# 一、Nginx简介

## 1、什么是 Nginx

Nginx 是俄罗斯人编写的十分轻量级的 HTTP 服务器,Nginx，它的发音为“engine X”，是一个高性能的HTTP和反向代理服务器，同时也是一个 IMAP/POP3/SMTP 代理服务器。

Nginx 因为它的稳定性、丰富的模块库、灵活的配置和低系统资源的消耗而闻名．业界一致认为它是 Apache2.2＋mod\_proxy\_balancer 的轻量级代替者，不仅是因为响应静态页面的速度非常快，而且它的模块数量达到 Apache 的近 2/3。对 proxy 和 rewrite 模块的支持很彻底，还支持 mod\_fcgi、ssl、vhosts ，适合用来做 mongrel clusters 的前端 HTTP 响应。

目前Nginx在国内很多大型企业都有应用，且普及率呈逐年上升趋势。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。

## 2、Nginx在架构中发挥的作用

* **网关**

---面向客户的总入口。

* **虚拟主机**

---一台机器为不同的域名/ip/端口提供服务

* **路由**

---使用反向代理，整合后续服务为一个完整业务

* **静态服务器**

---mvvm模式中，用来发布前端html/css/js/img

* **负载集群**

---使用upstream，负载多个tomcat

# 二、Nginx架构设计

## 1、Nginx的模块化设计

高度模块化的设计是 Nginx 的架构基础。Nginx 服务器被分解为多个模块，每个模块就是一个功能模块，只负责自身的功能，模块之间严格遵循“高内聚，低耦合”的原则。



* **核心模块**

核心模块是 Nginx 服务器正常运行必不可少的模块，提供错误日志记录、配置文件解析、事件驱动机制、进程管理等核心功能。

* **标准 HTTP 模块**

标准 HTTP 模块提供 HTTP 协议解析相关的功能，如：端口配置、网页编码设置、HTTP 响应头设置等。

* **可选 HTTP 模块**

可选 HTTP 模块主要用于扩展标准的 HTTP 功能，让 Nginx 能处理一些特殊的服务，如：Flash 多媒体传输、解析 GeoIP 请求、SSL 支持等。

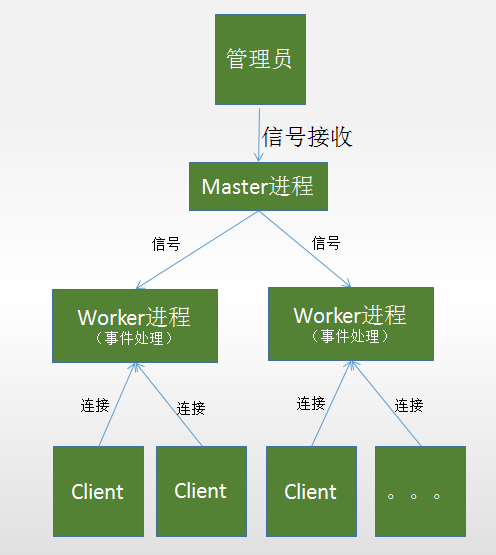
* **邮件服务模块**

邮件服务模块主要用于支持 Nginx 的邮件服务，包括对 POP3 协议、IMAP 协议和 SMTP 协议的支持。

* **第三方模块**

第三方模块是为了扩展 Nginx 服务器应用，完成开发者自定义功能，如：Json 支持、Lua 支持等。

## 2、Nginx多进程模型



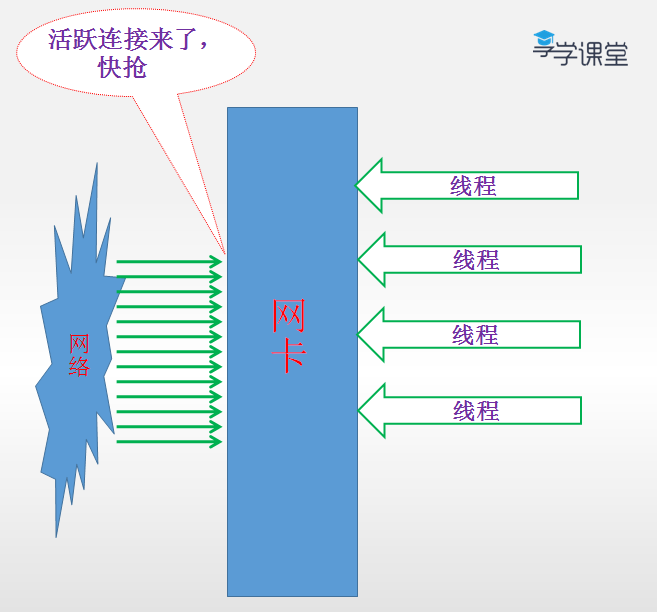
1、服务器每当收到一个客户端时。就有服务器主进程（master process）生成一个子进程（worker  process）出来和客户端建立连接进行交互，直到连接断开，该子进程结束。

