Nginx web服务 调度器器 4层代理服务

**Nginx 简介**

--源码安装依赖包--解压--./configure (添加模块)--make--make install

--with-http\_ssl\_module：提供SSL加密功能

--with-http\_stub\_status\_module 开启nginx的状态查询页面

--ngx\_http\_core\_module（核心模块，包含http、server\_name、root等配置参数）

--ngx\_http\_access\_module（访问控制模块，包含allow和deny配置参数）

--ngx\_http\_auth\_basic\_module（用户认证模块，包含auth\_basic等配置参数）

--ngx\_http\_charset\_module（字符集模块，包含charset utf8等配置参数）

--ngx\_http\_fastcgi\_module（fastcgi模块，包含fastcgi\_pass等配置参数）

--ngx\_http\_gzip\_module（压缩模块，包含gzip、gzip\_type等配置参数）

--ngx\_http\_limit\_conn\_module（限制并发量模块，包含limit\_conn等参数）

--ngx\_http\_log\_module（日志模块，包含access\_log等配置参数）

--ngx\_http\_proxy\_module（代理模块，包含proxy\_pass等配置参数）

--ngx\_http\_rewrite\_module（地址重写模块，包含rewrite、break、last等配置参数）

--ngx\_http\_ssl\_module（加密模块，包含ssl\_certificate、ssl\_certificate\_key等参数）

--ngx\_http\_stub\_status\_module（状态模块，包含stub\_status配置参数）

--ngx\_http\_upstream\_module(调度器模块，包含upstream、hash、ip\_hash等配置参数)

--ngx\_stream\_core\_module（4层代理模块）

--在proxy模块中有proxy\_pass、proxy\_cache、proxy\_cache\_path、proxy\_connect\_timeout、 proxy\_limit\_rate等参数）

--yum -y install gcc openssl-devel pcre-devel nginx依赖包

（具体作用可以查百度）

--group：指定组

--user：指定账户

--依赖包：gcc(编译工具) pcre-devel（正则）openssl-deve(加密网站依赖)

--/usr/local/nginx //默认安装目录

--conf/nginx.conf //主配置文件

--html //网页文件目录

--logs //日志文件

--sbin/nginx //启动脚本

**用户认证**

--访问Web页面需要进行用户认证

--用户名为：tom，密码为：12345

--通过Nginx实现Web页面的认证，需要修改Nginx配置文件，在配置文件中添加auth语句实现用户认证。最后使用htpasswd命令创建用户及密码即可。

**基于域名的虚拟主机**

--修改Nginx配置文件，添加server容器实现虚拟主机功能；

对于需要进行用户认证的虚拟主机添加auth认证语句。

--虚拟主机一般可用分为：基于域名、基于IP和基于端口的虚拟主机

**ssl虚拟主机（htpps加密网站）**

--域名为www.c.com

--该站点通过https访问

--通过私钥、证书对该站点所有数据加密

--源码安装Nginx时必须使用--with-http\_ssl\_module参数，启用加密模块，对于需要进行SSL加密处理的站点添加ssl相关指令（设置网站需要的私钥和证书）。

--加密算法一般分为对称算法、非对称算法、信息摘要。

--对称算法有：AES、DES，主要应用在单机数据加密。

--非对称算法有：RSA、DSA，主要应用在网络数据加密。

--信息摘要：MD5、sha256，主要应用在数据完整性校验。

**LNMP动态web网站**

--安装部署Nginx、MariaDB、PHP环境

--安装部署Nginx、MariaDB、PHP、PHP-FPM；

--启动Nginx、MariaDB、FPM服务；

--并测试LNMP是否工作正常。

--目前的网站一般都会有动态和静态数据，默认nginx仅可以处理静态数据，用户访问任何数据都是直接返回对应的文件，如果如果访问的是一个脚本的话，就会导致直接返回一个脚本给用户，而用户没 有脚本解释器，也看不懂脚本源代码！网站拓扑如图-1所示。

因此需要整合LNMP（Linux、Nginx、MySQL、PHP）实现动态网站效果。

**Nginx的地址重写**

--所有访问a.html的请求，重定向到b.html;

--所有访问192.168.4.5的请求重定向至www.tmooc.cn；

--所有访问192.168.4.5/下面子页面，重定向至www.tmooc.cn/下相同的页面；

--实现firefox与curl访问相同页面文件，返回不同的内容。

--关于Nginx服务器的地址重写，主要用到的配置参数是rewrite：

--rewrite regex replacement flag

--rewrite 旧地址 新地址 [选项]

**Httpd apache基金会 传统工具 功能比较齐全**

**Nginx 支持高并发**

**搭建Nginx服务器**

--源码安装依赖包--解压--./configure (添加模块)--make--make install

--with-http\_ssl\_module：提供SSL加密功能

--group：指定组

--user：指定账户

--依赖包：gcc(编译工具) pcre-devel（正则）openssl-deve(加密网站依赖)

