

NGINX

1. 高并发场景下的问题

在开发好的项目真正上线后，一旦访问的并发量上升，由于大量的并发涌入，往往会造成服务器运行时的各种问题，包括服务器并发压力、数据库访问压力、数据库数据量限制等等。我们将在互联网架构阶段讨论问题的产生和解决的方案。

2. 服务器并发压力

单台tomcat在理想情况下可支持的最大并发数量在200~500之间，如果大于这个数量可能会造成响应缓慢甚至宕机。

解决方案是通过多台服务器分摊并发压力，这不仅需要有多台tomcat服务器，这就需要有一台服务器专门用来分配压力，我们称之为反向代理服务器程序。

3. Nginx概述

Nginx是一款优秀的反向代理服务器程序，能够为若干台服务器提供反向代理服务，一方面为客户端提供的统一的访问地址，另一方面为后台多个服务器提供了负载均衡的能力。

Nginx是目前最主流的反向代理服务器，能够提供可靠的负载均衡、动静分离的能力。

4. Nginx的安装配置

认准一手QQ3195303913微信wxywd8

a. 下载 <http://nginx.org/en/download.html>

b. 安装 将下载好的安装包解压到一个没有中文没有空格的目录下即可。

c. 命令

验证配置是否正确: `nginx -t`

查看Nginx的版本号: `nginx -V`

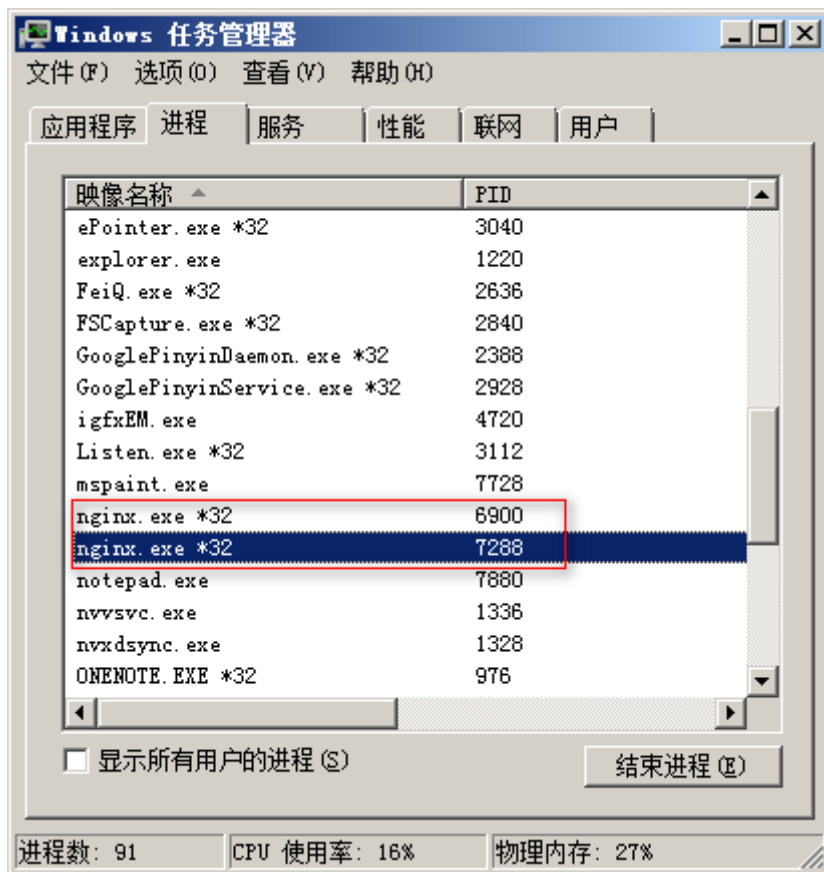
启动Nginx: `start nginx`

快速停止或关闭Nginx: `nginx -s stop`

正常停止或关闭Nginx: `nginx -s quit`

配置文件修改重载命令: `nginx -s reload`

nginx是否在运行可以在windows的任务管理器中查看，正在运行ngxin会由两个进程显示。



d.配置

nginx的工作是基于[conf/nginx.conf]配置文件来进行的。

nginx.conf的配置结构:

```
http{ #代表处理http请求
    #配置一个虚拟服务器
    server{
        #此虚拟服务器接收对80端口的访问
        listen 80;
        #此虚拟服务器接收对localhost主机名的访问
        server_name localhost;
        #当访问/user资源时由此配置处理
        location /user{
            规则
        }
        #当访问/order资源时由此配置处理
        location /order{
            规则
        }
        ...
    }

    #其他Server配置
    server ...
    ...
}
```

