# 随记

Virsh 命令

1,查看所有虚拟机

Virsh list --all

2,链接虚拟机

Virsh console 域名

3,退出虚拟机

ctrl+]

4,关闭虚拟机

Virsh shutdown 域名

5,只是在virsh list中删除了该虚拟机。

virsh destroy +域名

6,挂起,恢复

virsh suspend +域名(resume)

7,子机随宿主主机启动而启动(取消自启)

Virsh autostart 域名 (autostart --disable)

8,彻底删除虚拟机

1> 删除虚拟机 virsh destroy +域名

2>解除标记 virsh undefine +域名

3>删除虚拟机文件

9,启动虚拟机并进入该虚拟机

virsh start 域名 --console

Operation7天

1. Web服务类型:网页-->F12-->网络-->消息头--->响应头--->server:Nginx
2. 权限继承,用户启动的程序则自动继承这个用户的权限身份,最小权限,所以创建一个用户来指定启动服务
3. <https://github.com/redhatedu/course/>
4. nginx中的stub\_status模块主要用于查看Nginx的一些状态信息. 本模块默认没有安装，需要编译安装。

在浏览器地址栏中输入 "域名/status" ,查看nginx上次



Active connections: 当前nginx正在处理的活动连接数.

Server accepts handled requests request\_time: nginx总共处理了2 个连接,成功创建2 握手(证明中间没有失败的),总共处理了16 个请求.一次连接可以实现多次请求

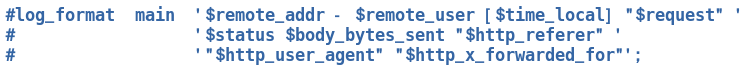
Reading: nginx读取到客户端的Header信息数.

Writing: nginx返回给客户端的Header信息数.

Waiting: 开启keep-alive的情况下,这个值等于 active – (reading + writing),意思就是nginx已经处理完成,正在等候下一次请求指令的驻留连接。

所以,在访问效率高,请求很快被处理完毕的情况下,Waiting数比较多是正常的.如果reading +writing数较多,则说明并发访问量非常大,正在处理过程中。

1. Apache和Nginx日志/usr/local/nginx/logs/access.log每一列



来源IP - 来源用户名tom(-) [访问时的时间] “request页面” “访问状态404不存在200正常” “ 访问产生的流量” “来源渠道微信” “$http\_user\_agent用户信息”

1. 数据存储位置

-性能:CPU缓存>内存>磁盘>数据库

-价格:CPU>内存>磁盘>数据库

7.systemctl 启动的文件在/usr/lib/systemd/system下

#########################################

**运维Day01**

Nginx安装与升级 Nginx配置 Nginx虚拟机 Https加密网站

**Nginx安装与升级**

Web服务器对比

-Apache 历史久,目前有些需求不能满足生产==>高并发量

-Nginx (engine x)

