

### 一 案例 1: PHP 的本地 Session 信息

#### 1.1 问题

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

#### 1.2 方案

概念:

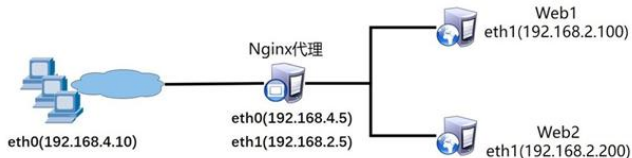
**Session:** 存储在服务器端, 保存用户名、登陆状态等信息。

**Cookies:** 保存文件名, 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。

实验拓扑环境:

使用 4 台 RHEL7 虚拟机, 其中一台作为 Nginx 前端调度器服务器 (eth0:192.168.4.5, eth1:192.168.2.5)、两台虚拟机部署为 LNMP 服务器, 分别为 Web1 服务器 (192.168.2.100) 和 Web2 服务器 (192.168.2.200), 另外一台作为测试用的 Linux 客户机 (192.168.4.10), 拓扑如图所示。



## 1.3 步骤

实现此案例需要按照如下步骤进行。

### 1.3.1 部署后端 LNMP 服务器相关软件

注意：以下部署 LNMP 服务器的操作，需要在两台后端服务器做相同的操作，下面我们以一台 Web2 服务器（192.168.2.200）为例，对 Web1 服务器执行相同操作即可。

#### 1) 使用 yum 安装基础依赖包

```
web2 ~]# yum -y install gcc openssl-devel pcre-devel
```

#### 2) 源码安装 Nginx

```
web2 ~]# tar -xf nginx-1.12.2.tar.gz
```

```
web2 ~]# cd nginx-1.12.2
```

```
web2 nginx-1.12.2]# ./configure \
```

```
> --with-http_ssl_module
```

```
web2 nginx-1.12.2]# make && make install
```

#### 3) 安装 MariaDB 数据库

```
web2 ~]# yum -y install mariadb mariadb-server mariadb-devel
```

#### 4) 安装 PHP

```
web2 ~]# yum -y install php php-mysql
```

```
web2 ~]# yum -y install php-fpm
```

### 5) 修改 Nginx 配置文件（修改默认首页与动静分离）

```
web2 ~]# vim /usr/local/nginx/conf/nginx.conf
```

```
location / {  
    root    html;  
    index  index.php index.html index.htm;  
}  
  
location ~ \.php$ {  
    root            html;  
    fastcgi_pass    127.0.0.1:9000;  
    fastcgi_index   index.php;  
    #fastcgi_param  $_script_name;  
    include         fastcgi.conf;  
}
```

## 1.3.2 启动 LNMP 服务器相关的服务

### 1) 启动 Nginx 服务

这里需要注意的是，如果服务器上已经启动了其他监听 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。

```
web2 ~]# systemctl stop httpd #如果该服务存在，则关闭该服务
```

```
web2 ~]# /usr/local/nginx/sbin/nginx #启动 nginx
```

```
web2 ~]# netstat -utnlp | grep :80
```

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

## 2) 启动 MySQL 服务

```
web2 ~]# systemctl start mariadb
```

```
web2 ~]# systemctl status mariadb
```

## 3) 启动 PHP-FPM 服务

```
web2 ~]# systemctl start php-fpm
```

```
web2 ~]# systemctl status php-fpm
```

### 1.3.3 部署前端 Nginx 调度服务器

#### 1) 使用源码安装 nginx 软件（如果 Nginx 软件包已安装可以忽略此步骤）

```
proxy ~]# yum -y install gcc pcre-devel openssl-devel
```

```
proxy ~]# tar -xf nginx-1.12.2.tar.gz
```

```
proxy ~]# cd nginx-1.12.2
```

```
proxy nginx-1.12.2]# ./configure
```

```
proxy nginx-1.12.2]# make && make install
```

#### 2) 修改 Nginx 配置文件，实现代理服务器

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：

```
proxy ~]# vim /usr/local/nginx/conf/nginx.conf
```

