# [Nginx 之一：编译安装nginx 1.8.1 及配置](http://www.cnblogs.com/zhang-shijie/p/5294162.html)

一：基介绍

  官网地址www.nginx.org，nginx是由1994年毕业于俄罗斯国立莫斯科鲍曼科技大学的同学为俄罗斯rambler.ru公司开发的，开发工作最早从2002年开始，第一次公开发布时间是2004年10月4日，版本号是0.1.0

　Nginx是单进程单线程模型，即启动的工作进程只有一个进程响应客户端请求，不像apache可以在一个进程内启动多个线程响应可请求，因此在内存占用上比apache小的很多。Nginx维持一万个非活动会话只要2.5M内存。Nginx和Mysql是CPU密集型的，就是对CPU的占用比较大，默认session在本地文件保存，支持将session保存在memcache，但是memcache默认支持最大1M的课hash对象。

  nginx的版本分为开发版、稳定版和过期版，nginx以功能丰富著称，它即可以作为http服务器，也可以作为反向代理服务器或者邮件服务器，能够快速的响应静态网页的请求，支持FastCGI/SSL/Virtual Host/URL Rwrite/Gzip/HTTP Basic Auth等功能，并且支持第三方的功能扩展。

  nginx安装可以使用yum或源码安装，推荐使用源码，一是yum的版本比较旧，二是使用源码可以自定义功能，方便业务的上的使用，源码安装需要提前准备标准的编译器，GCC的全称是（GNU Compiler collection），其有GNU开发，并以GPL即LGPL许可，是自由的类UNIX即苹果电脑Mac OS X操作系统的标准编译器，因为GCC原本只能处理C语言，所以原名为GNU C语言编译器，后来得到快速发展，可以处理C++,Fortran，pascal，objective-C，java以及Ada等其他语言，此外还需要Automake工具，以完成自动创建Makefile的工作，Nginx的一些模块需要依赖第三方库，比如pcre（支持rewrite），zlib（支持gzip模块）和openssl（支持ssl模块）

二：安装

1、环境准备：先安装准备环境

yum install gcc gcc-c++ automake pcre pcre-devel zlip zlib-devel openssl openssl-devel

[IMG_256](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

　　gcc为GNU Compiler Collection的缩写，可以编译C和C++源代码等，它是GNU开发的C和C++以及其他很多种语言 的编译器（最早的时候只能编译C，后来很快进化成一个编译多种语言的集合，如Fortran、Pascal、Objective-C、Java、Ada、 Go等。）

　　gcc 在编译C++源代码的阶段，只能编译 C++ 源文件，而不能自动和 C++ 程序使用的库链接（编译过程分为编译、链接两个阶段，注意不要和可执行文件这个概念搞混，相对可执行文件来说有三个重要的概念：编译（compile）、链接（link）、加载（load）。源程序文件被编译成目标文件，多个目标文件连同库被链接成一个最终的可执行文件，可执行文件被加载到内存中运行）。因此，通常使用 g++ 命令来完成 C++ 程序的编译和连接，该程序会自动调用 gcc 实现编译。

　　gcc-c++也能编译C源代码，只不过把会把它当成C++源代码，后缀为.c的，gcc把它当作是C程序，而g++当作是c++程序；后缀为.cpp的，两者都会认为是c++程序，注意，虽然c++是c的超集，但是两者对语法的要求是有区别的。

　　automake是一个从Makefile.am文件自动生成Makefile.in的工具。为了生成Makefile.in，automake还需用到perl，由于automake创建的发布完全遵循GNU标准，所以在创建中不需要perl。libtool是一款方便生成各种程序库的工具。

　　pcre pcre-devel：在Nginx编译需要 PCRE(Perl Compatible Regular Expression)，因为Nginx 的Rewrite模块和HTTP 核心模块会使用到PCRE正则表达式语法。

　　zlip zlib-devel：nginx启用压缩功能的时候，需要此模块的支持。

　　openssl openssl-devel：开启SSL的时候需要此模块的支持。

[IMG_257](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

 2、下载nginx 安装包：  官网地址：http://nginx.org/

截止得到当前，最新的版本为1.8.1，在linux使用wget下载:

[IMG_258](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

[root@Server1 ~]# wget http://nginx.org/download/nginx-1.8.1.tar.gz

--2016-04-23 10:22:55-- http://nginx.org/download/nginx-1.8.1.tar.gz

Resolving nginx.org (nginx.org)... 206.251.255.63, 95.211.80.227, 2001:1af8:4060:a004:21::e3, ...

Connecting to nginx.org (nginx.org)|206.251.255.63|:80... connected.

HTTP request sent, awaiting response... 200 OK

Length: 833473 (814K) [application/octet-stream]

Saving to: ‘nginx-1.8.1.tar.gz.1’

100%[==============================================================================================================>] 833,473 251KB/s in 3.2s

2016-04-23 10:23:00 (251 KB/s) - ‘nginx-1.8.1.tar.gz.1’ saved [833473/833473]

[IMG_259](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

3、解压安装包:

[root@Server1 ~]# tar xvf nginx-1.8.1.tar.gz

[root@Server1 ~]# cd nginx-1.8.1

[root@Server1 nginx-1.8.1]$ ls

auto CHANGES CHANGES.ru conf configure contrib html LICENSE man README src

4、编译nginx：make

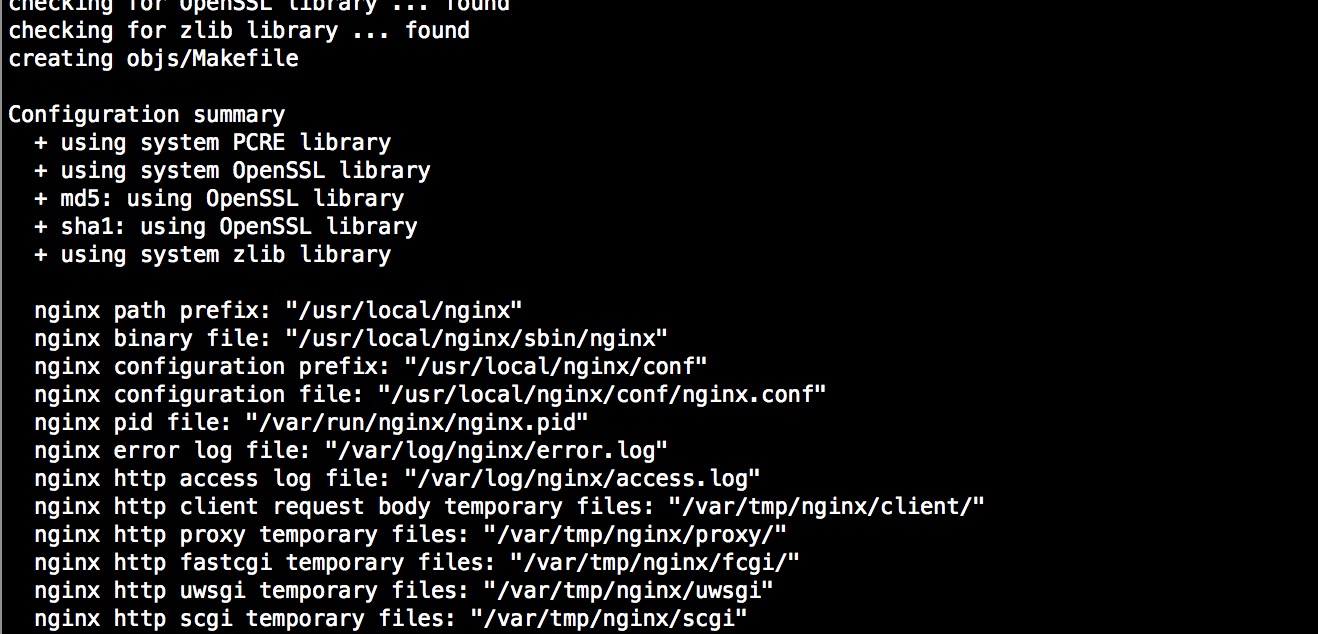
编译是为了检查系统环境是否符合编译安装的要求，比如是否有gcc编译工具，是否支持编译参数当中的模块，并根据开启的参数等生成Makefile文件为下一步做准备：