2、使用进程的好处是各个进程之间相互独立，不需要加锁，减少了使用锁对性能造成影响，同时降低编程的复杂度，降低开发成本。

其次，采用独立的进程，可以让进程互相之间不会影响，如果一个进程发生异常退出时，其它进程正常工作，master 进程则很快启动新的 worker 进程，确保服务不中断，将风险降到最低。

缺点是操作系统生成一个子进程需要进行内存复制等操作，在资源和时间上会产生一定的开销；当有大量请求时，会导致系统性能下降。

## 3、Nginx的epoll模式



select和poll的处理模式如上图：

--在某一时刻，进程收集所有的连接，其实这100万连接中大部分是没有事件发生的。因此，如果每次收集事件时，都把这100万连接的套接字传给操作系统（这首先就是用户态内存到内核内存的大量复制），而由操作系统内核寻找这些链接上没有处理的事件，将会是巨大的浪费。

而epoll改进了收集连接的动作，提高效率。

**epoll的优点：**

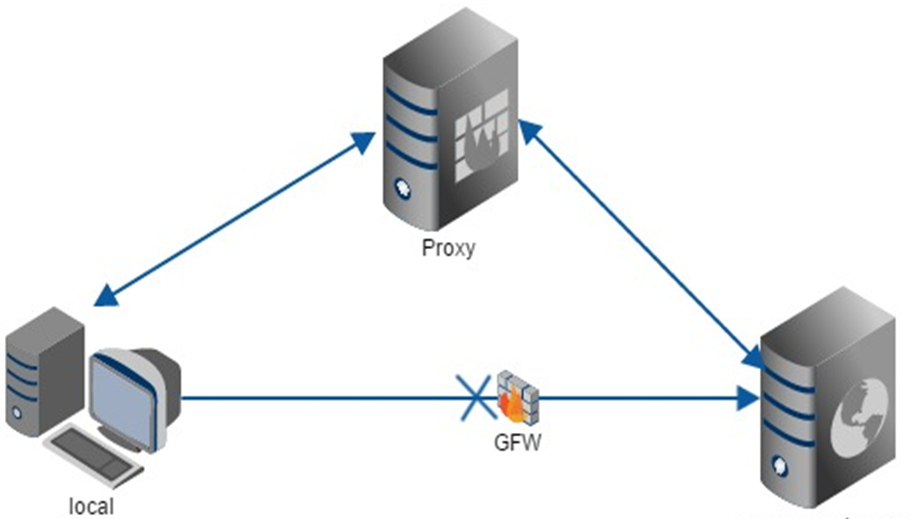
1.支持一个进程打开大数目的socket描述符(FD)

2.IO效率不随FD数目增加而线性下降

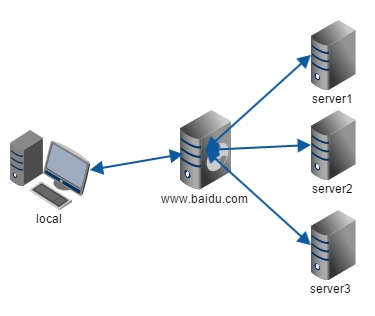
3.使用mmap加速内核与用户空间的消息传递

## 4、正向代理与反向代理

1、代理：意思是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。



2、反向代理，服务端推出的一个代理招牌。



# 四、nginx安装配置：

## 1、源码编译方式：

安装make：yum -y install autoconf automake make

安装g++: yum -y install gcc gcc-c++

#一般系统中已经装了了make和g++，无须再装

yum -y install pcre pcre-devel

yum -y install zlib zlib-devel

yum install -y openssl openssl-devel

#安装nginx依赖的库

wget  <http://nginx.org/download/nginx-1.15.8.tar.gz>

tar -zxvf nginx-1.15.8.tar.gz

cd nginx-1.15.8

./configure   --prefix=/usr/local/nginx --with-http\_stub\_status\_module --with-http\_ssl\_module