5.入门案例 - 通过Nginx实现请求转发

当客户端访问<http://www.pq.com>时，由nginx转发给<http://127.0.0.1:8080>端口进行处理

配置hosts文件

127.0.0.1 www.pq.com

在nginx.conf中配置

```
http{
    #为nginx配置一个虚拟服务器，
    server {
        #监听本机80端口
        listen 80;
        #接收对www.pq.com主机名的访问
        server_name www.pq.com;
        #对/即任意路径的访问进行处理
        location / {
            #转发到指定地址
            proxy_pass http://127.0.0.1:8080;
        }
        #可以配置多个location
        ...
    }
    #可以配置多个server
    ...
}
```

认准一手QQ3195303913微信wxywd8

6.location路径配置和匹配规则

a.location路径的写法

在配置虚拟服务器时，可以配置多个location，指定不同路径采用不同的处理方案，location支持多种写法，规则如下：

=	<code>=/aaa/1.jpg</code>	路径严格匹配，路径必须一模一样才会匹配到
<code>^~</code>	<code>^~/aaa</code>	只要是指定路径开头的路径都可以匹配
<code>~</code>	<code>~.png\$</code>	区分大小写按正则匹配路径
<code>~*</code>	<code>~*.png\$</code>	不区分大小写按正则匹配路径
/	/	通用匹配，所有路径都可以匹配到

b.location路径配置的优先级

由于location的路径配置非常灵活，所有有可能一个路径被多个location所匹配，此时按照如下规则判断匹配优先级：

- 首先匹配 =
 - 其次匹配 ^~
 - 其次是按文件中顺序的正则匹配
 - 最后是交给 / 通用匹配
 - 当有匹配成功时候，停止匹配，按当前匹配规则处理请求
- 总的规律是，精度越高优先级越高

案例：

```
location = / {  
    #规则A  
}  
location = /login {  
    #规则B  
}  
location ^~ /static/ {  
    #规则C  
}  
location ~ \.(gif|jpg|png|js|css)$ {  
    #规则D  
}  
location ~* \.png$ {  
    #规则E  
}  
location / {  
    #规则F  
}
```

认准一手QQ3195303913微信wxywd8

访问根目录 /，比如 <http://localhost/> 将匹配规则 A

访问 <http://localhost/login> 将匹配规则 B

<http://localhost/register> 则匹配规则 F

访问 <http://localhost/static/a.html> 将匹配规则 C

访问 <http://localhost/a.gif>, <http://localhost/b.jpg> 将匹配规则 D和规则 E，但是规则 D 顺序优先，规则 E不起作用

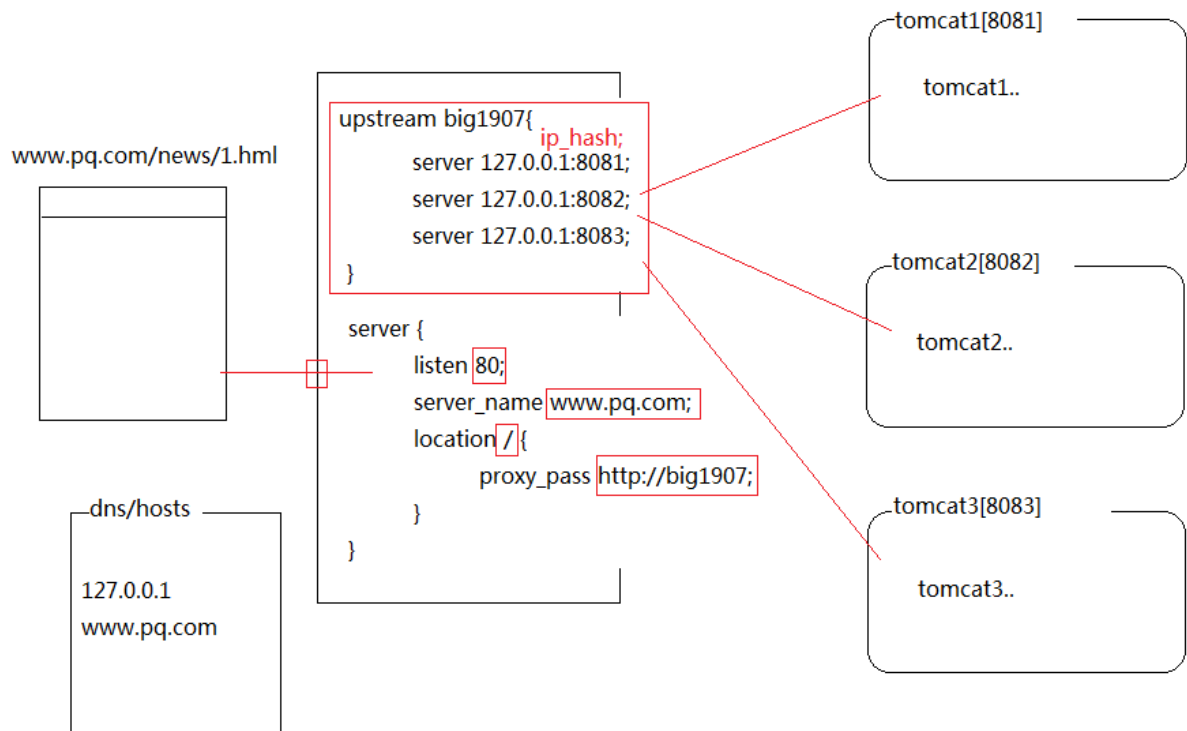
<http://localhost/static/c.png>则优先匹配到规则 C

