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是由俄罗斯人 Igor Sysoev为俄罗斯访问量第二的 Rambler.ru站点开发.

Nginx以事件驱动（epoll）的方式编写，所以有非常好的性能，同时也是一个非常高效的反向代理、负载平衡。但是Nginx并不支持cgi方式运行，原因是可以减少因此带来的一些程序上的漏洞。所以必须使用FastCGI方式来执行PHP程序。

**由于Nginx本身的一些优点，轻量，开源，易用，越来越多的公司使用nginx作为自己公司的web应用服务器，介绍nginx源码安装的同时并对nginx进行优化配置。**

**一、Nginx的优化**

**1、编译安装前优化**

**编译前的优化主要是用来修改程序名等等，**目的更改源码隐藏软件名称和版本号

安装zlib-devel、pcre-devel等依赖包

[root@www ~]# yum -y install gcc gcc-c++ make libtool zlib zlib-devel pcre pcre-devel openssl openssl-devel

下载nginx的源码包：<http://nginx.org/download>

解压源码包：

[root@www ~]# tar zxf nginx-1.10.2.tar.gz

[root@www ~]# cd nginx-1.10.2/

隐藏软件名称和版本号

[root@www nginx-1.10.2]# vim src/core/nginx.h

//此行修改的是你想要的版本

#define NGINX\_VERSION "1.10.2" //第13行

//此行修改的是你想修改的软件名称

#define NGINX\_VER "nginx/" NGINX\_VERSION //第14行

修改上面的信息，即可更改nginx显示版本。例如：

#define NGINX\_VERSION "7.0"

#define NGINX\_VER "IIS/" NGINX\_VERSION

修改HTTP头信息中的connection字段，防止回显具体版本号

拓展：通用http头，通用头包含请求和响应消息都支持的头，通用头包含Cache-Control、 Connection、Date、Pragma、Transfer-Encoding、Upgrade、Via。对通用头的扩展要求通讯双方都支持此扩展，如果存在不支持的通用头，一般将会作为实体头处理。那么也就是说有部分设备，或者是软件，能获取到connection，部分不能，要隐藏就要彻底！

[root@www nginx-1.10.2]# vi src/http/ngx\_http\_header\_filter\_module.c

修改前：

**static char ngx\_http\_server\_string[] = "Server: nginx" CRLF; //第49行**

修改后：

static char ngx\_http\_server\_string[] = "Server: IIS" CRLF;

定义了http错误码的返回

有时候我们页面程序出现错误，Nginx会代我们返回相应的错误代码，回显的时候，会带上nginx和版本号，我们把他隐藏起来

[root@www nginx-1.10.2]# vi src/http/ngx\_http\_special\_response.c

修改前

static u\_char ngx\_http\_error\_tail[] = //第29行

"<hr><center>nginx</center>" CRLF

"</body>" CRLF

"</html>" CRLF

;

修改后

static u\_char ngx\_http\_error\_tail[] =

"<hr><center>IIS</center>" CRLF

"</body>" CRLF

"</html>" CRLF

;

2、安装ngnix

[root@www ~]# groupadd www #添加www组

[root@www ~]# useradd -g www www -s /sbin/nologin  #创建nginx运行账户www并加入到www组，不允许www用户直接登录系统

[root@www nginx-1.10.2]# ./configure --prefix=/usr/local/nginx1.10 --with-http\_dav\_module --with-http\_stub\_status\_module --with-http\_addition\_module --with-http\_sub\_module --with-http\_flv\_module --with-http\_mp4\_module --with-pcre --with-http\_ssl\_module --with-http\_gzip\_static\_module

--user=www --group=www

[root@www nginx-1.10.2]# make&& make install

相关选项说明

--with-http\_dav\_module #增加PUT,DELETE,MKCOL：创建集合，COPY和MOVE方法

--with-http\_stub\_status\_module #获取Nginx的状态统计信息

--with-http\_addition\_module #作为一个输出过滤器，支持不完全缓冲，分部分相应请求

--with-http\_sub\_module #允许一些其他文本替换Nginx相应中的一些文本

--with-http\_flv\_module #提供支持flv视频文件支持

--with-http\_mp4\_module #提供支持mp4视频文件支持，提供伪流媒体服务端支持

--with-http\_ssl\_module #启用ngx\_http\_ssl\_module

如果pcre是通过编译安装的话，例如

# tar zxvf /usr/local/src/pcre-8.36.tar.gz -C /usr/local/src/

# cd /usr/local/src/pcre-8.36

# ./configure &&　make && make install

则--with-pcre=/usr/local/src/pcre-8.36 #需要注意，这里指的是源码,用#./configure --help |grep pcre查看帮助