将lnmp\_soft软件拷贝到虚拟机 然后释放

tar -xf lnmp\_soft.gz

cd lnmp\_soft

Tar -xf nginx-1.12.2.tar.gz

cd nginx-1.12.2/

Yum -y install gcc pcre-devel openssl-devel 安装依赖包

gcc 是编译工具 pcre-devel 是让nginx支持正则 openssl-devle是让nginx支持安全的网络访问(加密，https)

./configure --help

--user=nginx \ #指定用户

--group=nginx \ #指定组

--with-http\_ssl\_module #指定安全模块（ssl加密）

--prefix=/usr/local/nginx # 安装路径

./configure --with-http\_ssl\_module 配置nginx功能，添加ssl

make 编译

make install 安装程序（拷贝已经编译好的程序到安装目录

安装好之后到/usr/local/nginx/sbin (默认安装路径)即可开启程序

如果到安装路径执行nginx程序报以下错误，是用户或组没有创建，可以用useradd nginx解决

/usr/local/nginx/sbin/nginx -V 查看安装信息，比如版本，安装时添加的参数等

nginx安装好之后，安装路径会产生4个目录

conf 配置文件

html 默认网页根路径

logs 日志 ：/usr/local/nginx/logs nginx日志存放地，access.log是访问日志 error.log是报错日志

sbin 主程序

Sbin/nginx -s reload 重新加载配置文件

Sbin/nginx 加载配置文件

Sbin/ngixn -s stop 关闭程序

**用户认证**

--访问Web页面需要进行用户认证

--用户名为：tom，密码为：12345

--通过Nginx实现Web页面的认证，需要修改Nginx配置文件，在配置文件中添加auth语句实现用户认证。最后使用htpasswd命令创建用户及密码即可。

为网站添加用户认证功能：

/usr/local/nginx/conf/nginx.conf 主配置文件

vim +38 nginx.conf 打开到主配置文件的第38行，添加下面两行配置

auth\_basic "input password"; 提示信息，双引号中随意输入

auth\_basic\_user\_file "/usr/local/nginx/pass"; 认证的用户信息存放文件，配置语句不要忘记分号结尾

/usr/local/nginx/sbin/nginx 开启服务

/usr/local/nginx/sbin/nginx -s reload 执行主程序，从新读取配置文件

yum -y install httpd-tools 安装生成网站账户名与密码的工具包

htpasswd -c /usr/local/nginx/pass tom 利用htpasswd工具创建认证文件，以及账户名，连续两次密码

htpasswd /usr/local/nginx/pass alice 追加账户，不要-c选项

**基于域名的虚拟主机**

--修改Nginx配置文件，添加server容器实现虚拟主机功能；

对于需要进行用户认证的虚拟主机添加auth认证语句。

--虚拟主机一般可用分为：基于域名、基于IP和基于端口的虚拟主机

配置虚拟主机：

1，修改默认配置的域名，将默认的localhost修改为

server\_name www.a.com; 大概在37行

2，再添加新虚拟主机，在第34行添加以下内容

server {

listen 80; #监听的端口

server\_name [www.b.com](http://www.a.com/) ; #域名

root "/usr/local/nginx/www" ;#server\_name:网页文件的路径

index index.html index.htm ; # 在网页文件中的格式 按顺序匹配

}

保存退出之后，从新读配置文件，或者从起服务，无报错即可

在 /etc/hosts 添加域名解析 即可访问

---------------------------------------------------------------------------

**ssl虚拟主机（htpps加密网站）**

--域名为www.c.com

--该站点通过https访问

--通过私钥、证书对该站点所有数据加密

--源码安装Nginx时必须使用--with-http\_ssl\_module参数，启用加密模块，对于需要进行SSL加密处理的站点添加ssl相关指令（设置网站需要的私钥和证书）。

--加密算法一般分为对称算法、非对称算法、信息摘要。

--对称算法有：AES、DES，主要应用在单机数据加密。

--非对称算法有：RSA、DSA，主要应用在网络数据加密。

--信息摘要：MD5、sha256，主要应用在数据完整性校验。

安装nginx安全模块--with-http\_ssl\_nodule

cd /usr/local/nginx/conf #一定在conf的目录下

cp nginx.conf.default nginx.conf 恢复默认配置文件

vim nginx.conf 编辑主配置文件

将98~115行的#去掉

:98,115s/#//

openssl genrsa > cert.key 生成私钥

openssl req -new -x509 -key cert.key > cert.pem 根据私钥生成证书

/usr/local/nginx/sbin/nginx -s reload 重加载配置文件,如果nginx服务没启动，则执行/usr/local/nginx/sbin/nginx