访问 <http://localhost/a.PNG> 则匹配规则 E，而不会匹配规则 D，因为规则 E 不区分大小写

访问 <http://localhost/category/id/1111> 则最终匹配到规则 F

7.Nginx负载均衡实现

a.负载均衡原理



b.配置三台tomcat，分别监听不同端口，并启动tomcat

第22行

tomcat1	8015
tomcat2	8025
tomcat3	8035

第69行

tomcat1	8081
tomcat2	8082
tomcat3	8083

第91行

tomcat1	8019
tomcat2	8029
tomcat3	8039

c.修改nginx配置，并启动nginx

#upstream是nginx配置文件中的关键字，用来配置一组服务器地址供后续使用

```
upstream big1907 {
    server 127.0.0.1:8081;
    server 127.0.0.1:8082;
```

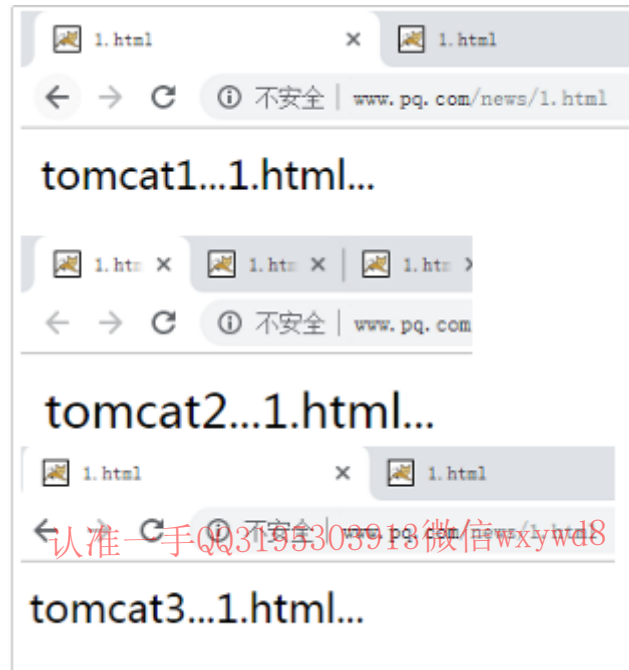
```

server 127.0.0.1:8083;
}

server {
    listen 80; #对80端口的访问
    server_name www.pq.com; #对此主机名的访问
    location / {
        #转发到上面配置的服务器组
        proxy_pass http://big1907;
    }
}

```

d.经过测试，发现确实可以通过nginx负载均衡访问到tomcat中的资源



8.负载均衡策略

nginx在分发资源到后端服务器时，如何分配请求是可以配置的，称之为nginx的负载均衡策略。

轮 询	默认不配置 就是轮询	连接请求轮流分配给后端服务器	<pre> http{ upstream sampleapp { server <<dns entry or IP Address(optional with port)>>; server <<another dns entry or IP Address(optional with port)>>; } server{ listen 80; ... location / { proxy_pass http://sampleapp; } } } </pre>
ip 哈 希	ip_hash;	<p>认准一手QQ3195303913微信wxywd8</p> <p>abs(客户端ip.hash())%服务器数量, 根据余数决定连接请求去往哪个服务器</p>	<pre> http{ upstream sampleapp { ip_hash; server <<dns entry or IP Address(optional with port)>>; server <<another dns entry or IP Address(optional with port)>>; } server{ listen 80; ... location / { proxy_pass http://sampleapp; } } } </pre>