#配置

#--prefix指定安装目录

#--with-http\_ssl\_module安装https模块

#creating objs/Makefile 代表编译成功

make && make install

#make编译

#make install安装

## 2、yum方式：

yum install yum-utils

yum-config-manager --add-repo https://openresty.org/package/centos/openresty.repo

yum install openresty

## 3、Nginx目录结构：

* Conf  配置文件
* Html   网页文件
* Logs   日志文件
* Sbin    二进制程序

## 4、Nginx常用命令

**启停命令：**

./nginx -c nginx.conf的文件。如果不指定，默认为NGINX\_HOME/conf/nginx.conf

./nginx -s stop  停止

./nginx -s quit退出

./nginx -s reload 重新加载nginx.conf

# 五、nginx模型概念：

**Nginx会按需同时运行多个进程：**

一个主进程(master)和几个工作进程(worker)，配置了缓存时还会有缓存加载器进程(cache loader)和缓存管理器进程(cache manager)等。

所有进程均是仅含有一个线程，并主要通过“共享内存”的机制实现进程间通信。

主进程以root用户身份运行，而worker、cache loader和cache manager均应以非特权用户身份（user配置项）运行。

**主进程主要完成如下工作：**

1. 读取并验正配置信息；

2. 创建、绑定及关闭套接字；

3. 启动、终止及维护worker进程的个数；

4. 无须中止服务而重新配置工作特性；

5. 重新打开日志文件；

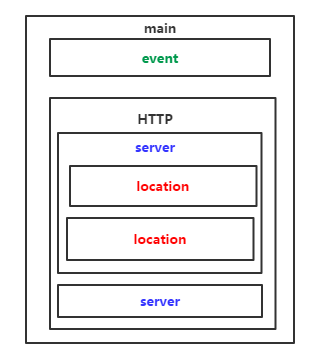
**worker进程主要完成的任务包括：**

1. 接收、传入并处理来自客户端的连接；

2. 提供反向代理及过滤功能；

3. nginx任何能完成的其它任务；

# 六、nginx.conf配置文件结构



#user nobody; #主模块命令， 指定Nginx的worker进程运行用户以及用户组，默认由nobody账号运行。

worker\_processes 1;#指定Nginx要开启的进程数。

worker\_rlimit\_nofile 100000; #worker进程的最大打开文件数限制

#error\_log logs/error.log;

#error\_log logs/error.log notice;

#error\_log logs/error.log info;

#pid logs/nginx.pid;

events {

use epoll;

worker\_connections 1024;

}

/\*

**以上这块配置代码是对nginx全局属性的配置。**

  user :主模块命令， 指定Nginx的worker进程运行用户以及用户组，默认由nobody账号运行。

  worker\_processes: 指定Nginx要开启的进程数。

  error log:用来定义全局错设日志文件的路径和日志名称。

日志输出级别有debug，info，notice，warn，error，crit 可供选择，其中debug输出日志最为详细，面crit（严重）输出日志最少。默认是error

  pid: 用来指定进程id的存储文件位置。

  event：设定nginx的工作模式及连接数上限，

其中参数use用来指定nginx的工作模式（这里是epoll，epoll是多路复用IO(I/O Multiplexing)中的一种方式）,

nginx支持的工作模式有select ,poll,kqueue,epoll,rtsig,/dev/poll。

其中select和poll都是标准的工作模式，kqueue和epoll是高效的工作模式，对于linux系统，epoll是首选。

    worker\_connection是设置nginx每个进程最大的连接数，默认是1024，所以nginx最大的连接数max\_client=worker\_processes \* worker\_connections。

进程最大连接数受到系统最大打开文件数的限制，需要设置ulimit。

\*/

**#下面部分是nginx对http服务器相关属性的设置**

http {

include mime.types; 主模块命令，对配置文件所包含文件的设定，减少主配置文件的复杂度，相当于把部分设置放在别的地方，然后在包含进来，保持主配置文件的简洁

default\_type application/octet-stream; 默认文件类型，当文件类型未定义时候就使用这类设置的。

#log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" ' 指定nginx日志的格式

# '$status $body\_bytes\_sent "$http\_referer" '

# '"$http\_user\_agent" "$http\_x\_forwarded\_for"';

#access\_log logs/access.log main;

sendfile on; 开启高效文件传输模式（zero copy 方式），避免内核缓冲区数据和用户缓冲区数据之间的拷贝。

#tcp\_nopush on; 开启TCP\_NOPUSH套接字（sendfile开启时有用）

#keepalive\_timeout 0; 客户端连接超时时间

keepalive\_timeout 65;