在火狐浏览器打开 https://192.168.4.5 或者 https://192.168.2.5

**部署lnmp环境**

1，编译安装nginx

2，安装数据库，mysql（mariadb）

yum -y install mariadb 安装数据库客户端

yum -y install mariadb-server 安装数据库服务端

yum -y install mariadb-devel 安装数据库依赖包

3，安装php工具

yum -y install php php-mysql

yum -y install php-fpm

**php-fpm配置文件**

1）查看php-fpm配置文件

[root@proxy etc]# vim /etc/php-fpm.d/www.conf

[www]

listen = 127.0.0.1:9000             //PHP端口号

pm.max\_children = 32                //最大进程数量

pm.start\_servers = 15                //最小进程数量

开启服务

1，开启nginx

2，开启数据库

systemctl start mariadb

systemctl enable mariadb

netstat -ntulp | grep :3306 查看服务状态

3，开启php-fpm

systemctl start php-fpm

systemctl enable php-fpm

netstat -ntulp | grep :9000 查看服务状态

----------------------------------------------------------------

测试文件：

cd /root/lnmp\_soft/php\_scripts

cp test.php /usr/local/nginx/html/ 拷贝php脚本文件到网站根目录

fastcgi 公共网关接口，可以让网站服务与脚本解释器沟通的一个标准

----------------------------------------------------------------

**修改nginx配置文件，开启脚本解析功能（php-fpm）**

**（注释：在开启http机密往网站后在开启 lnmp动态网站需要把这个配置复制到 https的 的虚拟主机里面）**

location ~ \.php$ { ~ 开启正则匹配

root html; #网页文件的根目录

fastcgi\_pass 127.0.0.1:9000; #由nginx转交给php-fpm

fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name; 无效配置，需要注释或删除

include fastcgi.conf; #修改配置文件名为.conf

}

/usr/local/nginx/sbin/nginx #开服务

/usr/local/nginx/sbin/nginx -s reload #如果服务已经开启，就重置配置文件

killall nginx 如果nginx #异常可以尝试杀所有nginx程序，再从新开启服务

用浏览器访问 192.168.2.5/test.php

----------------------------------------------------------------

**配置 php-fpm服务**

vim /etc/php-fpm.d/www.conf

listen = 127.0.0.1:9000 #PHP端口号

pm.max\_children = 50 #最大进程数量

pm.start\_servers = 5 # 开启服务时的子进程数量，无论当前是否需要

pstree -a # 使用进程树查看php-fpm开启情况

排错：防火墙、9000端口、nginx.conf配置文件的修改、数据库

----------------------------------------------------------------

**地址重写**

可以实现访问a.html页面而看到b.html页面的内容

或者访问网站www.360buy.com 而看到www.jd.com网站的内容

语法结构

rewrite regex replacement flag

rewrite 旧地址（支持正则） 新地址 [选项]

cd /usr/local/nginx/html/

echo aaa > a.html

echo bbb > b.html

vim /usr/local/nginx/conf/nginx.conf

rewrite a.html /b.html; 在大概38行的位置添加，当客户访问的网址中带a.html，则跳转到b.html

rewrite a.html$ /b.html; 改进，更严谨一些

./nginx -s reload 重置配置文件

火狐打开192.168.2.5/a.html 则会跳转到b.html页面，地址栏不变

-----------------------------------------

rewrite a.html /b.html redirect; 再将之前的地址重写追加redirect选项

./nginx -s reload 重置配置文件

火狐再打开192.168.2.5/a.html 则会跳转到b.html页面，地址栏发生变化

---------------------------------------

不同网址（域名），相同页面的跳转

rewrite /(.\*) http://www.tmooc.cn/$1 ; 访问本机域名下任何页面，都会跳转到www.tmooc.cn下 相同页面，（）是正则表达式保留的效果，$1是粘贴效果

地址重写

---------------------------------------

**实现不同浏览器访问不同页面的跳转，为火狐准备一套专用页面（html/firefox目录下的），但如果用户访问不是用的火狐，则显示另一套页面（html目录下的）**

cd /usr/local/nginx/html

mkdir firefox 制作火狐浏览器专用页面的目录

echo firefox\_test > firefox/index.html 创建默认页面

echo other\_test > a.html 再制作根目录的a页面

echo firefox\_test\_a > firefox/a.html 再制作火狐专用a页面

-------------------------------------------------------

打开nginx主配置文件，在第47行添加

if ($http\_user\_agent ~\* firefox) { 如果用户的浏览器是火狐$http\_user\_agent 是系统变量匹配访问日志中的关键字

rewrite ^(.\*)$ /firefox/$1; 则将页面跳转到专用目录下的页面

rewrite index.html /firefox/index.html; 默认页面的跳转

}

-------------------------------------------------------

charset utf-8; 在主配置文件第40行添加支持中文的配置

-------------------------------------------------------

~ 支持正则 \* 不区分大小写

使用火狐浏览器访问192.168.2.5

使用火狐浏览器访问192.168.2.5/a.html

再使用curl 192.168.2.5

再使用curl 192.168.2.5/a.html

-------------------------------------------------------

在主配置文件第42行添加地址重写配置

rewrite a.html /b.html ;

rewrite b.html /c.html ;