[IMG_260](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

[root@Server1 nginx-1.8.1]# ./configure --prefix=/usr/local/nginx --sbin-path=/usr/local/nginx/sbin/nginx --conf-path=/usr/local/nginx/conf/nginx.conf --error-log-path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --pid-path=/var/run/nginx/nginx.pid --lock-path=/var/lock/nginx.lock --user=nginx --group=nginx --with-http\_ssl\_module --with-http\_stub\_status\_module --with-http\_gzip\_static\_module --http-client-body-temp-path=/var/tmp/nginx/client/ --http-proxy-temp-path=/var/tmp/nginx/proxy/ --http-fastcgi-temp-path=/var/tmp/nginx/fcgi/ --http-uwsgi-temp-path=/var/tmp/nginx/uwsgi --http-scgi-temp-path=/var/tmp/nginx/scgi --with-pcre

[IMG_261](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

结果如下:



5、生成脚本及配置文件：make

编译步骤，根据Makefile文件生成相应的模块



6、安装：make install

创建目录，并将生成的模块和文件复制到相应的目录： 

备注：nginx完成安装以后，有四个主要的目录：

[IMG_265](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

conf：保存nginx所有的配置文件，其中nginx.conf是nginx服务器的最核心最主要的配置文件，其他的.conf则是用来配置nginx相关的功能的，例如fastcgi功能使用的是fastcgi.conf和fastcgi\_params两个文件，配置文件一般都有个样板配置文件，是文件名.default结尾，使用的使用将其复制为并将default去掉即可。

html目录中保存了nginx服务器的web文件，但是可以更改为其他目录保存web文件,另外还有一个50x的web文件是默认的错误页面提示页面。

logs：用来保存nginx服务器的访问日志错误日志等日志，logs目录可以放在其他路径，比如/var/logs/nginx里面。

sbin：保存nginx二进制启动脚本，可以接受不同的参数以实现不同的功能。

[IMG_266](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

7、启动：

将监听端口改为8090，避免80端口冲突：

listen 8090;

8、通过命令启动和关闭nginx：

[IMG_267](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

[root@Server1 sbin]# /usr/local/nginx/sbin/nginx/nginx

nginx: [emerg] getpwnam("nginx") failed #没有nginx用户

[root@Server1 sbin]# /usr/local/nginx/sbin/nginx/nginx

nginx: [emerg] mkdir() "/var/tmp/nginx/client/" failed (2: No such file or directory) #目录不存在

[root@Server1 sbin]# /usr/local/nginx/sbin/nginx/nginx #直到没有报错，才算启动完成

[IMG_268](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

9、重读配置文件和关闭服务：

[root@Server1 local]# /usr/local/nginx/sbin/nginx/nginx #启动 服务

[root@Server1 local]# /usr/local/nginx/sbin/nginx/nginx -s reload #不停止服务重读配置文件

[root@Server1 local]# /usr/local/nginx/sbin/nginx/nginx -s stop #停止服务 #停止服务

10.验证端口是否开启：

[IMG_269](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

[root@Server1 sbin]# ps -ef | grep nginx

root 13228 1 0 Apr23 ? 00:00:00 nginx: master process /usr/local/nginx/sbin/nginx/nginx #nginx的主进程,只有一个主进程

nginx 13229 13228 0 Apr23 ? 00:00:00 nginx: worker process #nginx工作进程，默认只有一个，可以通过修改nginx.conf中的worker\_processes 1; 参数启动多个工作进程

root 13295 1400 0 00:01 pts/0 00:00:00 grep --color=auto nginx

[root@Server1 local]# lsof -i:8090 #显示占用8090的进程

COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME

nginx 13337 root 6u IPv4 5932680 0t0 TCP \*:8090 (LISTEN)

nginx 13338 nginx 6u IPv4 5932680 0t0 TCP \*:8090 (LISTEN)

[IMG_270](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

 11、通过给nginx的主进程ID号发送信号启动或停止nginx：

获取nginx主进程号的办法：

[root@Server1 nginx]# cat /var/run/nginx/nginx.pid #查看nginx的pid文件，此文件保存的就是nginx的主进程id13337 #次ID是随机的，每次启动都不一样的

[root@Server1 nginx]# ps -ef | grep nginx #过滤nginx的进程号

root 13337 1 0 00:05 ? 00:00:00 nginx: master process /usr/local/nginx/sbin/nginx/nginx

nginx 21568 13337 0 10:58 ? 00:00:00 nginx: worker process

支持的信号：

[IMG_271](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

[root@Server1 nginx]# kill -QUIT 13337 #平缓关闭Nginx，即不再接受新的请求，但是等当前请求处理完毕后再关闭Nginx。

[root@Server1 nginx]# kill -TERM 21665 #快速停止Nginx服务

[root@Server1 nginx]# kill -HUP 21703 #使用新的配置文件启动进程然后平缓停止原有的nginx进程，即平滑重启。

[root@Server1 nginx]# kill -USR1 21703 #重新打开配置文件，用于nginx 日志切割

日期切割的脚本：

#!/bin/bash

PID=`cat /var/run/nginx/nginx.pid`mv /var/log/nginx/access.log /var/log/nginx/`date +%Y\_%m\_%d:%H:%M:%S`.access.logkill -USR1 $PID

[root@Server1 nginx]# kill -USR2 21703 #使用新版本的nginx文件启动服务，然后在平缓停止原有的nginx服务，即平滑升级。

[root@Server1 nginx]# kill -WINCH 21703 #平滑停止nginx的工作进程，用于nginx平滑升级。

[IMG_272](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

三：nginx 主配置文件:nginx.conf

3.1：默认配置：配置文件默认保存在path/conf当中，默认的配置文件为nginx.conf，以下是编译安装后的默认配置：

[IMG_273](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

[root@Server1 conf]# grep -v "#" nginx.conf | grep -v "^$"

　　#全局生效，主要设置nginx的启动用户/组，启动的工作进程数量，Nginx的PID路径，日志路径等。

worker\_processes 1; #默认启动一个工作进程

events { #events设置快，主要影响nginx服务器与用户的网络连接，比如是否允许同时接受多个网络连接，使用哪种事件驱动模型处理请求，每个工作进程可以同时支持的最大连接数，是否开启对多工作进程下的网络连接进行序列化等。

worker\_connections 1024; #设置nginx可以接受的最大并发，多个进程只和

}

http { #http块是Nginx服务器配置中的重要部分，缓存、代理和日志格式定义等绝大多数功能和第三方模块都可以在这设置，http块可以包含多个server块，而一个server块中又可以包含多个location块，server块可以配置文件引入、MIME-Type定义、日志自定义、是否启用sendfile、连接超时时间和单个链接的请求上限等。

include mime.types; #文件扩展名与文件类型映射表

default\_type application/octet-stream; #默认文件类型

sendfile on; #是否调用 sendfile 函数（zero copy -->零copy方式）来输出文件，普通应用打开，可以大幅提升nginx的读文件性能，如果服务器是下载的就需要关闭，

keepalive\_timeout 65; #长连接超时时间，单位是秒

server { #设置一个虚拟机主机，可以包含自己的全局快，同时也可以包含多个locating模块。比如本虚拟机监听的端口、本虚拟机的名称和IP配置，多个server 可以使用一个端口，比如都使用8090端口提供web服务、

listen 8090; #server的全局配置，配置监听的端口

server\_name localhost; #本server的名称，当访问此名称的时候nginx会调用当前serevr内部的配置进程匹配。

location / { #location其实是server的一个指令，为nginx服务器提供比较多而且灵活的指令，都是在location中提现的，主要是基于nginx接受到的请求字符串，对用户请求的UIL进行匹配，并对特定的指令进行处理，包括地址重定向、数据缓存和应答控制等功能都是在这部分实现，另外很多第三方模块的配置也是在location模块中配置。

root html; #相当于默认页面的目录名称，默认是相对路径，可以使用绝对路径配置。

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html; #错误页面的文件名称

location = /50x.html { #location处理对应的不同错误码的页面定义到/50x.html，这个跟对应其server中定义的目录下。

root html; #定义默认页面所在的目录

}

}

}

[IMG_274](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

 3.2：配置nginx 主进程的启动用户和工作进程数：

user xxxx; #每一条指令都要以分号结尾

worker\_processes 1; #可以指定启动的固定nginx进程数，或使用auto，auto是启动与当前CPU 线程相同的进程数，如CPU是四核八线程的就启动八个进程的Nginx工作进程。

