**高性能反向代理服务器——Nginx第一节课程**

授课老师：\*\*\*Mic老师

授课时间：2017年12月6号 20：30分

**学习目标**

对Nginx有一个全面的了解，能够独立完成Nginx的部署和配置。

**本次课课程大纲**

1. 关于反向代理和正向代理
2. 常用Web服务器介绍
3. Nginx、Apache、Tomcat之间的联系
4. Nginx安装部署
5. Nginx核心配置分析
6. 虚拟主机的配置
7. Location匹配规则剖析
8. 关于Nginx日志配置和及切割处理

**课程内容笔记**

# 关于反向代理和正向代理

## 正向代理

正向代理的对象是客户端

## 反向代理

反向代理代理的是服务端

# 常用Web服务器介绍

apache、Nginx、tomcat、weblogic、iis、jboss、websphere、jetty、netty、lighttpd、glassfish、resin

## Apache

Apache仍然是时长占用量最高的web服务器，据最新数据统计，市场占有率目前是50%左右。主要优势在于一个是比较早出现的一个Http静态资源服务器，同时又是开源的。所以在技术上的支持以及市面上的各种解决方案都比较成熟。Apache支持的模块非常丰富。

## Lighttpd

Lighttpd其设计目标是提供一个专门针对高性能网站、安全、快速、兼容性好并且灵活的web server环境。特点是：内存开销低、CPU占用率低、性能好、模块丰富。Lighttpd跟Nginx一样，是一款轻量级的Web服务器。跟Nginx的定位类似

## Tomcat

Tomcat大家都比较熟悉，是一个开源的JSP Servlet容器。

## Nginx

Nginx是俄罗斯人编写的一款高性能的HTTP和反向代理服务器，在高连接并发的情况下，它能够支持高达50000个并发连接数的响应，但是内存、CPU等系统资源消耗却很低，运行很稳定。目前Nginx在国内很多大型企业都有应用，据最新统计，Nginx的市场占有率已经到33%左右了。而Apache的市场占有率虽然仍然是最高的，但是是呈下降趋势。而Nginx的势头很明显。选择Nginx的理由也很简单：第一，它可以支持5W高并发连接；第二，内存消耗少；第三，成本低，如果采用F5、NetScaler等硬件负载均衡设备的话，需要大几十万。而Nginx是开源的，可以免费使用并且能用于商业用途

## Nginx服务器、Apache Http Server、Tomcat之间的关系

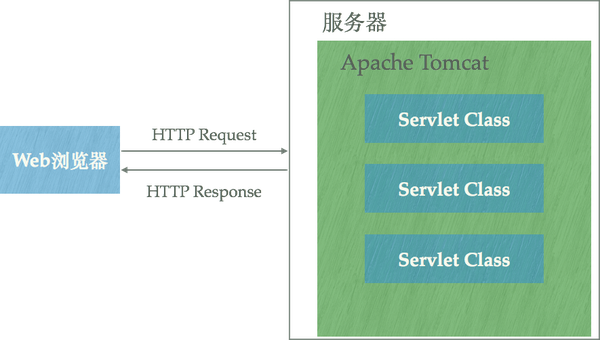
HTTP服务器本质上也是一种应用程序——它通常运行在服务器之上，绑定服务器的IP地址并监听某一个tcp端口来接收并处理HTTP请求，这样客户端（一般来说是IE, Firefox，Chrome这样的浏览器）就能够通过HTTP协议来获取服务器上的网页（HTML格式）、文档（PDF格式）、音频（MP4格式）、视频（MOV格式）等等资源



Apache Tomcat是Apache基金会下的一款开源项目，Tomcat能够动态的生成资源并返回到客户端。

Apache HTTP server 和Nginx都能够将某一个文本文件的内容通过HTTP协议返回到客户端，但是这些文本文件的内容是固定的，也就是说什么情况下访问该文本的内容都是完全一样的，这样的资源我们称之为静态资源。动态资源则相反，不同时间、不同客户端所得到的内容是不同的 ； （虽然apache和nginx本身不支持动态页面，但是他们可以集成模块来支持，比如PHP、Python）

如果想要使用java程序来动态生成资源内容，使用apache server和nginx这一类的http服务器是基本做不到。而Java Servlet技术以及衍生出来的（jsp）Java Server Pages技术可以让Java程序也具有处理HTTP请求并且返回内容的能力，而Apache Tomcat正是支持运行Servlet/JSP应用程序的容器