重启或者重新加载服务

在/usr/local/nginx/html目录中创建测试文件

echo aaa > a.html

echo bbb > b.html

echo ccc > c.html

-----------------------------------------------------

rewrite a.html /b.html last ; 添加last选项，则不会跳转到c页面

rewrite b.html /c.html ;

-----------------------

rewrite a.html /b.html break ; 或替换为break，可以实现相同效果

rewrite b.html /c.html ;

-----------------------------------------------------

再次修改，删除之前的rewrite，修改为下面的状态

location / {

rewrite a.html /b.html;

root html;

index index.html index.htm;

}

location /b.html {

rewrite b.html /c.html;

}

重置服务之后，测试效果与之前rewrite不加选项时一样

-----------------------------------------------------

location / {

rewrite a.html /b.html break; 加了last只会停止本location内部的地址跳转，加了 break之后 可以停止另外的location的地址跳转

root html;

index index.html index.htm;

}

location /b.html {

rewrite b.html /c.html;

}

-----------------------------------------------------

location / {

rewrite a.html /b.html redirect; 添加重定向

root html;

index index.html index.htm;

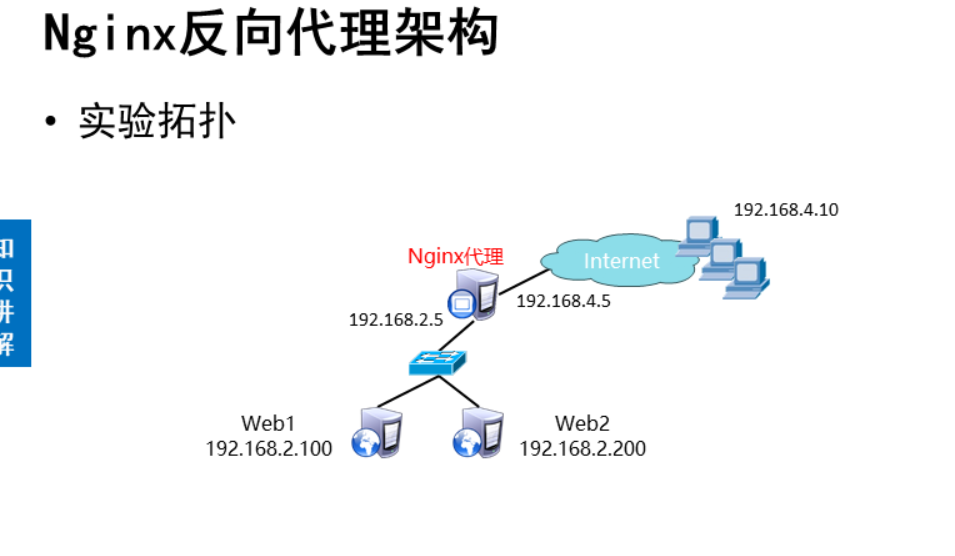
｝

**组建反向代理环境**

**虚拟机1：**

**ip 192.168.2.5 192.168.4.5**

**主机名 proxy 还原nginx服务到默认状态**





**nginx web 代理（调度）**

打开proxy主机的nginx的主配置文件，在34行添加集群

upstream webserver { 定义服务器集群，webserver是名字，可以自定义

server 192.168.2.100:80; 集群的成员

server 192.168.2.200:80;

}

location / {

proxy\_pass http://webserver; 在已有的 location中添加代理的集群地址

root html;

index index.html index.htm;

}

**nginx 具备健康检查功能，可以随时保证客户访问的是可用的服务器**

upstream webserver {

ip\_hash; 相同客户端访问相同服务器，可以避免客户反复验证身份的情况出现

server 192.168.2.100:80 weight=1 max\_fails=2 fail\_timeout=20; weight

是权重，默认值为1，性能较强的服务器可以适当增加该值，max\_fails=2与fail\_timeout=20是2 次无法连接到对应的主机，则等20秒再连接

server 192.168.2.200:80 weight=1 max\_fails=2 fail\_timeout=20;

}

测试，将web1的httpd服务关闭，访问192.168.2.5（或4.5），发现已经不再轮询web1，然后打开web1的httpd服务需要等待20秒才参与轮询。

**配置nginx实现四层代理**

**进入到nginx源码文件所在的目录 重新**

**./configure --with-stream --with-http\_stub\_status\_module 重新编译**

备份老版本nginx，更新新版本nginx

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

[root@web1 nginx-1.15.8]# mv /usr/local/nginx/sbin/nginx{,.old}

[root@web1 nginx-1.15.8]# cp objs/nginx /usr/local/nginx/sbin/

升级或重启服务

注意：必须在nginx-1.15.8源码包目录下执行make upgrade命令。

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

[root@web1 nginx-1.15.8]# make upgrade

**./configure --with-stream --with-http\_stub\_status\_module**

**make**

**make install**

打开nginx的主配置文件，添加下面内容在整个nginx server外面添加

stream { 配置四层代理功能

upstream backend { 创建集群，名称为backend

server 192.168.2.100:22; 添加主机以及对应的端口

server 192.168.2.200:22;