#gzip on; 设置是否开启gzip模块

**#下面是server段虚拟主机的配置**

server {

listen 80; 虚拟主机的服务端口

server\_name localhost; 用来指定ip或者域名，多个域名用逗号分开

#charset koi8-r;

location / {

#地址匹配设置，支持正则匹配，也支持条件匹配，这里是默认请求地址，用户可以location命令对nginx进行动态和静态网页过滤处理

root html; 虚拟主机的网页根目录

index index.html index.htm; 默认访问首页文件

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

}

# 七、Nginx日志

Nginx日志对于统计、系统服务排错很有用。

Nginx日志主要分为两种：access\_log(访问日志)和error\_log(错误日志)。

通过访问日志我们可以得到用户的IP地址、浏览器的信息，请求的处理时间等信息。

错误日志记录了访问出错的信息，可以帮助我们定位错误的原因。

因此，将日志好好利用，可以得到很多有价值的信息。

查看日志命令：

1. tail -f /usr/local/nginx/logs/access.log

## 1、设置access\_log

访问日志主要记录客户端的请求。客户端向Nginx服务器发起的每一次请求都记录在这里。客户端IP，浏览器信息，referer，请求处理时间，请求URL等都可以在访问日志中得到。当然具体要记录哪些信息，你可以通过log\_format指令定义。

### 语法

1. access\_log path [format [buffer=size] [gzip[=level]] [flush=time] [if=condition]]; *# 设置访问日志*
2. access\_log off; *# 关闭访问日志*

* path 指定日志的存放位置。
* format 指定日志的格式。默认使用预定义的combined。
* buffer 用来指定日志写入时的缓存大小。默认是64k。
* gzip 日志写入前先进行压缩。压缩率可以指定，从1到9数值越大压缩比越高，同时压缩的速度也越慢。默认是1。
* flush 设置缓存的有效时间。如果超过flush指定的时间，缓存中的内容将被清空。
* if 条件判断。如果指定的条件计算为0或空字符串，那么该请求不会写入日志。
* 另外，还有一个特殊的值off。如果指定了该值，当前作用域下的所有的请求日志都被关闭。

### 示例

1. http {
2. include mime.types;
3. default\_type application/octet-stream;
4. log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '
5. '$status $body\_bytes\_sent "$http\_referer" '
6. '"$http\_user\_agent" "$http\_x\_forwarded\_for"';
7. ##日志格式使用默认的combined，指定日志的缓存大小为32k，日志写入前启用gzip进行压缩，压缩比使用默认值1，缓存数据有效时间为1分钟。
8. access\_log /var/logs/nginx-access.log buffer=32k gzip flush=1m;
9. ...
10. }

### 作用域

access\_log指令的作用域分别有http，server，location。

## 2、log\_format自定义格式

**默认的日志格式**

1. log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '
2. '$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

各参数明细表：

|  |  |
| --- | --- |
| $remote\_addr | 客户端的ip地址(代理服务器，显示代理服务ip) |
| $remote\_user | 用于记录远程客户端的用户名称（一般为“-”） |
| $time\_local | 用于记录访问时间和时区 |
| $request | 用于记录请求的url以及请求方法 |
| $status | 响应状态码，例如：200成功、404页面找不到等。 |
| $body\_bytes\_sent | 给客户端发送的文件主体内容字节数 |
| $http\_user\_agent | 用户所使用的代理（一般为浏览器） |
| $http\_x\_forwarded\_for | 可以记录客户端IP，通过代理服务器来记录客户端的ip地址 |
| $http\_referer | 可以记录用户是从哪个链接访问过来的 |

## 3、设置error\_log

错误日志在Nginx中是通过error\_log指令实现的。该指令记录服务器和请求处理过程中的错误信息。

错误日志不支持自定义。

### 语法

1. error\_log path [level];

* path参数指定日志的写入位置。
* level参数指定日志的级别（不写为全部）。level可以是debug, info, notice, warn, error, crit, alert,emerg中的任意值（等级从低到高排列）。

只有日志的错误级别等于或高于level指定的值才会写入错误日志中。默认值是error。

### 示例

error\_log logs/error.log;

error\_log logs/error\_notice.log notice;

error\_log logs/error\_info.log info; ##可以将不同的错误类型分开存储

## 4、日志配置和及切割

/etc/init.d/rsyslog start #系统日志，如不开启，看不到定时任务日志

/etc/rc.d/init.d/crond start #定时任务开启

**编写sh：**

#!/bin/bash

#设置日志文件存放目录

LOG\_HOME="/usr/local/nginx/logs/"

#备分文件名称

LOG\_PATH\_BAK="$(date -d yesterday +%Y%m%d%H%M)"

#重命名日志文件

mv ${LOG\_HOME}/access.log ${LOG\_HOME}/access.${LOG\_PATH\_BAK}.log

mv ${LOG\_HOME}/error.log ${LOG\_HOME}/error.${LOG\_PATH\_BAK}.log

#向nginx主进程发信号重新打开日志

kill -USR1 `cat ${LOG\_HOME}/nginx.pid`

**配置cron：**

\*/1 \* \* \* \* /usr/local/nginx/sbin/logcut.sh