Tomcat运行在JVM之上，它和HTTP服务器一样，绑定IP地址并监听TCP端口，同时还包含以下职责：

* 管理Servlet程序的生命周期
* 将URL映射到指定的Servlet进行处理
* 与Servlet程序合作处理HTTP请求——根据HTTP请求生成HttpServletResponse对象并传递给Servlet进行处理，将Servlet中的HttpServletResponse对象生成的内容返回给浏览器

虽然Tomcat也可以认为是HTTP服务器，但通常它仍然会和Nginx配合在一起使用：

* 动静态资源分离——运用Nginx的反向代理功能分发请求：所有动态资源的请求交给Tomcat，而静态资源的请求（例如图片、视频、CSS、JavaScript文件等）则直接由Nginx返回到浏览器，这样能大大减轻Tomcat的压力。
* 负载均衡，当业务压力增大时，可能一个Tomcat的实例不足以处理，那么这时可以启动多个Tomcat实例进行水平扩展，而Nginx的负载均衡功能可以把请求通过算法分发到各个不同的实例进行处理

## Apache HTTP Server和Nginx的关系

Apache Http Server是使用比较广泛也是资格最老的web服务器，是Apache基金会下第一个开源的WEB服务器。在Nginx出现之前，大部分企业使用的都是Apache。

在互联网发展初期，流量不是特别大的时候，使用Apache完全满足需求。但是随着互联网的飞速发展，网站的流量以指数及增长，这个时候除了提升硬件性能以外，Apache Http server也开始遇到瓶颈了，于是这个时候Nginx的出现，就是为了解决大型网站高并发设计的，所以对于高并发来说，Nginx有先天的优势。因此Nginx也在慢慢取代Apache Http server。 而Nginx另一个强大的功能就是反向代理，现在大型网站分工详细，哪些服务器处理数据流，哪些处理静态文件，这些谁指挥，一般都是用nginx反向代理到内网服务器，这样就起到了负载均衡分流的作用。再次nginx高度模块化的设计，编写模块相对简单。

# Nginx的安装

## nginx安装

1. tar -zxvf 安装包
2. ./configure --prefix=/mic/data/program/nginx 默认安装到/usr/local/nginx
3. make & make install

## 启动停止

./nginx -c /mic/data/program/nginx/conf/nginx.conf 启动nginx -c表示指定nginx.conf的文件。如果不指定，默认为NGINX\_HOME/conf/nginx.conf

发送信号的方式

kill -QUIT 进程号

kil -TERM 进程号

停止nginx

./nginx -s stop 停止

./nginx -s quit 退出

./nginx -s reload 重新加载nginx.conf

安装过程中可能会出现的问题

缺少pcre的依赖

缺少openssl的依赖

yum install pcre-devel

yum install openssl-devel

yum install zlib-devel

# Nginx核心配置分析

nginx的核心配置文件,主要包括三个段

Main、 Event 、 Http

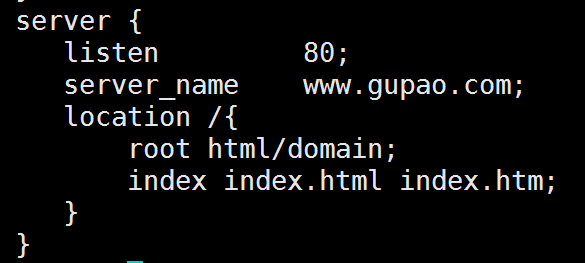
# 虚拟主机配置

**基于域名的虚拟主机**

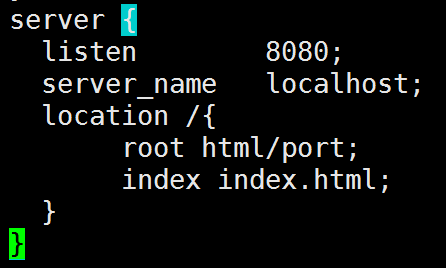
**修改windows/system32/drivers/etc/hosts**



**修改nginx.conf文件，在http段中增加如下内容，说明会去${nginx}/html/domain目录下去找index.html,或者index.htm**



**基于端口的虚拟主机**



基于ip的虚拟主机

# Nginx的日志配置

通过access\_log进行日志记录

nginx中有两条是配置日志的：一条是log\_format 来设置日志格式 ； 另外一条是access\_log

#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 格式

#error\_log logs/error.log notice;