俄罗斯人开发 ,轻量级

高性能的HTTP和反向代理服务器, 处理高并发

* Nginx安装

Gcc:c语言解释器

Pcre: Nginx支持正则

1. 构建Nginx服务器

## 服务的模块化安装设计===>服务的各项功能(qq的:视频,截图,等功能)

1. 使用源码包安装nginx软件包

[root@proxy ~]# yum -y install gcc pcre-devel openssl-devel //安装依赖包

[root@proxy ~]# useradd -s /sbin/nologin nginx

[root@proxy ~]# tar -xf nginx-1.10.3.tar.gz

[root@proxy ~]# cd nginx-1.10.3

[root@proxy nginx-1.10.3]# ./configure \

> --prefix=/usr/local/nginx \ //指定安装路径

> --user=nginx \ //指定用户

> --group=nginx \ //指定组

> --with-http\_ssl\_module //开启SSL加密功能

[root@proxy nginx-1.10.3]# make && make install //编译并安装

1. nginx命令的用法

[root@proxy ~]# /usr/local/nginx/sbin/nginx

//启动服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s stop

//关闭服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

//重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -V

//查看软件信息

[root@proxy ~]# ln -s /usr/local/nginx/sbin/nginx /sbin/

//软链接方便后期使用

netstat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）

3）设置防火墙与SELinux（非必须的操作，如果有则关闭）

1. 测试首页文件
2. 平滑升级Nginx

将老版本mv改名,cp新的过来

Src里放的所有程序源码包

Objs里放的选择安装的模块

1）编译新版本nginx软件

[root@proxy ~]# tar -zxvf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2

[root@proxy nginx-1.12.2]# ./configure \

> --prefix=/usr/local/nginx \

> --user=nginx \

> --group=nginx \

> --with-http\_ssl\_module

[root@proxy nginx-1.12.2]# make

2) 备份老的nginx主程序，并使用编译好的新版本nginx替换老版本

[root@proxy nginx-1.12.2]# mv /usr/local/nginx/sbin/nginx

>/usr/local/nginx/sbin/nginxold

## [root@proxy nginx-1.12.2]# cp objs/nginx /usr/local/nginx/sbin/ //拷贝新版本

## [root@proxy nginx-1.12.2]# make upgrade //升级

## /usr/local/nginx/sbin/nginx -t

## nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok

## nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

## kill -USR2 `cat /usr/local/nginx/logs/nginx.pid`

## sleep 1

## test -f /usr/local/nginx/logs/nginx.pid.oldbin

## kill -QUIT `cat /usr/local/nginx/logs/nginx.pid.oldbin`

[root@proxy nginx-1.12.2]# killall Nginx

[root@proxy nginx-1.12.2]# Nginx

//make upgrade等于后面这两个过程

[root@proxy ~]# /usr/local/nginx/sbin/nginx –v //查看版本

1. 客户端访问测试

**Nginx配置解析**

通过调整Nginx服务端配置，实现：

访问Web页面需要进行用户认证

1. 修改Nginx配置文件：

1）修改/usr/local/nginx/conf/nginx.conf

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

.. ..

server {

listen 80;

server\_name localhost;

auth\_basic "Input Password:"; //认证提示符

auth\_basic\_user\_file "/usr/local/nginx/pass"; //认证密码文件

location / {

root html;

index index.html index.htm;

}

}

//文件每行末尾加分号;

1. 生成密码文件，创建用户及密码

使用htpasswd命令创建账户文件，需要确保系统中已经安装了httpd-tools

[root@proxy ~]# yum -y install httpd-tools

## [root@proxy ~]# htpasswd -c /usr/local/nginx/pass tom //创建密码文件

## New password:

## Re-type new password:

## Adding password for user tom

## [root@proxy ~]# htpasswd /usr/local/nginx/pass jerry //追加用户，不使用-c选项

## New password:

## Re-type new password:

## Adding password for user jerry

[root@proxy ~]# cat /usr/local/nginx/pass

1. 重启Nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload //重新加载配置文件

**Nginx虚拟主机**

三种模式虚拟主机:

-基于域名的虚拟主机

-基于端口的虚拟主机

-基于IP的虚拟主机

公司网站架构:

宣传门户

用户购物

论坛

邮箱

1. 修改配置文件

1）修改Nginx服务配置，添加相关虚拟主机配置如下

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

… …

server {

listen 80; //端口

server\_name www.b.com; //域名

charset utf-8; //指定字符集编码方式

location / {

root www; //指定网站根路径

index index.html index.htm;

}

}

2）创建网站根目录及对应首页文件

[root@proxy ~]# mkdir /usr/local/nginx/www

[root@proxy~]# echo "www" > /usr/local/nginx/www/index.html

1. 重启nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

1. 客户端测试

1）修改客户端主机192.168.4.10的/etc/hosts文件，进行域名解析

[root@client ~]# vim /etc/hosts

192.168.4.5 www.a.com [www.b.com](http://www.b.com)

1. 登录192.168.4.10客户端主机进行测试
2. 扩展：基于其他类型的虚拟主机
   1. 基于端口的

修改 listen 端口号

* 1. 基于IP的

修改listen ip:80

**SSL虚拟主机**

实现:

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：AES、DES，主要应用在单机数据加密。

-非对称算法：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。

//md5sum 文件名 :文件的md5码

1. 配置SSL虚拟主机
2. 生成私钥与证书

[root@proxy ~]# cd /usr/local/nginx/conf

[root@proxy ~]# openssl genrsa > cert.key

//生成私钥

[root@proxy ~]# openssl req -new -x509 -key cert.key > cert.pem

//生成证书, -x509 固定格式, 后面国家为CN

1. 修改Nginx配置文件，设置加密网站的虚拟主机

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

… …

server {

listen 443 ssl;

server\_name www.c.com;

ssl\_certificate cert.pem; #这里是证书文件

ssl\_certificate\_key cert.key; #这里是私钥文件

## ssl\_session\_cache shared:SSL:1m;

## ssl\_session\_timeout 5m;

## ssl\_ciphers HIGH:!aNULL:!MD5;

## ssl\_prefer\_server\_ciphers on;

location / {

root html;

index index.html index.htm;