}

server { 调用集群

listen 12345; 端口号，为了不与本机冲突，不要使用22号端口

proxy\_pass backend; 当有人访问代理服务器的12345端口时，去集群中轮询调用主机

}

}

修改完配置文件，重置服务

测试：

如果是linux：

ssh 192.168.4.5 -p 12345

rm -rf ~/.ssh/known\_hosts

如果是xshell

ssh 192.168.4.5 12345

killall nginx

**开启服务器状态信息功能，打开主配置文件，在默认的location下面新添加一个**

**在添加--with-http\_stub\_status\_module 模块**

**location，并写入如下内容：**

**location /status {**

**stub\_status on;**

**Allow 192.168.2.0/24; //允许2.0网段的访问 可以不写**

**Deny all; //拒绝其他人**

**}**

**保存重置服务之后，使用浏览器查看192.168.2.5/status**

1. [root@proxy ~]# curl http://192.168.4.5/status
2. Active connections: 1
3. server accepts handled requests
4. 10 10 3
5. Reading: 0 Writing: 1 Waiting: 0

Active connections：当前活动的连接数量。

Accepts：已经接受客户端的连接总数量。

Handled：已经处理客户端的连接总数量。

（一般与accepts一致，除非服务器限制了连接数量）。

Requests：客户端发送的请求数量。

Reading：当前服务器正在读取客户端请求头的数量。

Writing：当前服务器正在写响应信息的数量。

Waiting：当前多少客户端在等待服务器的响应。

**Nginx的高级优化**

**Nginx的版本升级 添加模块**

**升级nginx版本，开启status模块**

1. [root@web1 ~]# tar -xf nginx-1.15.8.tar.gz
2. [root@web1 ~]# cd nginx-1.15.8
3. [root@web1 nginx-1.15.8]# ./configure \
4. --with-http\_ssl\_module \
5. --with-http\_stub\_status\_module
6. [root@web1 nginx-1.15.8]# make

[root@web1 ~]# tar -xf nginx-1.15.8.tar.gz

[root@web1 ~]# cd nginx-1.15.8

[root@web1 nginx-1.15.8]# ./configure \

--with-http\_ssl\_module \

--with-http\_stub\_status\_module

[root@web1 nginx-1.15.8]# make

2) 备份老版本nginx，更新新版本nginx

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

1. [root@web1 nginx-1.15.8]# mv /usr/local/nginx/sbin/nginx{,.old}
2. [root@web1 nginx-1.15.8]# cp objs/nginx /usr/local/nginx/sbin/

3）修改配置文件

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

1. [root@web1 ~]# vim /usr/local/nginx/conf/nginx.conf
2. ... ...
3. location /status {
4. stub\_status on;
5. allow 192.168.2.0/24;
6. deny all;
7. }
8. ... ...

[root@web1 ~]# vim /usr/local/nginx/conf/nginx.conf

... ...

location /status {

stub\_status on;

allow 192.168.2.0/24;

deny all;

}

... ...

4) 升级或重启服务

注意：必须在nginx-1.15.8源码包目录下执行make upgrade命令。

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

1. [root@web1 nginx-1.15.8]# make upgrade

[root@web1 nginx-1.15.8]# make upgrade

或者手动执行killall命令杀死进程后重新启动

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

1. [root@web1 ~]# killall nginx
2. [root@web1 ~]# /usr/local/nginx/sbin/nginx

## **Nginx常见问题处理**

## **自定义Nginx404报错页面**

**自定义报错页面**

1）优化前，客户端使用浏览器访问不存在的页面，会提示404文件未找到

修改Nginx配置文件，自定义报错页面

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY03/CASE/01/index.html)

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. .. ..
3. charset utf-8; //仅在需要中文时修改该选项
4. error\_page 404 /404.html;    //自定义错误页面
5. .. ..
6. [root@proxy ~]# vim /usr/local/nginx/html/404.html        //生成错误页面
7. Oops,No NO no page …
9. [root@proxy ~]# nginx -s reload
10. #请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：
11. #[error] open() "/usr/local/nginx/logs/nginx.pid" failed (2: No such file or directory）