[root@www nginx-1.10.2]# ln -s /usr/local/nginx1.10/sbin/nginx /usr/local/sbin/

[root@www nginx-1.10.2]# nginx –t

更改nginx安装目录所属组和用户，否则可能出现403

chown nginx:nginx -R /usr/local/nginx1.10/

启动nginx

[root@www nginx-1.10.2]# nginx

[root@www nginx-1.10.2]# netstat -anpt | grep nginx

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 9834/nginx: master

**测试是否隐藏了版本和软件名**

[root@www ~]# curl -I http://127.0.0.1

HTTP/1.1 200 OK

Server: IIS/7.0

Date: Sat, 05 Nov 2016 14:38:21 GMT

Content-Type: text/html

Content-Length: 612

Last-Modified: Sat, 05 Nov 2016 14:19:47 GMT

Connection: keep-alive

ETag: "581dea83-264"

Accept-Ranges: bytes

[root@www ~]# nginx -h

nginx version: IIS/7.0

Usage: nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:

-v : show version and exit

-V : show version and configure options then exit

-t : test configuration and exit

-T : test configuration, dump it and exit

-q : suppress non-error messages during configuration testing

-s signal : send signal to a master process: stop, quit, reopen, reload

-p prefix : set prefix path (default: /usr/local/nginx1.10/)

-c filename : set configuration file (default: conf/nginx.conf)

-g directives : set global directives out of configuration file

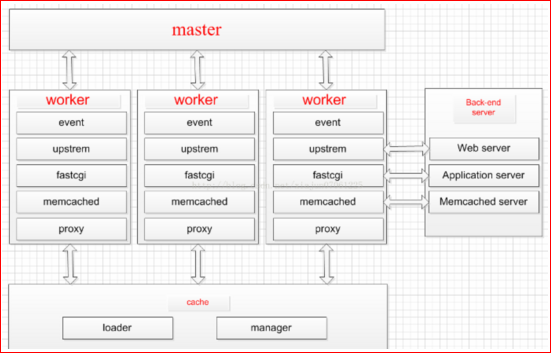
3、nginx配置项优化

[root@www ~]# ps -ef | grep nginx

root 9834 1 0 22:36 ? 00:00:00 nginx: master process nginx

www 9953 9834 0 22:43 ? 00:00:00 nginx: worker process

在这里我们还可以看到在查看的时候，work进程是nginx程序用户，但是master进程还是root，其中，master是监控进程，也叫主进程，work是工作进程，部分还有cache相关进程，关系如图：



可以直接理解为master是管理员，work进程才是为用户提供服务的！

**(1):Nginx运行工作进程个数，一般我们设置CPU的核心或者核心数x2**

如果不了解cpu的核数，可以top命令之后按1也可以看出来，也可以查看/proc/cpuinfo文件#grep ^processor /proc/cpuinfo | wc -l

[root@www ~]# vi /usr/local/nginx1.10/conf/nginx.conf

worker\_processes 4;

[root@www ~]# /usr/local/nginx1.10/sbin/nginx -s reload

[root@www ~]# ps -aux | grep nginx | grep -v grep

root 9834 0.0 0.0 47556 1948 ? Ss 22:36 0:00 nginx: master process nginx

www 10135 0.0 0.0 50088 2004 ? S 22:58 0:00 nginx: worker process

www 10136 0.0 0.0 50088 2004 ? S 22:58 0:00 nginx: worker process

www 10137 0.0 0.0 50088 2004 ? S 22:58 0:00 nginx: worker process

www 10138 0.0 0.0 50088 2004 ? S 22:58 0:00 nginx: worker process

**Nginx运行CPU亲和力**

比如4核配置

worker\_processes 4;

worker\_cpu\_affinity 0001 0010 0100 1000

比如8核配置

worker\_processes 8;

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

worker\_processes最多开启8个，8个以上性能提升不会再提升了，而且稳定性变得更低，所以8个进程够用了。

Nginx最多可以打开文件数

worker\_rlimit\_nofile 65535;

这个指令是指当一个nginx进程打开的最多文件描述符数目，理论值应该是最多打开文件数（ulimit -n）与nginx进程数相除，但是nginx分配请求并不是那么均匀，所以最好与ulimit -n的值保持一致。

