

Nginx 的安装及使用

1.Nginx 简介

1.1：正向代理

客户端通过代理服务器访问原始服务器。对于客户端而言，正向代理所代理的是客户端，那么客户端必须要进行一些特别的设置。代理流程如下图 1 所示：



图 1

1.2：反向代理

客户端发送请求到代理服务器，然后代理服务器将请求转发给内部网络上的其他服务器（原始服务器），并将从原始服务器上得到的结果返回给客户端，此时代理服务器就是代理的服务端，客户端无须进行特别的设置，对外就表现为一个反向代理服务器。代理流程如下图 2 所示：

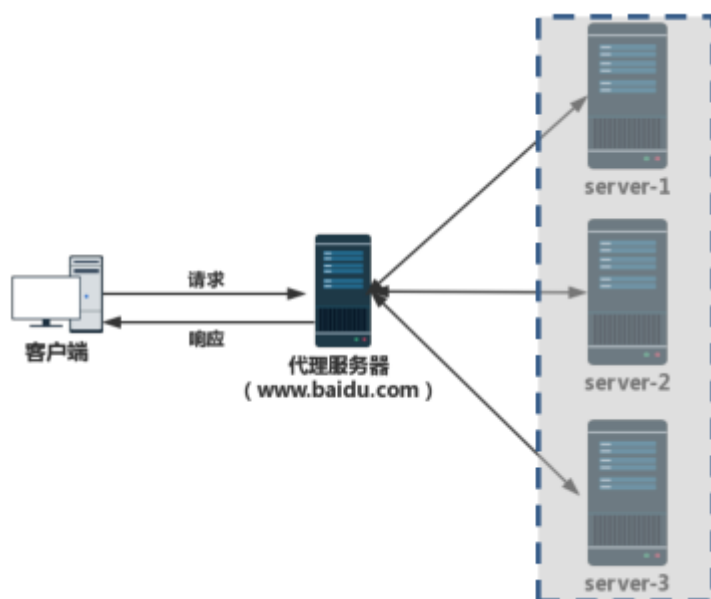


图 2

1.3: 负载均衡(Load Balance)

负载均衡是指将请求/数据【均匀】分摊到多个操作单元上执行，负载均衡的关键在于均匀，均衡的分摊压力。下图 3 是一个简单负载均衡集群。我们的爱旅行项目就是采用这种集群架构来实现负载均衡的。



图 3

1.4: Nginx

Nginx (engine x) 是一个很强大的高性能 web 服务器、反向代理服务器。并且作为反向代理服务器实现了负载均衡。目前国内使用 Nginx 网站有：百度、京东、新浪、网易、腾讯、淘宝 (Tengine) 等。

2. 安装配置

2.1: 环境准备

1. 1. Nginx 安装包

1> Widows 版

2> Linux 版

安装包：nginx-1.14.0.tar.gz

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

2. 服务器环境

1> Ubuntu14.04 64 位/CentOS6.5

2.2: 安装配置步骤

1. 安装模块依赖库

更新软件源: `sudo apt-get update`

1> pcre 库 (rewrite)

`sudo apt-get install libpcre3 libpcre3-dev`

下载地址: <http://www.pcre.org/>

2> gcc c++ 库 (c++)

`sudo apt-get install build-essential`

`sudo apt-get install libtool`

3> openssl 库 (ssl)

`sudo apt-get install openssl`

下载地址: <http://www.openssl.org/>

4> zlib 库 (gzip)

`sudo apt-get install zlib1g-dev`

下载地址: <http://www.zlib.net/>

CentOS 安装 Nginx 环境准备:

```
yum -y install gcc-c++
```

```
yum -y install pcre pcre-devel
```

```
yum -y install zlib zlib-devel
```

```
yum -y install openssl openssl-devel
```

2. 安装 Nginx

下载 nginx

`wget http://nginx.org/download/nginx-1.14.0.tar.gz`

解压:

```
tar -zxvf nginx-1.14.0.tar.gz
```

进入解压目录:

```
cd nginx-1.14.0
```

配置:

```
./configure --prefix=/usr/local/nginx --with-stream
```

编译:

```
make
```

安装:

```
make install
```

启动:

```
sudo /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf
```

注意: -c 指定配置文件的路径, 不加的话, nginx 会自动加载默认路径的配置文件, 可以通过-h 查看帮助命令。

查看进程:

```
ps -ef | grep nginx
```

3. 开放 80 端口

```
gedit /etc/sysconfig/iptables
```

```
service iptables restart #最后重启防火墙使配置生效
```

(不想开放指定端口可以直接关闭防火墙

```
service iptables stop)
```

4. 启动 Nginx

命令: `usr/local/nginx/sbin/nginx`

5. 访问 Nginx `http://主机 ip`

2.3: 常用命令

1. 负责裁剪的 Lua 脚本 (`/usr/local/Tengine/lua/ImageResizer.lua`)
2. 启动: `usr/local/nginx/sbin/nginx`
3. 停止: `usr/local/nginx/sbin/nginx -s stop`
4. 重启: `usr/local/nginx/sbin/nginx -s reload`
5. 检查配置文件(`nginx.conf`)是否合法: `usr/local/nginx/sbin/nginx -t`