**常见http状态码**

常见http状态码可用参考表-1所示。



**如何查看服务器状态信息（非常重要的功能）**

启用Nginx服务并查看监听端口状态

ss命令可以查看系统中启动的端口信息，该命令常用选项如下：

-a显示所有端口的信息

-n以数字格式显示端口号

-t显示TCP连接的端口

-u显示UDP连接的端口

-l显示服务正在监听的端口信息，如httpd启动后，会一直监听80端口

-p显示监听端口的服务名称是什么（也就是程序名称）

注意：在RHEL7系统中可以使用ss命令替代netstat命令，功能一样，选项一样。

**优化Nginx并发量**

1）优化前使用ab高并发测试

1. [root@proxy ~]# ab -n 2000 -c 2000 <http://192.168.4.5/> -n是多少人 -c 访问的次数
2. Benchmarking 192.168.4.5 (be patient)
3. socket: Too many open **files** (24)                //提示打开文件数量过多

2）修改Nginx配置文件，增加并发量

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY03/CASE/01/index.html)

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. .. ..
3. worker\_processes 2;                    //与CPU核心数量一致 在配置文件的最前面
4. events {
5. worker\_connections 65535;        //每个worker最大并发连接数
6. }
7. .. ..
8. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

**优化Linux内核参数（最大文件数量**

优化Linux内核参数（最大文件数量）

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY03/CASE/01/index.html)

1. [root@proxy ~]# ulimit -a                        *//查看所有属性值*
2. [root@proxy ~]# ulimit -Hn 100000                *//设置硬限制（临时规则）*
3. [root@proxy ~]# ulimit -Sn 100000                *//设置软限制（临时规则）*
4. [root@proxy ~]# vim /etc/security/limits.conf
5. .. ..
6. \* soft nofile 100000
7. \* hard nofile 100000
9. #该配置文件分4列，分别如下：
10. #用户或组 硬限制或软限制 需要限制的项目 限制的值

**优化Nginx数据包头缓存**

**优化地址栏过长导致414错误的配置：**

修改Nginx配置文件，增加数据包头部缓存大小

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY03/CASE/01/index.html)

1. **[**root@proxy **~]**# vim **/**usr**/**local**/**nginx**/**conf**/**nginx**.**conf
2. **..** **..**
3. http **{**
4. client\_header\_buffer\_size 1k**;**        //默认请求包头信息的缓存
5. large\_client\_header\_buffers 4 4k**;**        //大请求包头部信息的缓存个数与容量
6. **..** **..**
7. **}**
8. **[**root@proxy **~]**# /usr/local**/**nginx**/**sbin**/**nginx **-**s reload

**浏览器本地缓存静态数据**

**优化缓存大文件的时间，使客户机存储服务器数据的时间增加，**

**达到缓解服务器压力的目的**

修改Nginx配置文件，定义对静态页面的缓存时间

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY03/CASE/01/index.html)

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. server {
3. listen 80;
4. server\_name localhost;
5. location / {
6. root html;
7. index index.html index.htm;
8. }
9. location ~\* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ { //定义在html网页目录中以什么结尾的文件所为缓存

expires        30d;            //定义客户端缓存时间为30天

1. }
2. }
3. [root@proxy ~]# cp /usr/share/backgrounds/day.jpg /usr/local/nginx/html
4. [root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload
5. #请先确保nginx是启动状态，否则运行该命令会报错,报错信息如下：
6. #[error] **open**() "/usr/local/nginx/logs/nginx.pid" **failed** (2: No such file or directory)

**编写日志切割脚本**

编写脚本（以web1为例） 解决时间久了日志越来越大的问题

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

1. [root@web1 ~]# vim /usr/local/nginx/logbak.sh
2. #!/bin/bash
3. date=`date +%Y%m%d`
4. logpath=/usr/local/nginx/logs
5. mv $logpath/access.log $logpath/access-$date.log
6. mv $logpath/error.log $logpath/error-$date.log
7. kill -USR1 $(cat $logpath/nginx.pid) //kill 不只是杀死进程 可以kill -l 查看kill功能 不影响 正在访问的客户生成新的日志进程id 从而生成新的日志 目录 在 nginx.Pid 存 放nginx日志文件的进程号

[root@web1 ~]# vim /usr/local/nginx/logbak.sh

#!/bin/bash

date=`date +%Y%m%d`

logpath=/usr/local/nginx/logs

mv $logpath/access.log $logpath/access-$date.log

mv $logpath/error.log $logpath/error-$date.log

kill -USR1 $(cat $logpath/nginx.pid)

2）创建计划任务

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

1. [root@web1 ~]# crontab -e
2. 03 03 \* \* 5 /usr/local/nginx/logbak.sh

**对页面进行压缩处理**

1）修改Nginx配置文件

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

1. [root@proxy ~]# cat /usr/local/nginx/conf/nginx.conf
2. http {
3. .. ..
4. gzip on;                            //开启压缩
5. gzip\_min\_length 1000;                //小文件不压缩
6. gzip\_comp\_level 4;                //压缩比率 1~9的数值 越大压缩效率越好 但占用内存资源
7. gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;
8. //对特定文件压缩，类型参考nginx/mime.types
9. .. ..
10. }