logo声明 路径及文件名 日志标识

nginx日志切割

crontab

1.mv access.log access.log.20171206

2.kill -USR1 Nginx 主进程号 让nginx重新生成一个日志文件access.log

# location的语法和匹配规则

location [~|=|^~|~\*] /uri {

}

## location的匹配规则

1.精准匹配

location=/uri{}

优先级最高的匹配规则

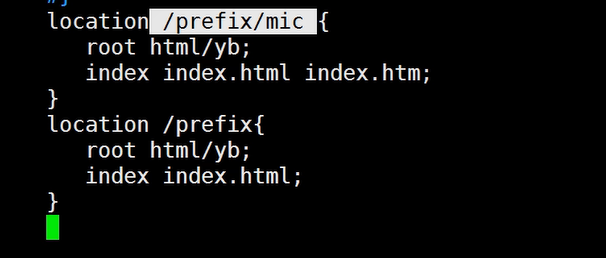
2.一般匹配

location /uri{

}

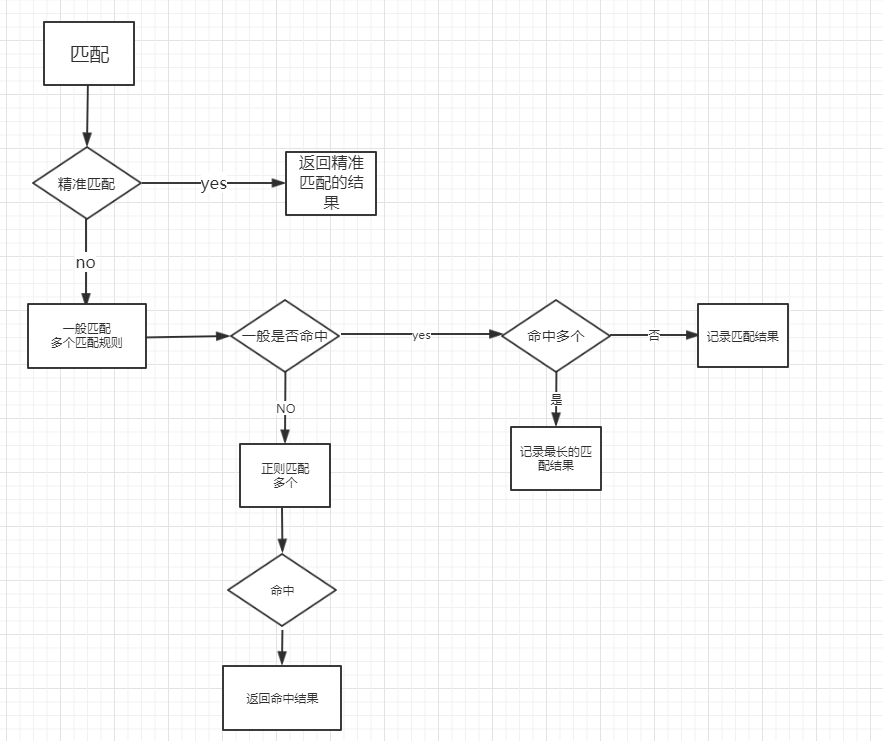
1.普通匹配的优先级要高于正则匹配

如果存在多个相同的前缀的一般匹配，那么最终会按照最大长度来做匹配

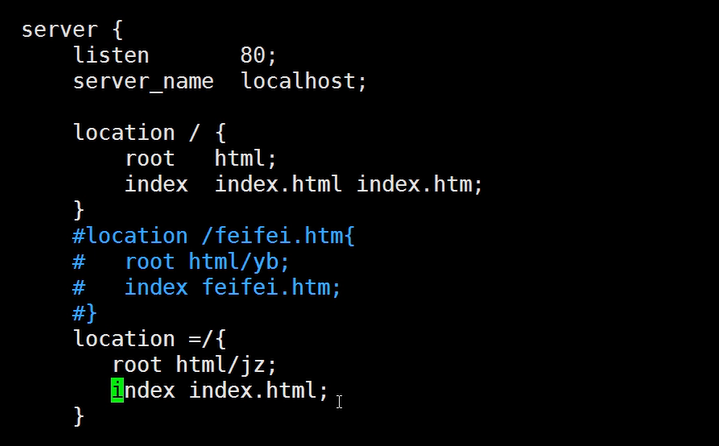


prefix/mic比/prefix更精准，会先匹配prefix/mic

2.正则匹配



举例：



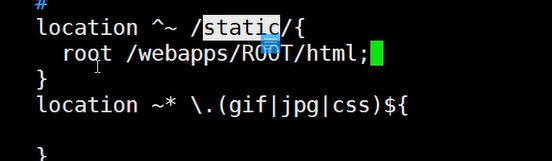
输入localhost:

会按精准匹配先匹配到location=/ 他正面要去找index.html

那就会去访问localhost://index.html这时候他就不属于精准匹配了，所以会去找到location /里的index.html

所以最终会访问的是location /下面的index.html

实例应用：



**Q&A**