200	jpeg	(index)	60.5 KB	111 ms	

并且我们查看响应头会看到gzip这样的压缩，如下所示

gzip压缩前后效果对比：jquery原大小90kb，压缩后只有30kb。

gzip虽然好用，但是以下类型的资源不建议启用。

图片类型

原因：图片如jpg、png本身就会有压缩，所以就算开启gzip后，压缩前和压缩后大小没有多大区别，所以开启了反而会白白的浪费资源。（Tips：可以试试将一张jpg图片压缩为zip，观察大小并没有多大的变化。虽然zip和gzip算法不一样，但是可以看出压缩图片的价值并不大）

大文件

原因：会消耗大量的cpu资源，且不一定有明显的效果。

ngx_http_upstream_module的作用是什么?

ngx_http_upstream_module用于定义可通过fastcgi传递、proxy传递、uwsgi传递、memcached传递和scgi传递指令来引用的服务器组。

什么是C10K问题?

C10K问题是指无法同时处理大量客户端(10,000)的网络套接字。

Nginx是否支持将请求压缩到上游?

您可以使用Nginx模块gunzip将请求压缩到上游。gunzip模块是一个过滤器，它可以对不支持gzip编码方法的客户机或服务器使用内容编码:gzip来解压缩响应。

如何在Nginx中获得当前的时间?

要获得Nginx的当前时间，必须使用SSI模块、`$date_gmt`和date_local的变量。

Proxy_set_header THE-TIME $date_gmt;

用Nginx服务器解释-s的目的是什么?

用于运行Nginx -s参数的可执行文件。

如何在Nginx服务器上添加模块?

在编译过程中，必须选择Nginx模块，因为Nginx不支持模块的运行时间选择。

生产中如何设置worker进程的数量呢?

在有多个cpu的情况下，可以设置多个worker，worker进程的数量可以设置到和cpu的核心数一样多，如果在单个cpu上起多个worker进程，那么操作系统会在多个worker之间进行调度，这种情况会降低系统性能，如果只有一个cpu，那么只启动一个worker进程就可以了。

nginx状态码

- 499：服务端处理时间过长，客户端主动关闭了连接。
- 502：服务器错误

下面是502的一些可能性

- (1).FastCGI进程是否已经启动
- (2).FastCGI worker进程数是否不够
- (3).FastCGI执行时间过长

```
fastcgi_connect_timeout 300;
fastcgi_send_timeout 300;
fastcgi_read_timeout 300;
```

(4).FastCGI Buffer不够，nginx和apache一样，有前端缓冲限制，可以调整缓冲参数

```
fastcgi_buffer_size 32k;
fastcgi_buffers 8 32k;
```

(5). Proxy Buffer不够，如果你用了Proxying，调整

```
proxy_buffer_size 16k;
proxy_buffers 4 16k;
```

(6).php脚本执行时间过长

将php-fpm.conf的os的os改成一个时间

本公众号全部博文已整理成一个目录，请在公众号里回复「m」获取！
推荐阅读：

Linux 超级漂亮的 Shell

18000 字的 SQL 优化大全，收藏直接起飞！

45 个 Git 经典操作场景，专治不会合代码

5T技术资源大放送！包括但不限于：C/C++，Linux，Python，Java，PHP，人工智能，单片机，树莓派，等等。在公众号内回复「1024」，即可免费获取！！

良许Linux
良许，自学转行IT并顺利进入500强外企担任Linux开发工程师。公众号分享大量Linux干货，包括Linux基础、Linux应用、Linux工具软件，以及Git、数据库、树莓派等方面技术知识（后台回复 Linux 获取必备Linux资源）

224篇原创内容

公众号