**服务器内存缓存**

如果需要处理大量静态文件，可以将文件缓存在内存，下次访问会更快。

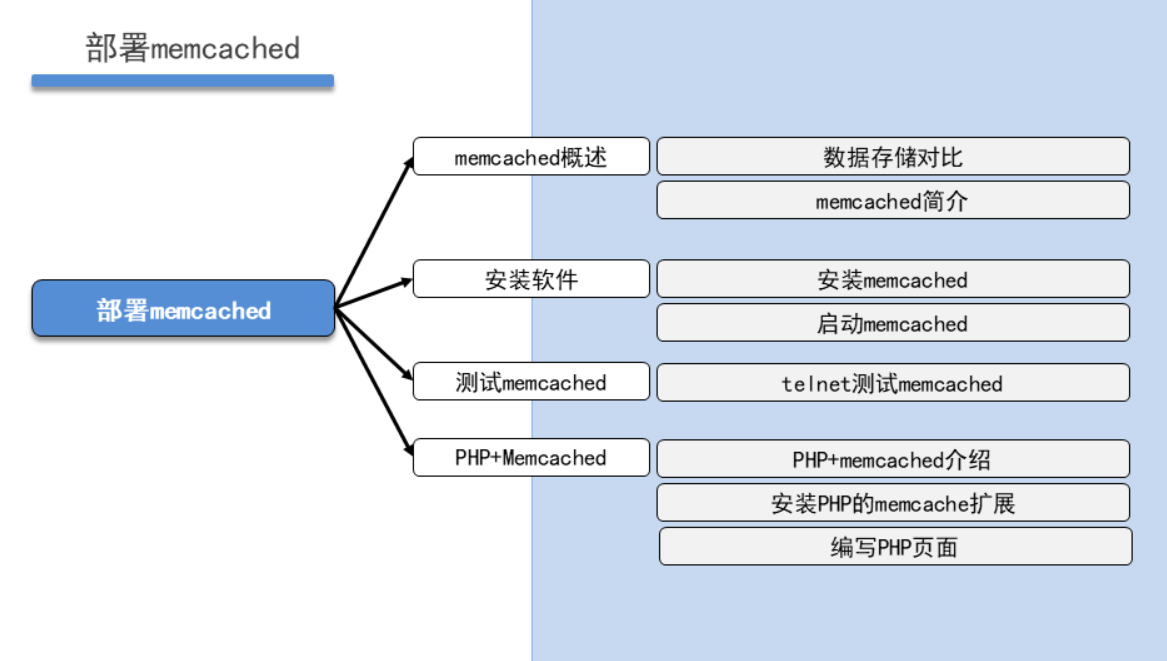
[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/PROJECT1/DAY04/CASE/01/index.html)

1. http {
2. open\_file\_cache max=2000 inactive=20s;
3. open\_file\_cache\_valid 60s;
4. open\_file\_cache\_min\_uses 5;
5. open\_file\_cache\_errors off;
6. //设置服务器最大缓存2000个文件句柄，关闭20秒内无请求的文件句柄
7. //文件句柄的有效时间是60秒，60秒后过期 无论60s后有没有人访问自动更新数据
8. //只有访问次数超过5次会被缓存
9. }

**Session：存储在服务器端，保存用户名、登陆状态等信息。**

**Cookies：由服务器下发给客户端，保存在客户端的一个文件里。**

**默认用户登录后会在 /var/lib/php/session产生用户登录信息**



**本地session**

**用户登录后在web服务器主机查看**



**Memcached简介**

Memcached 数据储存对比

--性能

- cpu缓存>内存>磁盘>数据库

--价格

- cpu缓存>内存>磁盘>数据库

- memcached是高性能的分布式缓存服务器

-用来集中缓存数据库查询结果 减少数据库访问次数 以提高动态web应用的响应速度

- 官方网站：

在proxy(调度服务器)主机安装memcache （是一个利用

yum -y install memcached telnet 装2个工具

vim /etc/sysconfig/memcached 打开memcache的配置文件

Port 默认端口号 11211

user 默认用户 memcached

maxconn 连接用户数量

cachesize 缓存大小 默认64m

telnet 192.168.4.5 11211 登录服务

登录之后，控制命令：

add name 0 180 10 添加变量，名称是name，0是不压缩，180是缓存时间（秒） 10是要存10个字符

set name 0 180 10 set是设置（定义或创建的意思）

replace name 0 180 10 替换，必须存在被替换的变量，否则失败

get name 读取变量

delete name 删除变量

flush\_all 清空所有

add abc 0 100 3 如果变量名已经存在，则不能在同名变量中存储

123

set abc 0 100 3 如果变量名已经存在，可以覆盖存储

456

quit 退出

**构建memcached服务**

* 安装memcached软件，并启动服务
* 使用telnet测试memcached服务
* 对memcached进行增、删、改、查等操作

**构建memcached服务**

1）使用yum安装软件包memcached

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY04/CASE/01/index.html)