最少连接	least_conn;	将连接请求分配给目前连接数最少的服务器	<pre> http{ upstream sampleapp { least_conn; server <<dns entry or IP Address(optional with port)>>; server <<another dns entry or IP Address(optional with port)>>; } server{ listen 80; ... location / { proxy_pass http://sampleapp; } } </pre>
基于权重	直接在地址后配置 weight=x	根据权重进行分配，权重值越大，被分配的连接越多。可以直接配置为down，则不再分配连接。	<pre> http{ upstream sampleapp { server <<dns entry or IP Address(optional with port)>> weight=2; server <<another dns entry or IP Address(optional with port)>> weight=5; server <<another dns entry or IP Address(optional with port)>> down; } server{ listen 80; ... location / { proxy_pass http://sampleapp; } } </pre>

9.Nginx的动静分离实现

a.动静分离原理

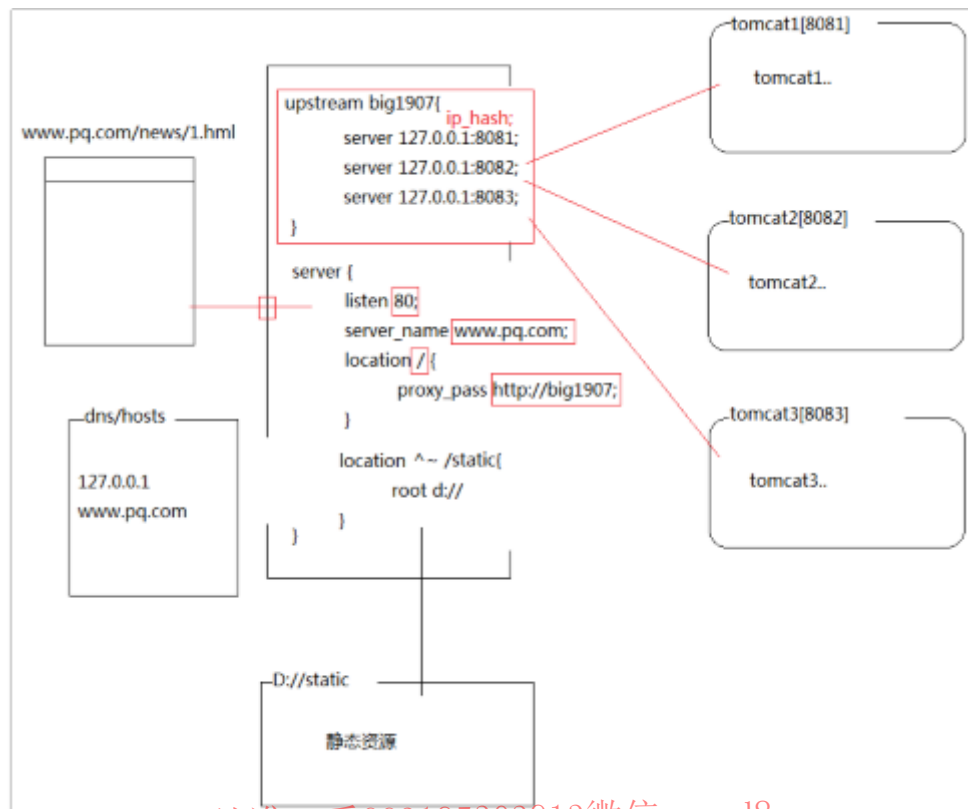
动 --> 动态资源 --> servlet jsp --> 程序

静 --> 静态资源 --> jpg mp3 mp4 html css js --> 文件

tomcat能够处理动态和静态资源，但本质上是为处理动态资源而设计的服务器，过多静态资源交由tomcat管理会降低tomcat处理动态资源的能力，得不偿失。

nginx本身无法处理动态资源，但可以处理静态资源，而且性能优良。

因此可以将静态资源和动态资源拆分，将静态资源交由nginx处理，动态资源仍由tomcat处理，从而解放了tomcat对动态资源的处理能力，整体上实现动静分离，提升了效率。



认准一手QQ3195303913微信wxywd8

b.动静分离实现

配置方式：

```
server {
    listen 80;
    server_name www.staticfile.com;
    location / {
        #root可以指向nginx服务器中的本地磁盘地址
        #静态文件就放置在这个磁盘地址中
        #之后对server中资源的访问会被转换到对本地磁盘资源的访问
        #www.staticfile.com/aaa/bbb/1.html-->d://html/aaa/bbb/1.html
        root D://html;
        #默认访问的首页配置
        index index.html;
    }
}
```

c.案例：

对<http://www.pq.com/html>路径及其子路径的访问会被转到对磁盘d://html文件夹的静态资源的访问。

```
server {  
    listen 80;  
        server_name www.pq.com;  
    location ^~/html {  
        root d://;  
        index index.html;  
    }  
}
```

认准一手QQ3195303913微信wxywd8