3.3：绑定Nginx 工作进程到不同的CPU上:

　　默认Nginx是不进行保定的，绑定并不能是当前nginx进程独占以一核心CPU，但是可以保证此进程不会运行在其他核心上，这就极大减少了nginx 工作进程在不同cpu上的跳转，减少了CPU对进程的资源分配与回收，因此可以有效的提升nginx服务器的性能，配置如下：

[IMG_275](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

[root@Server1 nginx]# grep process /proc/cpuinfo | wc -l #确认CPU的核心数量4

四个线程CPU的配置：

worker\_processes 4;

worker\_cpu\_affinity 0001 0010 0100 1000;

八个线程CPU的配置：

worker\_processes 8;

worker\_cpu\_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000；

[IMG_276](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

3.4：PID和错误日志文件路径：

[IMG_277](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

#pid logs/nginx.pid; #可以指定绝对路径

#error\_log logs/error.log; #指定错误日志路径

#error\_log logs/error.log notice; #指定一个日志记录级别

#error\_log logs/error.log info;

支持的日志级别：

语法:

error\_log file [ debug | info | notice | warn | error | crit ] | [{ debug\_core | debug\_alloc | debug\_mutex | debug\_event | debug\_http | debug\_mail | debug\_mysql } ]

日志级别 = 错误日志级别 | 调试日志级别; 或者

日志级别 = 错误日志级别;

错误日志的级别: emerg, alert, crit, error, warn, notic, info, debug,

调试日志的级别: debug\_core, debug\_alloc, debug\_mutex, debug\_event, debug\_http, debug\_mail, debug\_mysql

error\_log 指令的日志级别配置分为错误日志级别和调试日志级别，错误日志只能设置一个级别，而且错误日志必须书写在调试日志级别的前面，另外调试日志可以设置多个级别，其他配置方法可能无法满足需求。

[IMG_278](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

 3.5：配置文件的引入：include

[IMG_279](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

include file； #file是要导入的文件，支持相对路径，一般在html目录里面

导入一个conf文件，并配置不同主机名的页面,编辑nginx.conf主配置文件:

include /usr/local/nginx/conf.d/samsung.conf; #在最后一个大括号里面加入一项，\*是导入任何以conf结尾的配置文件

在/usr/local/nginx/conf.d/创建一个samsung.conf，内容如下：

[root@Server1 nginx]# grep -v "#" conf.d/samsung.conf | grep -v "^$"

server {

listen 8090;

server\_name samsung.chinacloudapp.cn;

location / {

root html;

index index1.html index.htm;

}

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

location = /50x.html {

root html;

}

}

[IMG_280](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

# [Nginx 之二： nginx.conf 配置及基本优化](http://www.cnblogs.com/zhang-shijie/p/5428640.html)

一：常用功能优化：

1：网络连接的优化：

　　只能在events模块设置，用于防止在同一一个时刻只有一个请求的情况下，出现多个睡眠进程会被唤醒但只能有一个进程可获得请求的尴尬，如果不优化，在多进程的nginx会影响以部分性能。

events {

accept\_mutex on; #优化同一时刻只有一个请求而避免多个睡眠进程被唤醒的设置，on为防止被同时唤醒，默认为off，因此nginx刚安装完以后要进行适当的优化。

}

2.设置是否允许同时接受多个网络连接：

　　只能在events模块设置，Nginx服务器的每个工作进程可以同时接受多个新的网络连接，但是需要在配置文件中配置，此指令默认为关闭，即默认为一个工作进程只能一次接受一个新的网络连接，打开后几个同时接受多个，配置语法如下：

events {

accept\_mutex on;

multi\_accept on; #打开同时接受多个新网络连接请求的功能。

}

3.隐藏ngxin版本号：

　　当前使用的nginx可能会有未知的漏洞，如果被黑客使用将会造成无法估量的损失，但是我们可以将nginx的版本隐藏，如下：

server\_tokens off; #在http 模块当中配置

4.：选择事件驱动模型：

　　Nginx支持众多的事件驱动，比如select、poll、epoll，只能设置在events模块中设置：

events {

accept\_mutex on;

multi\_accept on;

use epoll; #使用epoll事件驱动，因为epoll的性能相比其他事件驱动要好很多

}

5：配置单个工作进程的最大连接数：

　　通过worker\_connections number；进行设置，numebr为整数，number的值不能大于操作系统能打开的最大的文件句柄数，使用ulimit -n可以查看当前操作系统支持的最大文件句柄数，默认为为1024.

[IMG_256](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

events {

worker\_connections 102400; #设置单个工作进程最大连接数102400

accept\_mutex on;

multi\_accept on;

use epoll;

}

[IMG_257](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

6：定义MIME-Type：

　　在浏览器当中可以显示的内容有HTML/GIF/XML/Flash等内容，浏览器为取得这些资源需要使用MIME Type，即MIME是网络资源的媒体类型，Nginx作为Web服务器必须要能够识别全都请求的资源类型，在nginx.conf文件中引用了一个第三方文件，使用include导入：

include mime.types;

default\_type application/octet-stream;

7：自定义访问日志：

　　访问日志是记录客户端即用户的具体请求内容信息，全局配置模块中的error\_log是记录nginx服务器运行时的日志保存路径和记录日志的level，因此有着本质的区别，而且Nginx的错误日志一般只有一个，但是访问日志可以在不同server中定义多个，定义一个日志需要使用access\_log指定日志的保存路径，使用log\_format指定日志的格式，格式中定义要保存的具体日志内容：

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 /var/log/nginx/access.log main;

8：将日志定义为json格式：

　　在使用日志分析工具如ELK对访问日志做统计的时候，就需要将日志格式定义为json格式，以便于取相应字段的key做统计，完整的定义如下：

[IMG_258](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

log\_format logstash\_json '{"@timestamp":"$time\_iso8601",'

'"host":"$server\_addr",'

'"clientip":"$remote\_addr",'

'"size":$body\_bytes\_sent,'

'"responsetime":$request\_time,'

'"upstreamtime":"$upstream\_response\_time",'

'"upstreamhost":"$upstream\_addr",'

'"http\_host":"$host",'

'"url":"$uri",'

'"domain":"$host",'

'"xff":"$http\_x\_forwarded\_for",'

'"referer":"$http\_referer",'

'"agent":"$http\_user\_agent",'

'"status":"$status"}';

server {

　　　　listen 8090;

　　　　server\_name samsung.chinacloudapp.cn;

　　　　access\_log /var/log/nginx/samsung1.access.log logstash\_json;

　　　　location / {

　　　　　　root html;

　　　　　　index index1.html index.htm;