}

}

1. 重启nginx服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

1. 客户端访问验证

########################################

**Day02**

部署LNMP Nginx+FastCGI Nginx高级技术

**LNMP概述**

-L : Linux操作系统

-N : Nginx网站服务

-M : MySQL Mariadb

-P :网站开发语言(PHP , Perl , Python)

PHP端口号9000

LNMP（Linux、Nginx、MySQL、PHP）

在RHEL7系统中，源码安装Nginx，使用RPM包安装MariaDB、PHP、PHP-FPM软件。

操作过程中需要安装的软件列表如下：

nginx

mariadb、mariadb-server、mariadb-devel

php、php-fpm、php-mysql

## 备注：mariadb（数据库客户端软件）、mariadb-server（数据库服务器软件）、mariadb-devel（其他客户端软件的依赖包）、php（解释器）、php-fpm（进程管理器服务）、php-mysql（PHP的数据库扩展包）。

1.安装软件

1）使用yum安装基础依赖包

[root@proxy ~]# yum -y install gcc openssl-devel pcre-devel

1. 源码安装Nginx（如果前面课程中已经安装Nginx，则忽略这一步）
2. 安装MariaDB

[root@proxy ~]# yum -y install mariadb mariadb-server mariadb-devel

1. php和php-fpm(该软件包在lnmp\_soft中提供)

[root@proxy ~]# yum -y install php php-mysql

[root@proxy lnmp\_soft]# yum -y install php-fpm-5.4.16-42.el7.x86\_64.rpm

1. 启动服务

1）启动Nginx服务（如果已经启动nginx，则可以忽略这一步）

注意: 如果启动了其他监听80端口的服务软件（如httpd），则需要先关闭该服务，否则会出现冲突。

[root@proxy ~]# systemctl stop httpd //如果该服务存在则关闭该服务

[root@proxy ~]# /usr/local/nginx/sbin/nginx //启动Nginx服务

[root@proxy ~]# netstat -utnlp | grep :80

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 32428/nginx

1. 启动MySQL服务

[root@proxy ~]# systemctl start mariadb //启动服务器

[root@proxy ~]# systemctl status mariadb //查看服务状态

[root@proxy ~]# systemctl enable mariadb //设置开机启动

1. 启动PHP-FPM服务

[root@proxy ~]# systemctl start php-fpm //启动服务

[root@proxy ~]# systemctl status php-fpm //查看服务状态

[root@proxy ~]# systemctl enable php-fpm //设置开机启动

1. 设置防火墙与SELinux

**构建LNMP平台**

通过修改Nginx及php-fpm配置文件实现对PHP页面的支持。

注意，FastCGI的内存消耗问题，一个PHP-FPM解释器将消耗约25M的内存。

多进程的程序

-程序:在硬盘上的

-进程:内存运行的,占用内存空间

-线程:进程下调用的线程,共享使用原进程的内存

1. php-fpm配置文件（实验中不需要修改该文件）

[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

pm.min\_spare\_servers = 5 //最少需要几个空闲着的进程

pm.max\_spare\_servers = 32 //最多允许几个进程处于空闲状态

1. 修改Nginx配置文件并启动服务

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

location / {

root html;

index index.php index.html index.htm;

## #设置默认首页为index.php，当用户在浏览器地址栏中只写域名或IP，不说访问什么页面时，服务器会把默认首页index.php返回给用户

}

location ~ \.php$ {

root html;

fastcgi\_pass 127.0.0.1:9000; #将请求转发给本机9000端口，PHP解释器

fastcgi\_index index.php;

## #fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;//注释掉

include fastcgi.conf;

}

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

1. 创建PHP页面，测试LNMP架构能否解析PHP页面
2. 创建PHP测试页面1，可以参考lnmp\_soft/php\_scripts/test.php：

[root@proxy ~]# vim /usr/local/nginx/html/test1.php

<?php

$i="This is a test Page";

echo $i;

?>

2）创建PHP测试页面,连接并查询MariaDB数据库,

可以参考lnmp\_soft/php\_scripts/mysql.php:

[root@proxy ~]# vim /usr/local/nginx/html/test2.php

<?php

$mysqli = new mysqli('localhost','root','密码','mysql');

## //注意：root为mysql账户名称，密码需要修改为实际mysql密码，无密码则留空即可

## //localhost是数据库的域名或IP，mysql是数据库的名称

if (mysqli\_connect\_errno()){

die('Unable to connect!'). mysqli\_connect\_error();

}

$sql = "select \* from user";

$result = $mysqli->query($sql);

while($row = $result->fetch\_array()){

printf("Host:%s",$row[0]);

printf("</br>");

printf("Name:%s",$row[1]);

printf("</br>");