注：

文件资源限制的配置可以在/etc/security/limits.conf设置，针对root/user等各个用户或者\*代表所有用户来设置。

\*     soft   nofile   65535

\*     hard  nofile    65535

用户重新登录生效（ulimit -n）

**(2)Nginx事件处理模型**

events {

use epoll;

worker\_connections 65535;

multi\_accept on;

}

nginx采用epoll事件模型，处理效率高

work\_connections是单个worker进程允许客户端最大连接数，这个数值一般根据服务器性能和内存来制定，实际最大值就是worker进程数乘以work\_connections

实际我们填入一个65535，足够了，这些都算并发值，一个网站的并发达到这么大的数量，也算一个大站了！

**multi\_accept** 告诉nginx收到一个新连接通知后接受尽可能多的连接，默认是on，设置为on后，多个worker按串行方式来处理连接，也就是一个连接只有一个worker被唤醒，其他的处于休眠状态，设置为off后，多个worker按并行方式来处理连接，也就是一个连接会唤醒所有的worker，直到连接分配完毕，没有取得连接的继续休眠。当你的服务器连接数不多时，开启这个参数会让负载有一定的降低，但是当服务器的吞吐量很大时，为了效率，可以关闭这个参数。

**(3)开启高效传输模式**

http {

include mime.types;

default\_type application/octet-stream;

……

sendfile on;

tcp\_nopush on;

……

Include mime.types; //媒体类型,include 只是一个在当前文件中包含另一个文件内容的指令

default\_type application/octet-stream; //默认媒体类型足够

sendfile on；//开启高效文件传输模式，sendfile指令指定nginx是否调用sendfile函数来输出文件，对于普通应用设为 on，如果用来进行下载等应用磁盘IO重负载应用，可设置为off，以平衡磁盘与网络I/O处理速度，降低系统的负载。

注意：如果图片显示不正常把这个改成off。

tcp\_nopush on；必须在sendfile开启模式才有效，防止网路阻塞，积极的减少网络报文段的数量（将响应头和正文的开始部分一起发送，而不一个接一个的发送。）

**(4)连接超时时间**

主要目的是保护服务器资源，CPU，内存，控制连接数，因为建立连接也是需要消耗资源的

keepalive\_timeout 60;

tcp\_nodelay on;

client\_header\_buffer\_size 4k;

open\_file\_cache max=102400 inactive=20s;

open\_file\_cache\_valid 30s;

open\_file\_cache\_min\_uses 1;

client\_header\_timeout 15;

client\_body\_timeout 15;

reset\_timedout\_connection on;

send\_timeout 15;

server\_tokens off;

client\_max\_body\_size 10m;

keepalived\_timeout客户端连接保持会话超时时间，超过这个时间，服务器断开这个链接

tcp\_nodelay；也是防止网络阻塞，不过要包涵在keepalived参数才有效

client\_header\_buffer\_size 4k;  
客户端请求头部的缓冲区大小，这个可以根据你的系统分页大小来设置，一般一个请求头的大小不会超过 1k，不过由于一般系统分页都要大于1k，所以这里设置为分页大小。分页大小可以用命令getconf PAGESIZE取得。  
open\_file\_cache max=102400 inactive=20s;  
这个将为打开文件指定缓存，默认是没有启用的，max指定缓存数量，建议和打开文件  
数一致，inactive 是指经过多长时间文件没被请求后删除缓存。  
open\_file\_cache\_valid 30s;  
这个是指多长时间检查一次缓存的有效信息。  
open\_file\_cache\_min\_uses 1;  
open\_file\_cache指令中的inactive 参数时间内文件的最少使用次数，如果超过这个数字，文  
件描述符一直是在缓存中打开的，如上例，如果有一个文件在inactive 时间内一次没被使用，它将被移除。

client\_header\_timeout设置请求头的超时时间。我们也可以把这个设置低些，如果超过这个时间没有发送任何数据，nginx将返回request time out的错误

client\_body\_timeout设置请求体的超时时间。我们也可以把这个设置低些，超过这个时间没有发送任何数据，和上面一样的错误提示

**reset\_timeout\_connection**告诉nginx关闭不响应的客户端连接。这将会释放那个客户端所占有的内存空间。

send\_timeout响应客户端超时时间，这个超时时间仅限于两个活动之间的时间，如果超过这个时间，客户端没有任何活动，nginx关闭连接

**server\_tokens**并不会让nginx执行的速度更快，但它可以关闭在错误页面中的nginx版本数字，这样对于安全性是有好处的。

client\_max\_body\_size上传文件大小限制