　　　　}

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

　　　　location = /50x.html {

　　　　root html;

　　　　　}

　　}

access\_log /var/log/nginx/json.access.log logstash\_json; #定义日志路径为/var/log/nginx/json.access.log,并引用在主配置文件nginx.conf中定义的json日志格式

[IMG_259](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

json格式的日志内如下：

[IMG_260](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

{"@timestamp":"2016-04-25T13:16:29+08:00","host":"192.168.0.202","clientip":"106.120.73.171","size":0,"responsetime":0.000,"upstreamtime":"-","upstreamhost":"-","http\_host":"samsung.chinacloudapp.cn","url":"/index1.html","domain":"samsung.chinacloudapp.cn","xff":"-","referer":"-","agent":"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.75 Safari/537.36","status":"304"}

[IMG_261](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

9：配置允许sendfile方式传输文件：

　　是由后端程序负责把源文件打包加密生成目标文件，然后程序读取目标文件返回给浏览器；这种做法有个致命的缺陷就是占用大量后端程序资源，如果遇到一些访客下载速度巨慢，就会造成大量资源被长期占用得不到释放（如后端程序占用的CPU/内存/进程等），很快后端程序就会因为没有资源可用而无法正常提供服务。通常表现就是 nginx报502错误，而sendfile打开后配合location可以实现有nginx检测文件使用存在，如果存在就有nginx直接提供静态文件的浏览服务，因此可以提升服务器性能.

　　可以配置在http、server或者location模块，配置如下：

sendfile on;

sendfile\_max\_chunk 512k; #Nginxg工作进程每次调用sendfile()传输的数据最大不能超出这个值，默认值为0表示无限制，可以设置在http/server/location模块中。

10：配置nginx工作进程最大打开文件数：

　　可以设置为linux系统最大打开的文件数量一致，在全局模块配置

worker\_rlimit\_nofile 65535;

11：会话保持时间：

　　用户和服务器建立连接后客户端分配keep-alive链接超时时间，服务器将在这个超时时间过后关闭链接，我们将它设置低些可以让ngnix持续工作的时间更长，1.8.1默认为65秒，一般不超过120秒。

keepalive\_timeout 65 60; #后面的60为发送给客户端应答报文头部中显示的超时时间设置为60s：如不设置客户端将不显示超时时间。

　　Keep-Alive:timeout=60 #浏览器收到的服务器返回的报文

如果设置为0表示关闭会话保持功能，将如下显示：

　　Connection:close #浏览器收到的服务器返回的报文

12配置网络监听：

　　使用命令listen，可以配置监听IP+端口，端口或监听unix socket:

listen 8090; #监听本机的IPV4和IPV6的8090端口，等于listen \*:8000

listen 192.168.0.1:8090; #监听指定地址的8090端口

listen Unix:/www/file #监听unix socket

 二：server部分主要配置：

1、基于域名和IP的虚拟主机

server\_name localhost www.a.com; ＃多个域名用空格间隔即可

server\_name 192.168.0.2; #IP是本机的网卡IP地址

2、location 模块正则匹配配置：

　　在没有使用正则表达式的时候，nginx会先在server中的多个location选取匹配度最高的一个uri，uri是用户请求的字符串，即域名后面的web文件路径，然后使用该location模块中的正则url和字符串，如果匹配成功就结束搜索，并使用此location处理此请求。

　　location 正则匹配的语法：

[IMG_262](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

= #用于标准uri前，需要请求字串与uri完全匹配，如果匹配成功就停止向下匹配并立即处理请求。~ #区分大小写~\* #不区分大写!~ #区分大小写不匹配!~\* #不区分大小写不匹配 ^ #匹配以什么开头

$ #匹配以什么结尾

\ #转义字符。可以转. \* ?等\* #代表任意长度的任意字符

-f和!-f #用来判断是否存在文件-d和!-d #用来判断是否存在目录-e和!-e #用来判断是否存在文件或目录-x和!-x #用来判断文件是否可执行

[IMG_263](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

 3、常见http状态码：

[IMG_264](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

200 #请求成功，即服务器返回成功301 #永久重定向302 #临时重定向403 #禁止访问，一般是服务器权限拒绝  
400 #错误请求，请求中有语法问题，或不能满足请求。    
403 #服务器权限问题导致无法显示404 #服务器找不到用户请求的页面500 #服务器内部错误，大部分是服务器的设置或内部程序出现问题  
501 #没有将正在访问的网站设置为浏览器所请求的内容502 #网关问题，是代理服务器请求后端服务器时，后端服务器不可用或没有完成 相应网关服务器，这通常是反向代理服务器下面的节点出问题导致的。503 ＃服务当前不可用，可能是服务器超载或停机导致的，或者是反向代理服务器后面没有可以提供服务的节点。504 #网关超时，一般是网关代理服务器请求后端服务器时，后端服务器没有在指定的时间内完成处理请求，多数是服务器过载导致没有在特定的时间内返回数据给前端代理服务器。  
505 #该网站不支持浏览器用于请求网页的ＨＴＴＰ协议版本（最为常见的是ＨＴＴＰ/1.1）

[IMG_265](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

 4.在server部分使用location配置一个web界面：

要求：在html/localtion/myweb 里面有个index.html文件里面写了myweb，当访问nginx 服务器的/myweb的时候要显示此html文件的内容：

[IMG_266](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

server {

listen 8090;

server\_name samsung.chinacloudapp.cn;

access\_log /var/log/nginx/samsung1.access.log logstash\_json;

location / {

root html;

index index1.html index.htm;

}

location ~/myweb { #区分大小写，即访问Myweb是不行的

root html/localtion; #定义myweb所在的路径，即在浏览器访问myweb的时候，实际是访问的html/localtion/myweb目录里面的web内容

index index.html; #默认首页文件类型

}

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

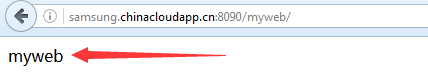
location = /50x.html {

root html;

}

[IMG_267](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

验证如下：



**三：sysctl.conf针对IPv4内核的7个参数的配置优化：**

1、net.core.netdev\_max\_backlog  #每个网络接口的处理速率比内核处理包的速度快的时候，允许发送队列的最大数目。

[root@Server1 nginx]# sysctl -a | grep max\_backlog

net.core.netdev\_max\_backlog = 1000 这里默认是1000，可以设置的大一些，比如：

net.core.netdev\_max\_backlog = 102400

2、net.core.somaxconn: #用于调节系统同时发起的TCP连接数，默认值一般为128，在客户端存在高并发请求的时候，128就变得比较小了，可能会导致链接超时或者重传问题。

net.core.somaxconn = 128 #默认为128，高并发的情况时候要设置大一些，比如：

net.core.somaxconn = 102400

3、net.ipv4.tcp\_max\_orphans:设置系统中做多允许多少TCP套接字不被关联到任何一个用户文件句柄上，如果超出这个值，没有与用户文件句柄关联的TCP套接字将立即被复位，同时给出警告信息，这个值是简单防止DDOS（Denial of service）的攻击，在内存比较充足的时候可以设置大一些：

net.ipv4.tcp\_max\_orphans = 32768 #默认为32768，可以改该打一些：

net.ipv4.tcp\_max\_orphans = 102400

4、net.ipv4.tcp\_max\_syn\_backlog #用于记录尚未收到客户度确认消息的连接请求的最大值，一般要设置大一些：

net.ipv4.tcp\_max\_syn\_backlog = 256 #默认为256，设置大一些如下：

net.ipv4.tcp\_max\_syn\_backlog = 102400

5、net.ipv4.tcp\_timestamps #用于设置时间戳，可以避免序列号的卷绕，有时候会出现数据包用之前的序列号的情况，此值默认为1表示不允许序列号的数据包，对于Nginx服务器来说，要改为0禁用对于TCP时间戳的支持，这样TCP协议会让内核接受这种数据包，从而避免网络异常，如下：