}

?>

1. 客户端使用浏览器访问服务器PHP首页文档，检验是否成功：

[root@client ~]# firefox http://192.168.4.5/test1.php

[root@client ~]# firefox <http://192.168.4.5/test2.php>

1. LNMP常见问题

Nginx的默认访问日志文件为/usr/local/nginx/logs/access.log

Nginx的默认错误日志文件为/usr/local/nginx/logs/error.log

PHP默认错误日志文件为/var/log/php-fpm/www-error.log

如果动态网站访问失败，可用参考错误日志，查找错误信息。

**地址重写**

域名别名定义,改写成服务器可以处理的另一个URL的过程

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]

1.修改配置文件(访问a.html重定向到b.html)

1）修改Nginx服务配置：

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

.. ..

server {

listen 80;

server\_name localhost;

rewrite /a.html /b.html;

location / {

root html;

index index.html index.htm;

}

}

[root@proxy ~]# echo "BB" > /usr/local/nginx/html/b.html

1. 重新加载配置文件

[root@proxy ~]# /usr/local/nginx/sbin/nginx -s reload

1. 客户端测试

[root@client ~]# firefox <http://192.168.4.5/a.html>

1. 访问a.html重定向到b.html（跳转地址栏）
2. 修改Nginx服务配置：

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

.. ..

rewrite /a.html /b.html redirect;

.. ..

1. 修改配置文件(访问192.168.4.5的请求重定向至www.tmooc.cn)
2. 修改Nginx服务配置

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

.. ..

rewrite ^/ http://www.tmooc.cn/;

.. ..

2）重新加载配置文件

1. **修改配置文件**(访问192.168.4.5/下面子页面，重定向至www.tmooc.cn/下相同的页面)
2. 修改Nginx服务配置

rewrite ^/(.\*)$ <http://www.tmooc.cn/$1;>

## // 正则表达式()保留,后面$1再次调用,与Shell的正则不一样,Shell的为\1调用

1. 重新加载配置文件
2. 修改配置文件(实现curl和火狐访问相同链接返回的页面不同)
3. 创建网页目录以及对应的页面文件：

[root@proxy ~]# echo "I am Normal page" > /usr/local/nginx/html/test.html

[root@proxy ~]# mkdir -p /usr/local/nginx/html/firefox/

[root@proxy ~]# echo "firefox page" > /usr/local/nginx/html/firefox/test.html

1. 修改Nginx服务配置

#这里，~符号代表正则匹配，\*符号代表不区分大小写

if ($http\_user\_agent ~\* firefox) { //识别客户端firefox浏览器

rewrite ^(.\*)$ /firefox/$1;

}

## //在PHP中HTTP\_USER\_AGENT是用来获取用户的相关信息的，包括用户使用的浏览器，操作系统等信息

**地址重写格式【总结】**

rewrite 旧地址 新地址 [选项];

last 不再读其他rewrite

break 不再读其他语句，结束请求

redirect 临时重定向

permament 永久重定向

正则匹配模式:

- 区分大小写匹配 : ~

- 不区分大小写匹配 :~\*

- 区分大小写不匹配: !~

- 不区分大小写不匹配: !~\*

Rewrite /a.html /b.html last; //last 匹配即停止,不执行下一个rewrite

Rewrite /b.html /c.html;

########################################

**Day03**

Nginx调度器 Nginx优化

Nginx调度

-七层代理,七层调度(HTTP)(模型第七层)

-四层代理,四层调度(TCP/UDP)

**Nginx反向代理**

1. **配置Nginx服务器，添加服务器池，实现反向代理功能**

将web1和web2搭建 httpd提供页面

1）修改/usr/local/nginx/conf/nginx.conf配置文件

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

.. ..

http {

.. ..

## #使用upstream定义后端服务器集群，集群名称任意(如webserver)

## #使用server定义集群中的具体服务器和端口

upstream webserver {

server 192.168.2.100:80;

server 192.168.2.200:80;

}

.. ..

server {

listen 80;

server\_name localhost;

location / {

## #通过proxy\_pass将用户的请求转发给webserver集群

proxy\_pass <http://webserver;> //优先级最高,不会查找本地root路径

}

}

1. 重启nginx服务,访问测试
2. **配置upstream服务器集群池属性**
3. 设置失败次数，超时时间，权重

upstream webserver {

server 192.168.2.100 weight=1 max\_fails=1 fail\_timeout=30;

server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=30;

server 192.168.2.101 down;