2.4: Nginx 配置文件

Nginx 有一个很重要的配置文件: `conf` 目录下 `nginx.conf` 文件, Nginx 服务器的基础配置, 默认的配置都存放于这个文件内。若要实现反向代理, 负载均衡, 均需要在该文件内进行配置。我们先来看下该配置文件的结构:

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

设定 http 服务器, 利用它的反向代理功能提供负载均衡支持, 可以嵌套多个 `server`,

如果 http 服务，支持了多个虚拟主机，那么在 http 上下文里，就会出现多个 server 上下文。里面包括配置反向代理功能提供负载均衡支持，缓存，日志定义等绝大多数功能和第三方模块的配置。

4. server 块

配置虚拟主机的相关参数，一个 http 中可以有多多个 server。

5. location 块

配置请求的路由，以及各种页面的处理情况，location 根据其后面的正则进行匹配，对请求 URL 过滤。

3.项目实现负载均衡

3.1：反向代理

Nginx 只处理静态不处理动态内容，动态内容交给后台 Tomcat 处理。关键代码片段如下：

```
upstream itripbiz_server
{
    server 127.0.0.1:8080;
}

server {

    listen 80;

    server_name itrip.project.bdqn.cn;

    root /data/itrip/itripfront; #前端静态工程

    location / {

        proxy_set_header Host $host;

        proxy_set_header X-Real-IP $remote_addr;
```

```
    proxy_pass http://itripbiz_server; #反向代理地址
}
```

3.2: 实现负载均衡

1> 轮询

```
upstream itripbiz_server
{
    server 127.0.0.1:8080;

    server 127.0.0.1:8082;
}

server {

    listen 80;

    server_name itrip.project.bdqn.cn;

    root /data/itrip/itripfront; #前端静态工程

    location / {

        proxy_set_header Host $host;

        proxy_set_header X-Real-IP $remote_addr;

        proxy_pass http://itripbiz_server; #反向代理地址
    }
}
```

2> 热备 (backup)

```
upstream itripbiz_server
{

    server 127.0.0.1:8080;

    server 127.0.0.1:8082 backup;
}
```

```
}
```

3> 权重 (weight)

```
upstream itripbiz_server
```

```
{
```

```
    server 127.0.0.1:8080;
```

```
    server 127.0.0.1:8082 weight=2;
```

```
}
```

4> IP 地址 hash (ip_hash)

可解决 Tomcat 之间 session 共享问题。并且基于权重的负载均衡和基于 IP 地址哈希的负载均衡可以组合在一起使用。

```
upstream itripbiz_server
```

```
{
```

```
    ip_hash;
```

```
    server 127.0.0.1:8080;
```

```
    server 127.0.0.1:8082;
```

```
}
```

5> fair (第三方)

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 itripbiz_server {
```

```
    fair;
```

```
    server localhost:8080;
```

```
    server localhost:8081;
```

```
}
```

6> url_hash (第三方)

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。在 upstream 中加入 hash 语句，server 语句中不能写入 weight 等其他的参数，hash_method 是使用的 hash 算法


```
upstream itripbiz_server {  
    hash $request_uri;  
    hash_method crc32;  
    server localhost:8080;  
    server localhost:8081;  
}
```

注意: fair 和 url_hash 需要安装第三方模块才能使用。

3.3: 配置负载均衡



3.4: 配置动静分离

在 `nginx.conf` 配置文件中配置好静态内容处理之后, 在 `nginx` 的 `html` 文件夹下创建一个与项目同名的文件夹, 设置其权限 `777`

```
[root@shuaige demo]# cd ..
[root@shuaige html]# chmod 777 demo
[root@shuaige html]#
```

4.常见错误

4.1: Nginx 使用非默认文件启动报错

```
nginx: [emerg] open() "/usr/local/nginx/mime.types" failed (2: No such file or
directory) in /usr/local/nginx/nginx.conf:18
```

Reson: 我们将自定义的 nginx 配置文件放在 conf 目录之外导致其启动需要引用的 mime.types

文件找不到 无法加载 所以报错

4.2: Nginx 在请求时报 400 错误

2.1: bad gateway 这个服务器不行 服务器不稳定


2.2: 转发时找不到目标服务器 需要在 http 中配置

```
server {
    listen      80;
    server_name localhost;

    charset UTF-8;
    client_header_buffer_size 2k;
    large_client_header_buffers 4 4k;
    client_max_body_size 8m;

    #access_log logs/host.access.log main;
    proxy_set_header Host $host;
    location ~ \.(jpg|gif|png|swf|html|css|js)$ {
        root html;
        index index.html index.htm;
    }

    location ~ \.(jsp|do)$ {
        proxy_pass http://local_tomcat;
    }
}
```

 不是必须的