net.ipv4.tcp\_timestamps = 1 #默认为1，改为0，如下：

net.ipv4.tcp\_timestamps = 0

6、net.ipv4.tcp\_synack\_retries #用于设置内核放弃TCP连接之前向客户端发生SYN+ACK包的数量，网络连接建立需要三次握手，客户端首先向服务器发生一个连接请求，服务器收到后由内核回复一个SYN+ACK的报文，这个值不能设置过多，会影响服务器的性能，还会引起syn攻击：

net.ipv4.tcp\_synack\_retries = 5 #默认为5，可以改为1避免syn攻击

net.ipv4.tcp\_synack\_retries = 1

7、net.ipv4.tcp\_syn\_retries  #与上一个功能类似，设置为1即可：

net.ipv4.tcp\_syn\_retries = 5 #默认为5，可以改为1

net.ipv4.tcp\_syn\_retries = 1

四**：配置文件中针对CPU的2个优化参数：**

1、woker\_precess #设置Nginx 启动多少个工作进程的数量  
2、woker\_cpu\_affinit #固定Nginx 工作进程所运行的CPU核心

**五：配置文件中与网络相关的4个指令：**

[IMG_269](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

1、keepalived\_timeout 60 50； #设置Nginx服务器与客户端保持连接的时间是60秒，到60秒后服务器与客户端断开连接，50s是使用Keep-Alive消息头与部分浏览器如 chrome等的连接事件，到50秒后浏览器主动与服务器断开连接。

　　keepalived\_timeout 60 50;2、sendtime\_out 10s #Http核心模块指令，指定了发送给客户端应答后的超时时间，Timeout是指没有进入完整established状态，只完成了两次握手，如果超过这个时间客户端没有任何响应，nginx将关闭与客户端的连接。

　　sendtime\_out 10s;3、client\_header\_timeout #用于指定来自客户端请求头的headerbuffer大小，对于大多数请求，1kb的缓冲区大小已经足够，如果自定义了消息头部或有更大的cookie，可以增加缓冲区大小。

　　client\_header\_timeout 4k;4、multi\_accept #设置是否允许，Nginx在已经得到一个新连接的通知时，接收尽可能更多的连接。

multi\_accept on;

[IMG_270](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

**六：配置文件中与驱动模型相关的8个指令：**

1、use； #用于指定Nginx 使用的事件驱动模型

2、woker\_process； #指定Nginx启动的工作进程的数量

3、woker\_connections  65535； #指定Nginx 每个工作进程的最大连接数，woker\_connections  \*  woker\_process即为Nginx的最大连接数量。

4、woker\_rlimit\_sigpending 65535  #Nginx每个进程的事件信号队列的上限长度，如果超出长度，Nginx则使用poll模型处理客户的请求。

5、devpoll\_changes 和 devpoll\_events #用于设置Nginx 在/dev/poll 模型下Nginx服务器可以与内核之间传递事件的数量，前一个设置传递给内核的事件数量，后一个设置从内核读取的事件数量，默认为512。

6、kqueue\_changes 和 kqueue\_events #设置在kqueue模型下Nginx服务器可以与内核之间传递事件的数量，前一个设置传递给内核的事件数量，后一个设置从内核读取的事件数量，默认为512。

7、epoll\_events #设置在epoll驱动模式下Nginx 服务器可以与内核之间传递事件的数量，默认为512。

8、rtsig\_signo  #设置Nginx在rtsig 模式使用的两个信号中的第一个，第二个信号是在第一个信号的编号上加1.

9、rtsig\_overflow #这些参数指定如何处理rtsig队列溢出。当溢出发生在nginx清空rtsig队列时，它们将连续调用poll()和 rtsig.poll()来处理未完成的事件，直到rtsig被排空以防止新的溢出，当溢出处理完毕，nginx再次启用rtsig模式，rtsig\_overflow\_events specifies指定经过poll()的事件数，默认为16，rtsig\_overflow\_test指定poll()处理多少事件后nginx将排空rtsig队列，默认值为32，rtsig\_overflow\_threshold只能运行在Linux 2.4.x内核下，在排空rtsig队列前nginx检查内核以确定队列是怎样被填满的。默认值为1/10，“rtsig\_overflow\_threshold 3”意为1/3。

# [Nginx 之三：nginx服务器模块、web请求处理机制及事件驱动模型、进程功能和进程间通信](http://www.cnblogs.com/zhang-shijie/p/5439210.html)

一：Nginx的模块化结构设计：

1、核心模块：指的是nginx服务器运行当中必不可少的模块，这些模块提供了最基本最核心的服务，比如权限控制、进程管理、错误日志、事件驱动、正则表达式解析等,nginx的源码模块位于/root/nginx-1.8.1/src目录：

[IMG_256](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

[root@Server1 src]# pwd

/root/nginx-1.8.1/src

[root@Server1 src]# ls

core #核心模块  
event #事件模块  
http #http模块  
mail #邮件模块  
misc #其他模块  
os #系统模块

[IMG_257](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

2、标准HTTP模块：默认即被编译到了Nginx当中，除非使用--with-out-module\_name参数声明不编译，如：

[IMG_258](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

ngx\_http\_core #配置端口、URL分析、服务器响应错误处理，别名控制以及其他HTTP核心事物。

ngx\_http\_auth\_basic\_module #基于http的认证

ngx\_http\_access\_module #基于IP地址的访问控制策略

ngx\_http\_autoindex\_module #处理以“/”结尾的请求并自动生成目录列表。

ngx\_http\_browser\_module #解析HTTP请求头中的”User-Agent“ 的值。

ngx\_http\_charset\_module#指定网页的编码。  
。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。

[IMG_259](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

3、可选HTTP模块：

[IMG_260](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

ngx\_http\_perl\_module #在nginx 的配置文件中可以使用perl脚本。

ngx\_http\_flv\_module #支持flash多媒体信息文件传输。

ngx\_http\_gzip\_module #支持时时压缩

ngx\_http\_image\_filter\_module #支持JPEG,GIF和PNG的图片的尺寸和旋转方向

ngx\_http\_ssl\_module #支持对HTTPS/SSL的支持

ngx\_http\_sub\_module #支持使用指定的字符串替换相应信息的中的信息

[IMG_261](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

4、邮件模块；默认没有编译，使用的场景也不多。

[IMG_262](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

ngx\_mail\_auth\_http\_module.c

ngx\_mail.c

ngx\_mail\_core\_module.c

ngx\_mail.h

ngx\_mail\_handler.c

ngx\_mail\_imap\_handler.c

ngx\_mail\_imap\_module.c

ngx\_mail\_imap\_module.h

ngx\_mail\_parse.c

ngx\_mail\_pop3\_handler.c

ngx\_mail\_pop3\_module.c

ngx\_mail\_pop3\_module.h

ngx\_mail\_proxy\_module.c

ngx\_mail\_smtp\_handler.c

ngx\_mail\_smtp\_module.c

ngx\_mail\_smtp\_module.h

ngx\_mail\_ssl\_module.c

ngx\_mail\_ssl\_module.h

[IMG_263](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

5、第三方模块：

echo-nginx-module #支持在配置文件中使用echo、sleep、time即exec等类似shell命令。

memc-nginx-module #对标准http模块ngx\_http\_memcached\_module的扩展，支持set、add、delete等命令

lua-nginx-module #支持lua脚本语言

6、安装 echo-nginx-module 模块：

6.1：模块地址： https://github.com/openresty/echo-nginx-module.git

6.2：进到在nginx源码目录下，执行编译安装echo-nginx-module模块：

cd nginx-1.8.1

./configure --prefix=/usr/local/nginx/ --add-module=/home/tianqi/echo-nginx-module-master

make

make install

6.3：配置nginx.conf：

[IMG_264](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

server {

listen 80;

server\_name hfnginx.chinacloudapp.cn;

location / {

root html;

index index.html index.htm;

echo $remote\_addr;