}

#weight设置服务器权重值，默认值为1

#max\_fails设置最大失败次数

#fail\_timeout设置失败超时时间，单位为秒

#down标记服务器已关机，不参与集群调度

1. **配置upstream服务器集群的调度算法**
2. 设置相同客户端访问相同Web服务器

upstream webserver {

#通过ip\_hash设置调度规则为：相同客户端访问相同服务器

ip\_hash;

server 192.168.2.100 weight=1 max\_fails=2 fail\_timeout=10;

server 192.168.2.200 weight=2 max\_fails=2 fail\_timeout=10;

}

**Nginx的TCP/UDP调度器**

1. **部署支持4层TCP/UDP代理的Nginx服务器**

1）部署nginx服务器

编译安装必须要使用--with-stream参数开启4层代理模块。

[root@proxy nginx-1.12.2]# ./configure \

> --with-http\_ssl\_module //开启SSL加密功能

> --with-stream //开启4层反向代理功能

1. **配置Nginx服务器，添加服务器池，实现TCP/UDP反向代理功能**

1）修改/usr/local/nginx/conf/nginx.conf配置文件

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

stream {

upstream backend {

server 192.168.2.100:22; //后端SSH服务器的IP和端口

server 192.168.2.200:22;

}

server {

listen 12345; //Nginx监听的端口

proxy\_connect\_timeout 3s; //连接超时时间

proxy\_timeout 10s; //连接后断开时间

proxy\_pass backend;

}

}

http {.....}

**Nginx优化**

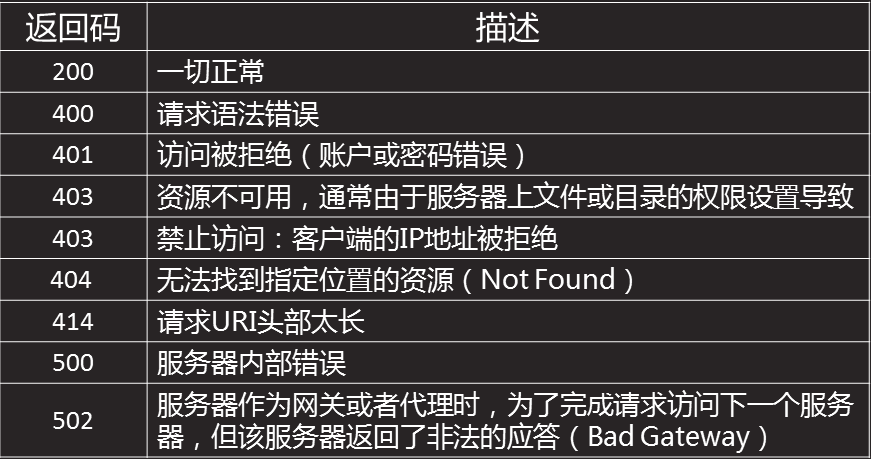
HTTP错误代码

-301永久重定向

-302临时重定向

-400类为用户错误

-500类为服务器错误



1. **自定义报错页面**
2. 修改Nginx配置文件，自定义报错页面

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

.. ..

error\_page 404 /404.html; //自定义错误页面

[root@proxy ~]# vim /usr/local/nginx/html/404.html //生成错误页面

1. 重启Nginx,访问测试

**2.如何查看服务器状态信息（非常重要的功能）**

1）编译安装时使用--with-http\_stub\_status\_module开启状态页面模块

[root@proxy ~]# yum -y install gcc pcre-devel openssl-devel //安装常见依赖包

[root@proxy ~]# tar -zxvf nginx-1.12.2.tar.gz

[root@proxy ~]# cd nginx-1.12.2

[root@proxy nginx-1.12.2]# ./configure \

> --with-http\_ssl\_module //开启SSL加密功能

> --with-stream //开启TCP/UDP代理模块

> --with-http\_stub\_status\_module //开启status状态页面

[root@proxy nginx-1.12.2]# make && make install //编译并安装

2）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。

[root@proxy ~]# /usr/local/nginx/sbin/nginx

[root@proxy ~]# netstat -anptu | grep nginx

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN 10441/nginx

[root@proxy ~]# ss -anptu | grep nginx

3) 修改Nginx配置文件，定义状态页面

location /status {

stub\_status on;

#allow IP地址;

#deny IP地址;

}

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(写返回给客户的数据包)。

Waiting：当前多少客户端在等待服务器的响应。

**3.优化Nginx并发量**

1）修改Nginx配置文件，增加并发量

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

.. ..

worker\_processes 2; //与CPU核心数量一致

events {

worker\_connections 65535; //每个worker最大并发连接数

## use epoll; //工作模式,默认

}

.. ..

[root@proxy ~]# nginx -s reload

1. 优化Linux内核参数（最大文件数量）

同一台机器不能同时打开1024个文件,同时运行3845个进程,需要修改上限.