1. **[**root@proxy **~]**# yum **-**y install memcached telnet
2. **[**root@proxy **~]**# rpm **-**qa memcached
3. memcached-1.4.15-10**.**el7\_3.1**.**x86\_64

memcached配置文件（查看即可，不需要修改）

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY04/CASE/01/index.html)

1. [root@proxy ~]# vim /usr/lib/systemd/system/memcached.service
2. ExecStart=/usr/bin/memcached -u $USER -p $PORT -m $CACHESIZE -c $MAXCONN $OPTIONS
4. [root@proxy ~]# vim /etc/sysconfig/memcached
5. PORT="11211"
6. USER="memcached"
7. MAXCONN="1024"
8. CACHESIZE="64"
9. OPTIONS=""

**使用telnet访问memcached服务器**

1)使用telnet连接服务器测试memcached服务器功能，包括增、删、改、查等操作。

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY04/CASE/01/index.html)

1. [root@proxy ~]# telnet 192.168.4.5 11211
2. Trying 192.168.4.5...
3. ……
4. ##提示：0表示不压缩，180为数据缓存时间，3为需要存储的数据字节数量。
5. set name 0 180 3                //定义变量，变量名称为name
6. plj                            //输入变量的值，值为plj
7. STORED
8. get name                        //获取变量的值
9. VALUE name 0 3             //输出结果
10. plj
11. END
12. ##提示：0表示不压缩，180为数据缓存时间，3为需要存储的数据字节数量。
13. add myname 0 180 10            //新建，myname不存在则添加，存在则报错
14. set myname 0 180 10            //添加或替换变量
15. replace myname 0 180 10        //替换，如果myname不存在则报错
16. get myname                    //读取变量
17. delete myname                    //删除变量
18. flush\_all                        //清空所有
19. quit                            //退出登录

## **PHP实现session共享**

* 配置PHP使用memcached服务器共享Session信息
* 客户端访问两台不同的后端Web服务器时，Session 信息一致

Nginx服务器除了承担调度器外，还需要担任memcached数据库的角色，

并在两台后端LNMP服务器上实现PHP的session会话共享

**为Web服务器安装PHP扩展**

1）为web1主机的PHP添加memcache扩展

1. **[**root@web1 **~]**# yum **-**y install php**-**pecl**-**memcache
2. **[**root@web1 **~]**# systemctl restart php**-**fpm

2）为web2主机的PHP添加memcache扩展

1. [root@web2 ~]# yum -y install php-pecl-memcache
2. [root@web2 ~]# systemctl restart php-fpm

**在后端LNMP服务器上部署Session共享**

**在nginx调度服务器上构建memcached服务**

1）使用yum安装软件包memcached

使用yum安装软件包memcached

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY04/CASE/01/index.html)

1. [root@proxy ~]# yum -y install memcached telnet
2. [root@proxy ~]# rpm -qa memcached
3. memcached-1.4.15-10.el7\_3.1.x86\_64

**启动服务并查看**

1. [root@proxy ~]# systemctl start memcached
2. [root@proxy ~]# systemctl status memcached
3. [root@proxy ~]# ss -anptu | grep Memcached

**这些操作在两台后端Web服务器上均需要执行，以下操作以Web1（192.168.2.100）服务器为例**

**为Web服务器安装PHP扩展**

1）为web1主机的PHP添加memcache扩展

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY04/CASE/01/index.html)

1. [root@web1 ~]# yum -y install php-pecl-memcache
2. [root@web1 ~]# systemctl restart php-fpm

**在后端LNMP服务器上部署Session共享**

**安装memcached**

修改PHP-FPM配置文件，并重启服务

注意，因为后端两台web服务器(web1,web2)都需要修改配置文件(下面也web1为例)。

[copytextpop-up](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN202001/OPERATION/DAY04/CASE/01/index.html)

1. [root@web1 ~]# vim /etc/php-fpm.d/www.conf            *//修改该配置文件的两个参数*
2. *//文件的最后2行*
3. 修改前效果如下:
4. php\_value[session.save\_handler] = files
5. php\_value[session.save\_path] = /var/lib/php/session
6. *//原始文件，默认定义Sessoin会话信息本地计算机（默认在/var/lib/php/session）*
7. +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
8. 修改后效果如下:
9. php\_value[session.save\_handler] = memcache
10. php\_value[session.save\_path] = "tcp://192.168.2.5:11211" //memcached服务器所在的IP地址
11. *//定义Session信息存储在公共的memcached服务器上，主机参数中为memcache（没有d）*
12. *//通过path参数定义公共的memcached服务器在哪（服务器的IP和端口）*
13. [root@web1 ~]# systemctl restart php-fpm