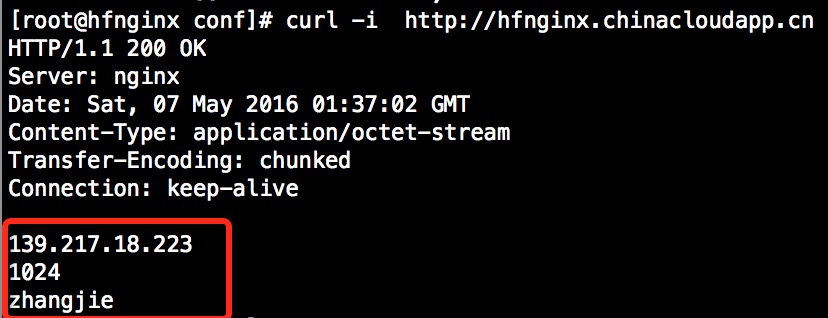
echo $remote\_port;

echo "zhangjie";

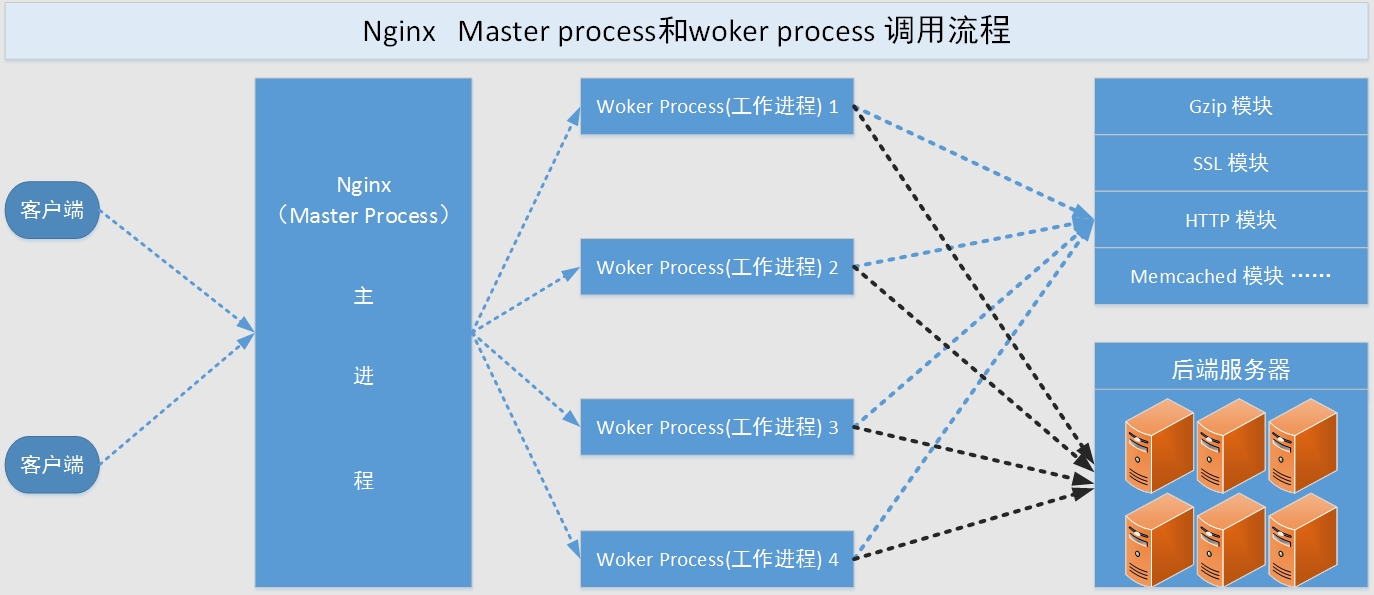
}

[IMG_265](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

6.4:访问测试：



6.5:Nginx模块组织工作图：



二：web请求处理机制：

1、多进程方式：服务器没接受到一个客户端请求就有服务器的主进程生成一个子进程响应客户端，直到用户关闭连接，这样的优势是处理速度快，子进程之间相互独立，但是如果访问过大会导致服务器资源耗尽而无法提供请求。

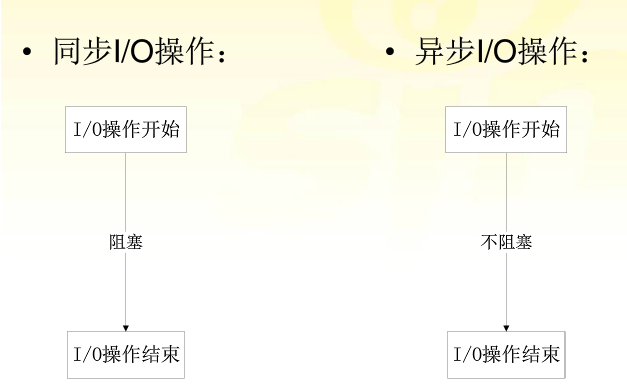
2、多线程方式：与多进程方式类似，但是每收到一个客户端请求会有服务进程派生出一个线程来个客户方进行交互，一个线程的开销远远小于一个进程，因此多线程方式在很大程度减轻了web服务器对系统资源的要求，但是多线程也有自己的缺点，即当多个线程位于同一个进程内工作的时候，可以相互访问同样的内存地址空间，所以他们相互影响，一旦主进程挂掉则所有子线程都不能工作了，IIS服务器使用了多线程的方式，需要间隔一段时间就重启一次才能稳定。

三：同步和异步、阻塞与非阻塞：

1、同步与异步：主要是针对应用程序与内核的交互方式而言的：

同步：进程发出数据后，等内核返回响应以后才继续下一个请求，即如果内核一直不返回数据，那么进程就一直等，直到天荒地老，死机error。

异步：进程发出数据后，不等内核返回响应，接着处理下一个请求,Nginx是异步的。



2、阻塞与非阻塞：

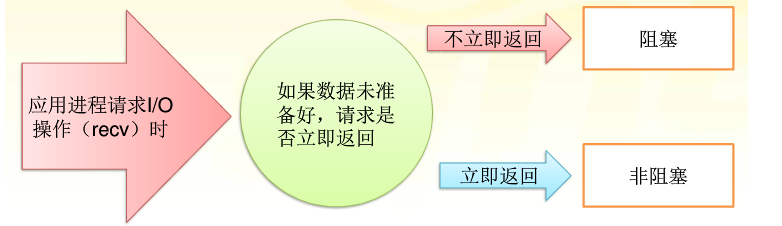
可以理解为内核与IO设备的交互方式，当内核收到进程请求IO数据时候的处理方式：

也可以简单理解为内核需要做一件事能不能立即得到返回应答，如果不能立即获得返回，需要等待，那就阻塞了，否则就可以理解为非阻塞。

阻塞：IO调用不能立即返回结果，即一个进程发起的IO请求不能得到立即满足时，进程就要一直等到内核响应，内核要把数据从IO设备复制到内核空间，再返回给进程，这是阻塞。

非阻塞：IO调用可以立即返回结果，一个进程发起的IO进程不能立即满足时，不在等待，而是一遍一遍的轮训查看IO是否完成。

详细区别如下图所示：



3、混合总结：

3.1：同步阻塞：程序进程向内核发送IO请求后一直等待内核响应，如果内核处理请求的IO操作不能立即返回,则进程将一直等待并不再接受新的请求，并由进程轮训查看IO是否完成，完成后进程将IO结果返回给Client，在IO没有返回期间进程不能接受其他客户的请求，而且是有进程自己去查看IO是否完成，这种方式简单，但是比较慢，用的比较少。