[root@proxy ~]# ulimit -a //查看所有属性值

[root@proxy ~]# ulimit -Hn 100000 //设置硬限制（临时规则）

[root@proxy ~]# ulimit -Sn 100000 //设置软限制（临时规则）

[root@proxy ~]# vim /etc/security/limits.conf

.. ..

\* soft nofile 100000

\* hard nofile 100000

## #该配置文件分4列，分别如下：

## #用户或组 硬限制或软限制 需要限制的项目 限制的值

//重启计算机后再生效,与临时一起搭配使用

1. **优化Nginx数据包头缓存**

**服务器读取地址栏,包头数据较长**

1. 修改Nginx配置文件，增加数据包头部缓存大小

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

.. ..

http {

client\_header\_buffer\_size 1k; //默认请求包头信息的缓存

large\_client\_header\_buffers 4 4k; //大请求包头部信息的缓存个数与容量

.. ..

}

[root@proxy ~]# nginx -s reload

1. **浏览器本地缓存静态数据**

1）使用Firefox浏览器查看缓存

地址栏内输入about:cache将显示Firefox浏览器的缓存信息

1. 清空firefox本地缓存数据
2. 修改Nginx配置文件，定义对静态页面的缓存时间

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

server {

listen 80;

server\_name localhost;

location / {

root html;

index index.html index.htm;

}

location ~\* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ {

expires 30d; //定义客户端缓存时间为30天

}

}

[root@proxy ~]# cp /usr/share/backgrounds/day.jpg /usr/local/nginx/html

[root@proxy ~]# nginx -s reload

1. **日志切割**

Nginx访问日志越来越大,日志切割

1）手动执行

[root@proxy ~]# mv access.log access2.log

[root@proxy ~]# kill -USR1 $(cat /usr/local/nginx/logs/nginx.pid)

2）自动完成

每周5的03点03分自动执行脚本完成日志切割工作。

[root@proxy ~]# 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)

设置计划任务

[root@proxy ~]# crontab -e

03 03 \* \* 5 /usr/local/nginx/logbak.sh

//kill : 给程序发送一个指令,kill -l查看有60个指令

Kill -9sigkill 杀死

Kill -19sigstop 暂停程序

Kill -USR1

USR1:用来告知应用程序重载配置文件；例如，向Apache HTTP服务器发送一个USR1信号将导致以下步骤的发生：停止接受新的连接，等待当前连接停止，重新载入配置文件，重新打开日志文件，重启服务器，从而实现相对平滑的不关机的更改。

1. **对页面进行压缩处理**
2. 修改Nginx配置文件

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

http {

.. ..

gzip on; //开启压缩

gzip\_min\_length 1000; //小文件不压缩

gzip\_comp\_level 4; //压缩比率

gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;

//对特定文件压缩，多媒体类型不压缩,类型参考mime.types

.. ..

}

**8.服务器内存缓存**

1）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。

http {

open\_file\_cache max=2000 inactive=20s;

open\_file\_cache\_valid 60s;

open\_file\_cache\_min\_uses 5;

open\_file\_cache\_errors off;

//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

//文件句柄的有效时间是60秒，60秒后过期

//只有访问次数超过5次会被缓存

}

########################################

**Day04**

Session共享 memcached原理 部署memcached

**Session**

Session

-存储在服务端,保存用户名 , 登录状态等信息

Cookies

-由服务器下发给客户端,保存在客户端的一个文件里.保存的内容主要包括:SessionID , 账号名 , 过期时间 , 路径和域

通过Nginx调度器负载后端两台Web服务器，实现以下目标：

部署Nginx为前台调度服务器

调度算法设置为轮询

后端为两台LNMP服务器

部署测试页面，查看PHP本地的Session信息

1. **部署后端LNMP服务器相关软件(web1和web2)**
2. 使用yum安装基础依赖包gcc openssl-devel pcre-devel
3. 源码安装Nginx
4. 安装MariaDB数据库
5. 安装PHP（php-fpm软件包在lnmp\_soft中有提供）
6. 修改Nginx配置文件（修改默认首页与动静分离）
7. **启动LNMP服务器相关的服务**
8. 启动Nginx服务

关闭其他80端口,避免冲突

1. 启动MySQL服务
2. 启动PHP-FPM服务
3. 关闭SELinux、防火墙
4. **部署前端Nginx调度服务器**
5. 使用源码安装nginx软件
6. 修改Nginx配置文件