3.2：同步非阻塞：，进程向内核发送请IO求后一直等待内核响应，如果内核处理请求的IO操作不能立即返回IO结果，进程将不再等待，而且继续处理其他请求，但是仍然需要进程隔一段时间就要查看内核IO是否完成。

3.3：异步阻塞：event-diver，进程向内核发送IO调用后，不用等待内核响应，可以继续接受其他请求，内核收到进程请求后进行的IO如果不能立即返回，就由内核等待结果，直到IO完成后内核再通知进程，此方式比较占用内核，因此也很少使用。

3.4：异步非阻塞：AIO,进程向内核发送IO调用后，不用等待内核响应，可以继续接受其他请求，内核调用的IO如果不能立即返回，内核会继续处理其他事物，直到IO完成后将结果通知给内核，内核在将IO完成的结果返回给进程，期间进程可以接受新的请求，内核也可以处理新的事物，因此相互不影响，可以实现较大的同时并实现较高的IO复用，因此异步非阻塞使用最多的一种通信方式。

 四：Nginx支持的事件驱动模型：

 1、select：

select库是在linux和windows平台都基本支持的 事件驱动模型库，并且在接口的定义也基本相同，只是部分参数的含义略有差异，最大并发限制1024，只最早期的事件驱动模型。

2、poll：

在Linux 的基本驱动模型，windows不支持此驱动模型，是select的升级版，取消了最大的并发限制，在编译nginx的时候可以使用--with-poll\_module和--without-poll\_module这两个指定是否编译select库。

3、epoll：

epoll是库是Nginx服务器支持的最高性能的事件驱动库之一，是公认的非常优秀的事件驱动模型，它和select和poll有很大的区别，epoll是poll的升级版，但是与poll的效率有很大的区别.  
epoll的处理方式是创建一个待处理的事件列表，然后把这个列表发给内核，返回的时候在去轮训检查这个表，以判断事件是否发生，epoll支持一个进程打开的最大事件描述符的上限是系统可以打开的文件的最大数，同时epoll库的IO效率不随描述符数目增加而线性下降，因为它只会对内核上报的“活跃”的描述符进行操作。

4、rtsig：

不是一个常用事件驱动，最大队列1024，不是很常用

5、kqueue：

用于支持BSD系列平台的高校事件驱动模型，主要用在FreeBSD 4.1及以上版本、OpenBSD 2.0级以上版本，NetBSD级以上版本及Mac OS X 平台上，该模型也是poll库的变种，因此和epoll没有本质上的区别，都是通过避免轮训操作提供效率。

 6、/dev/poll:

用于支持unix衍生平台的高效事件驱动模型，主要在Solaris 平台、HP/UX，该模型是sun公司在开发Solaris系列平台的时候提出的用于完成事件驱动机制的方案，它使用了虚拟的/dev/poll设备，开发人员将要见识的文件描述符加入这个设备，然后通过ioctl()调用来获取事件通知，因此运行在以上系列平台的时候请使用/dev/poll事件驱动机制。

7、eventport：

该方案也是sun公司在开发Solaris的时候提出的事件驱动库，只是Solaris 10以上的版本，该驱动库看防止内核崩溃等情况的发生。

五：Nginx 进程的功能和进程间的通信：

1、主进程(woker process)的功能：

[IMG_270](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

读取Nginx 配置文件并验证其有效性和正确性

建立、绑定和关闭socket连接

按照配置生成、管理和结束工作进程

接受外界指令，比如重启、升级及退出服务器等指令

不中断服务，实现平滑升级，重启服务并应用新的配置

开启日志文件，获取文件描述符

不中断服务，实现平滑升级，升级失败进行回滚处理

编译和处理perl脚本

[IMG_271](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

2、工作进程（woker process）的功能：

[IMG_272](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

接受处理客户的请求

将请求以此送入各个功能模块进行处理

IO调用，获取响应数据

与后端服务器通信，接收后端服务器的处理结果

缓存数据，访问缓存索引，查询和调用缓存数据

发送请求结果，响应客户的请求

接收主程序指令，比如重启、升级和退出等

[IMG_273](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

3、Ninx进程间的通信：

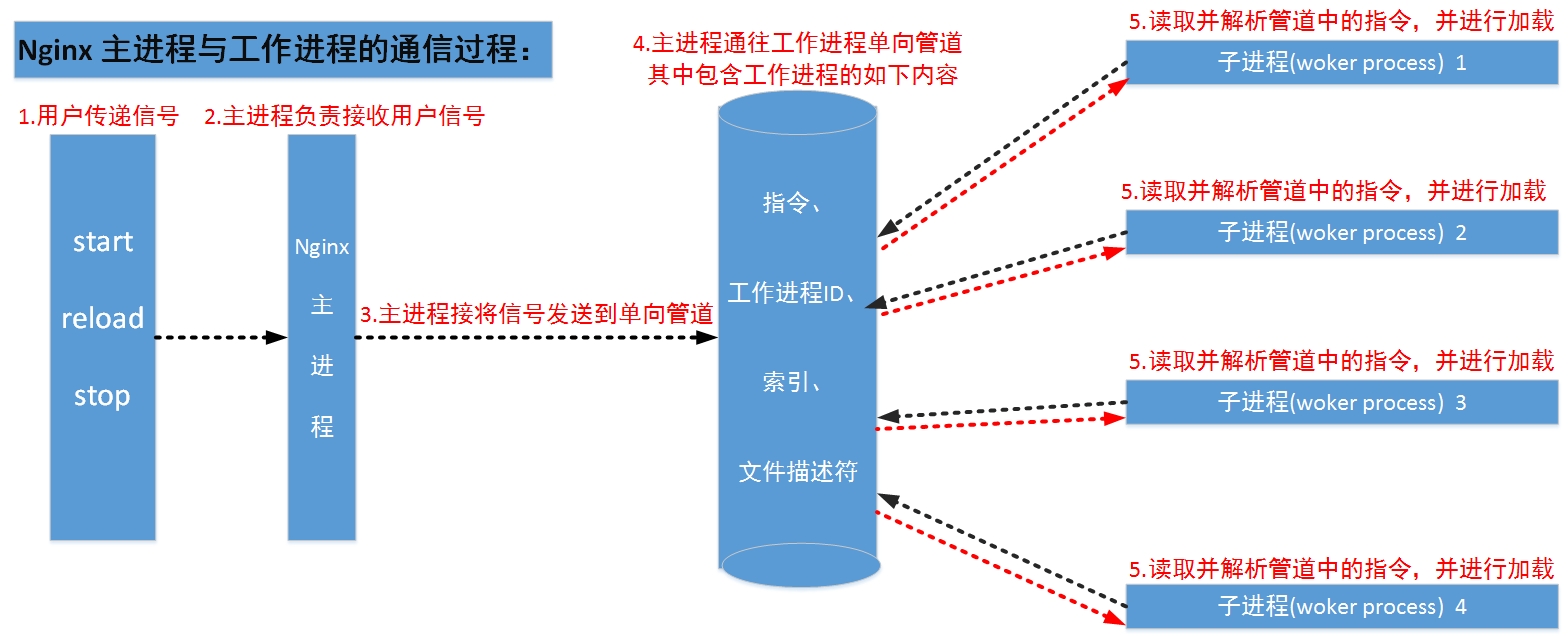
3.1：主进程和工作进程之间的通信：

[IMG_274](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

工作进程是有主进程生成的，主进程使用fork()函数，在Nginx服务器启动过程中主进程根据配置文件决定启动工作进程的数量，然后建立一张全局的工作表用于存放当前未退出的所有的工作进程，主进程生成工作进程后会将新生成的工作进程加入到工作进程表中，并建立一个单向的管道并将其传递给工作进程，该管道与普通的管道不同，它是由主进程指向工作进程的单项通道，包含了主进程想工作进程发出的指令、工作进程ID、工作进程在工作进程表中的索引和必要的文件描述符等信息。