Nginx配置文件中，通过upstream定义后端服务器地址池，默认调度策略为轮询，使用proxy\_pass调用upstream定义的服务器地址池

1. 重新加载配置文件
2. **部署测试页面**
3. 部署测试页面(Web1服务器）

[root@web1 ~]# cd lnmp\_soft/php\_scripts/

[root@web1 php\_scripts]# tar -xf php-memcached-demo.tar.gz

[root@web1 php\_scripts]# cd php-memcached-demo

[root@web1 php-memcached-demo]# cp -a \* /usr/local/nginx/html/

1. 浏览器直接访问后端服务器的测试页面（Web1服务器）

[root@web1 ~]# firefox http://192.168.2.100 //填写账户信息

[root@web1 ~]# cd /var/lib/php/session/ //查看服务器本地的Session信息

[root@web1 ~]# ls

sess\_ahilcq9bguot0vqsjtd84k7244 //注意这里的ID是随机的

[root@web1 ~]# cat sess\_ahilcq9bguot0vqsjtd84k7244

1. 浏览器访问前端调度器测试（不同后端服务器Session不一致）

**Memcached缓存数据库**

-属于非关系型数据库

-保存在内存中,重启数据会丢失

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。

--->./memcached -p 11211 -l 192.168.170.237 -u root \

>-m 100 -c 200 -n 10 -f 2 -vvv -d

选项:

-p (指定memcached监听的端口号，默认11211)

-l (memcached服务器的IP地址)

-u (memcached程序运行时使用的用户身份，必须是root用户)

-m (指定使用本机的多少物理内存存数据，默认64M)

-c (memcached服务的最大链接数)

-vvv (显示详细信息)

-d (在后台启动)

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。

1. **构建memcached服务**
2. 安装Memcached服务

[root@proxy ~]# yum -y install memcached

2) memcached配置文件（查看即可，不需要修改）

[root@proxy~]#vim /usr/lib/systemd/system/memcached.service

ExecStart=/usr/bin/memcached -u $USER -p $PORT -m $CACHESIZE -c $MAXCONN $OPTIONS

## //绝对路径执行

[root@proxy ~]# vim /etc/sysconfig/memcached

PORT="11211"

USER="memcached"

MAXCONN="1024"

CACHESIZE="64"

OPTIONS=""

1. 启动服务并查看网络连接状态验证是否开启成功

[root@proxy ~]# systemctl start memcached

[root@proxy ~]# systemctl status memcached

1. 使用telnet访问memcached服务器

1）使用yum安装telnet

[root@proxy ~]# yum -y install telnet

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。

[root@proxy ~]# telnet 192.168.4.5 11211

Trying 192.168.4.5...

……

## ##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。

set name 0 180 3 //定义变量，变量名称为name

plj //输入变量的值，值为plj

STORED

get name //获取变量的值

VALUE name 0 3 //输出结果

plj

END

##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。

add myname 0 180 10 //新建，myname不存在则添加，存在则报错

set myname 0 180 10 //添加或替换变量

replace myname 0 180 10 //替换，如果myname不存在则报错

get myname //读取变量

append myname 0 180 10 //向变量中追加数据

delete myname //删除变量

stats //查看状态

flush\_all //清空所有

quit //退出登录

**LNMP+memcached**

1. **部署LNMP**
2. **创建PHP页面，使用PHP语言测试memcached服务**
3. 部署测试页面

[root@web1 ~]# vim /usr/local/nginx/html/test.php

<?php

$memcache=new Memcache; //创建memcache对象

$memcache->connect('192.168.2.5',11211) or die ('could not connect!!');

$memcache->set('key','test'); //定义变量

$get\_values=$memcache->get('key'); //获取变量值

echo $get\_values;

?>

1. 客户端测试（结果会失败）

[root@web1 ~]# firefox <http://192.168.2.100/test.php>

注意：这里因为没有给PHP安装扩展包，默认PHP无法连接memcached数据库，需要给PHP安装扩展模块才可以连接memcached数据库。

1. 为PHP添加memcache扩展

[root@web1 ~]# yum -y install php-pecl-memcache

[root@web1 ~]# systemctl restart php-fpm

4）客户端再次测试（结果会成功显示数据结果）

[root@web1 ~]# firefox <http://192.168.2.100/test.php>

**PHP实现session共享**

1. 在两台后端LNMP服务器上部署Session共享
2. 为PHP添加memcache扩展

[root@web1 ~]# yum -y install php-pecl-memcache

1. 修改PHP-FPM配置文件，并重启服务

[root@web1 ~]# vim /etc/php-fpm.d/www.conf //修改该配置文件的两个参数

//文件的最后2行

修改前效果如下:

php\_value[session.save\_handler] = files

php\_value[session.save\_path] = /var/lib/php/session

//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）

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

修改后效果如下:

php\_value[session.save\_handler] = memcache

php\_value[session.save\_path] = "tcp://192.168.2.5:11211"

//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）

//通过path参数定义公共的memcached服务器在哪（服务器的IP和端口）

[root@web1 ~]# systemctl restart php-fpm

1. 客户端测试

**Day05**

Tomcat服务器 tomcat应用案例 varnish代理服务器

**Tomcat服务器**

**Java简介**

-java是一种跨平台的 , 面向对象的程序设计语言 , java技术具有卓越的通用性 , 高效性 , 平台移植性和安全性

Java体系

-Java SE(标准版)

-Java EE(企业版)

-Java ME(移动版)

JDK简介

是针对JAVA语言的软件开发工具包

JDK是整个java的核心

-包括了java运行环境

-java工具(如编译,排错,打包等工具)

-java基础的类库

**1.安装部署Tomcat服务器**

1）使用RPM安装JDK环境

[root@web1 ~]# yum -y install java-1.8.0-openjdk //安装JDK

[root@web1 ~]# yum -y install java-1.8.0-openjdk-headless //安装JDK

[root@web1 ~]# java -version //查看JAVA版本

2）安装Tomcat（apache-tomcat-8.0.30.tar.gz软件包，在lnmp\_soft中有提供）

[root@web1 ~]# tar -xf apache-tomcat-8.0.30.tar.gz

[root@web1 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat

[root@web1 ~]# ls /usr/local/tomcat

bin/ //主程序目录

lib/ //库文件目录

logs/ //日志目录

temp/ //临时目录

work/ //自动编译目录jsp代码转换servlet

conf/ //配置文件目录

webapps/ //页面目录

1. 启动服务

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

[root@web1 ~]# firewall-cmd --set-default-zone=trusted

[root@web1 ~]# setenforce 0

1. 服务器验证端口信息

[root@web1 ~]# netstat -nutlp |grep java //查看java监听的端口

## 提示：如果检查端口时，8005端口启动非常慢，可用使用下面的命令用urandom替换random（非必须操作）。

[root@web1 ~]# mv /dev/random /dev/random.bak

[root@web1 ~]# ln -s /dev/urandom /dev/random

5）客户端浏览测试页面(proxy作为客户端)

[root@proxy ~]# firefox <http://192.168.2.100:8080>

1. **修改Tomcat配置文件**
2. 创建测试JSP页面

[root@web1 ~]# vim /usr/local/tomcat/webapps/ROOT/test.jsp

<html>

<body>

<center>

Now time is: <%=new java.util.Date()%> //显示服务器当前时间

</center>

</body>

</html>

1. 重启服务

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

**使用Tomcat部署虚拟主机**

修改server.xml配置文件，创建两个域名的虚拟主机，修改如下两个参数块：

# cat /usr/local/tomcat/conf/server.xml

<Server>

<Service>

<Connector port=8080 />

<Connector port=8009 />

<Engine name="Catalina" defaultHost="localhost">

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

</Host>

… …

1. **配置服务器虚拟主机**

1）修改server.xml配置文件，创建虚拟主机

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

… …

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

</Host>

1. 创建虚拟主机对应的页面根路径

网页存放在 /usr/local/tomcat/域名/ROOT 下

[root@web1 ~]# mkdir -p /usr/local/tomcat/{a,b}/ROOT

[root@web1 ~]# echo "AAA" > /usr/local/tomcat/a/ROOT/index.html

[root@web1 ~]# echo "BBB" > /usr/local/tomcat/b/ROOT/index.html

3）重启Tomcat服务器

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

1. **修改www.b.com网站的首页目录为base**
2. 使用docBase参数可以修改默认网站首页路径

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

.. ..

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

<Context path="" docBase="base"/>

</Host>

… …

[root@web1 ~]# mkdir /usr/local/tomcat/b/base

[root@web1 ~]# echo "BASE" > /usr/local/tomcat/b/base/index.html

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

2）测试查看页面是否正确(proxy充当客户端角色)

[root@proxy ~]# firefox http://www.b.com:8080/ //结果为base目录下的页面内容

1. **实现跳转网页**
2. 访问http://www.a.com/test打开/var/www/html目录下的页面

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

… …

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

<Context path="/test" docBase="/var/www/html/" />

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

<Context path="" docBase="base" />

</Host>

… …

[root@web1 ~]# echo "Test" > /var/www/html/index.html

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

1. 测试查看页面是否正确(proxy充当客户端角色)

[root@proxy ~]# firefox <http://www.a.com:8080/test>