主进程与外界通过信号机制进行通信，当接收到需要处理的信号时，它通过管道向相关的工作进程发送正确的指令，每个工作进程都有能力捕获管道中的可读事件，当管道中有可读事件的时候，工作进程就会从管道中读取并解析指令，然后采取相应的执行动作，这样就完成了主进程与工作进程的交互。

[IMG_275](http://www.cnblogs.com/zhang-shijie/p/javascript:void(0);)

 3.2：工作进程与工作进程之间的通信：

工作进程之间的通信原理基本上和主进程与工作进程之间的通信是一样的，只要工作进程之间能够取得彼此的信息，建立管道即可通信，但是由于工作进程之间是完全隔离的，因此一个进程想要直到另外一个进程的状态信息就只能通过主进程来设置了。

为了实现工作进程之间的交互，主进程在生成工作进程只之后，在工作进程表中进行遍历，将该新进程的ID以及针对该进程建立的管道句柄传递给工作进程中的其他进程，为工作进程之间的通信做准备，当工作进程1向工作进程2发送指令的时候，首先在主进程给它的其他工作进程工作信息中找到2的进程ID，然后将正确的指令写入指向进程2的管道，工作进程2捕获到管道中的事件后，解析指令并进行相关操作，这样就完成了工作进程之间的通信。

# [Nginx 之四： Nginx服务器的压缩功能和缓存功能](http://www.cnblogs.com/zhang-shijie/p/5451919.html)

　　在Nginx服务器配置文件中可以通过配置Gzip的使用，可以配置在http块，server 块或者location块中设置，Nginx服务器可以通过ngx\_http\_gzip\_module模块、ngx\_http\_static\_module 模块功能ngx\_http\_gunzip\_module模块对这些指令进行分析和处理。

**一：ngx\_http\_gzip\_module模块处理的指令：**

1、gzip：

用于打开或者关闭gzip压缩的功能，用法如下：

gzip on | off ; #默认为off

2、gzip\_buffers：

用于指定gzip功能压缩文件时使用的缓存空间的大小，语法结构为：

gzip\_buffers numner | size ；

number #指定Nginx服务器需要向服务器申请的缓存空间的个数

size #指定每个缓存空间的大小，从Nginx 0.7.8开始，默认number \* siez的的值为128，其中size取系统中内存页一页的大小，为4k或者8k等  
  
gzip\_buffers 32 4k | 12 8k；

3、gzip\_comp\_level

　指定gzip压缩级别，级别为1-9，默认为1，级别越高压缩率越高即压缩后的文件越小，但是越占用CPU，压缩率越小压缩后的文件越大，但是节约CPU。

gzip\_comp\_level 4； #推荐先设置为中间的值，比如4或者5

4、gzip\_disbale

针对不同客户端发起的请求进行有选择的打开或关闭gzip命令，后面跟浏览器的名称，用法如下：

gzip\_disable MSIE[1-6]\. #使用正则表达式可以匹配UC字符串中MSIE 1-6 的浏览器，这三个浏览器在发起请求后Nginx将不进行数据包的压缩。

5、gzip\_http\_verson

早期的一些浏览器或者http客户端可能不支持gzip压缩，因此有客户有时候会看到乱码，那是应为客户端无法进行解压，因此可以根据不同的http协议版本进行选择性的打开或关闭gzip功能，用法如下：

gzip\_http\_version 1.0 | 1.1; #对使用htto协议1.0和1.1的请求进行压缩

6、gzip\_min\_length

设置最少压缩的数据包大小，小于此值的数据包将不被压缩，因为大量的小的数据压缩后没有明显的效果还占用了CPU的资源，因此可以设置最少多大的数据才进行压缩，用法如下：

gzip\_min\_length 1024; #默认为20，建议设置位1KB，如果设置为0则表示压缩全部数据包

7、gzip\_proxine

在Nginx 服务器作为反向代理的时候有效，用于设置nginx 服务器是否对后端返回的结果进行gzip压缩，用法如下：

gzip\_proxied off | any; #off为关闭，any为压缩所有后端服务器返回的数据。

8、gzip\_types

设置Nginx服务器可以根据响应页面的类型进行选择性的打开或关闭gzip功能，用法如下：

gzip\_types text/plain application/x-javascript text/css application/xml; #一般情况下如此设置

9、gzip\_vary

用于设置是否在使用gzip功能时发送带有"vary:Accept-Encoding" 头域的响应头部，该头域的主要功能时要告诉客户端数据已经在服务器进行了压缩，默认设置为off，用法如下：

gzip\_vary on| off; #默认为off

还可以使用Nginx 配置的add\_header指令强制在Nginx服务器的响应头部添加“Vary:Accept-Encoding”也可以实现相同的效果。

**二：ngx\_http\_gzip\_static\_module 模块：**

该模块辅助用于搜索和发送经过gzip功能压缩的数据，这些数据以\*.gz作为后缀名存储在服务器上，如果客户端请求的数据之前被压缩过，并且客户端的浏览器支持压缩，将直接范湖压缩后的数据，该模块使用的是静态编码，在http响应头部包含content-length头域来指明报文的长度，用于服务器可以确定响应数据的长度的情况，而ngx\_http\_gzip\_module使用chunked编码动态压缩，主要用于服务器无法确定响应数据长度的情况，比如较大文件的下载等情形，此时就要实时生成数据的长度，用法与ngx\_http\_gzip\_module一样，如下：

1、gzip\_static

gzip\_static off | on | always; #on为开启并检查客户端浏览器是否中吃gzip压缩功能，off为关闭，always一直发送gzip压缩文件，而不检查浏览器是否支持gzip压缩

 注：这是一个可选模块，如果要使用，必须在编译nginx时添加--with-http\_gzip\_static\_module编译，与该模块有关的 指令与**ngx\_http\_gzip\_module**的模块使用方法是一致的，本模块有gzip\_static、gzip\_http\_version、gzip\_proxied、gzip\_disable和gzip\_vary等。

**三：ngx\_http\_gunzip\_module 模块处理的两个指令**

1、gunzip

设置Nginx服务器对不支持gzip的客户端返回解压后的数据，如果客户的浏览器支持压缩还仍然返回压缩的后的数据，及可以同时响应支持gzip的浏览器进程压缩，有可以响应不支持gzip的浏览器。

gunzip on|off ；#on为打开off为关闭

2、gunzip\_buffers #用于设置nginx服务器解压gzip文件时候使用的缓存空间大小的，用法如下：

gunzip\_buffers number size； #number为nginx服务器向系统向系统申请缓存空间的个数，size为每个空间的大小，单位为k，默认情况下number \* size的大小为128k，其中size 的值取系统内存页一页的大小为4KB或者8KB即可，用法如下：

gunzip\_buffers 32 4k | 16 8k; #与ngx\_http\_gzip\_module模块下的使用方法一致。

注：这是一个可选模块，如果要使用需要在nginx 编译安装的时候使用--with-http\_gunzip\_module 指令进行编译

四：缓存功能：

1.memcached缓存数据库常用数据。

2.xcache缓存php编译的结果，可以在多个进程之间共享xcode结果。

3.nginx的缓存是缓存opcode处理后的结果，避免客户端请求同一个页面而导致程序每次都处理一下，即再访问同一个页面，直接使用处理后的缓存即可直接返回。

五：正向代理和反向代理：

1.正向代理:工作在用户前端，用户请求的所有内容都经过代理服务器。

2.反向代理：工作在服务器前端，只代理请求次服务器的客户端请求，被其代理的服务器叫做"代理服务器"或者"上游服务器"。