.. ..

```
upstream webs {  
  
    server 192.168.2.100:80;  
  
    server 192.168.2.200:80;  
  
}  
    #写在#gzip on;语句上  
  
server {  
  
    listen      80;  
  
    server_name localhost;  
  
    location / {  
  
        proxy_pass http://webs;  
  
        root    html;  
  
        index  index.php index.html index.htm;  
  
    }  
  
}
```

### 3) 重新加载配置文件

```
proxy ~]# /usr/local/nginx/sbin/nginx -s reload
```

#### 1.3.4 测试环境是否配置成功

##### 1) 浏览器访问测试页面验证。

```
client ~]# curl http://192.168.4.5/index.html    #查看是否有数据
```

#### 1.3.5 部署测试页面

### 1) 部署测试页面(Web1 服务器)

测试页面可以参考 `lnmp_soft/php_scripts/php-memcached-demo.tar.gz`。

```
web1 ~]# cd lnmp_soft/php_scripts/
```

```
web1 php_scripts]# tar -xf php-memcached-demo.tar.gz
```

```
web1 php_scripts]# cd php-memcached-demo
```

```
web1 php-memcached-demo]# cp -r * /usr/local/nginx/html/
```

### 2) 浏览器直接访问后端服务器的测试页面 (Web1 服务器)

```
web1 ~]# firefox http://192.168.2.100/index.php #填写账户信息
```

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

```
web1 ~]# ls
```

```
sess_ahilcq9bguot0vqsjtd84k7244 #注意这里的 ID 是随机的
```

```
web1 ~]# cat sess_ahilcq9bguot0vqsjtd84k7244
```

注意：可用修改 `index.php` 和 `home.php` 两个文件的内容，添加页面颜色属性，以区别后端两台不同的服务器：`<body bgcolor=blue>`。

### 3) 部署测试页面(Web2 服务器)

测试页面可以参考 `lnmp_soft/php_scripts/php-memcached-demo.tar.gz`。

```
web2 ~]# cd lnmp_soft/php_scripts/
```

```
web2 php_scripts]# tar -xf php-memcached-demo.tar.gz
```

```
web2 php_scripts]# cd php-memcached-demo
```

```
web2 php-memcached-demo]# cp -r * /usr/local/nginx/html/
```

#### 4) 浏览器直接访问后端服务器的测试页面 (Web2 服务器)

```
web2 ~]# firefox http://192.168.2.200 #填写账户信息
```

```
web2 ~]# cd /var/lib/php/session/ #查看服务器本地的 Session 信息
```

```
web2 ~]# ls
```

```
sess_qqekltmel07br8f63d6v9ch401 #注意这里的 ID 是随机的
```

```
web2 ~]# cat sess_qqekltmel07br8f63d6v9ch401
```

注意：可用修改 `index.php` 和 `home.php` 两个文件的内容，添加页面颜色属性，以区别后端两台不同的服务器：`<body bgcolor=green>`。

#### 5) 浏览器访问前端调度器测试 (不同后端服务器 Session 不一致)

推荐使用 google 浏览器测试。

```
client ~]# google-chrome http://192.168.4.5
```

#填写注册信息后，刷新，还需要再次注册，说明两台计算机使用的是本地 Session

#第二台主机并不知道你再第一台主机已经登录，第一台主机的登录信息也没有传递给第二台主机

## 二 案例 2：构建 memcached 服务

### 2.1 问题

本案例要求先快速搭建好一台 memcached 服务器，并对 memcached 进行简单的增、删、改、查操作：

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

## 2.2 方案

使用 1 台 RHEL7 虚拟机作为 memcached 服务器（192.168.4.5）。

在 RHEL7 系统光盘中包含有 memcached，因此需要提前配置 yum 源，即可直接使用 yum 安装，客户端测试时需要提前安装 telnet 远程工具。

验证时需要客户端主机安装 telnet，远程 memcached 来验证服务器的功能：

```
add name 0 180 10 #变量不存在则添加
set name 0 180 10 #添加或替换变量
replace name 0 180 10 #替换
get name #读取变量
append name 0 180 10 #向变量中追加数据
delete name #删除变量
flush_all #清空所有
```

提示：0 表示不压缩，180 为数据缓存时间(秒)，10 为需要存储的数据字节数量。

## 2.3 步骤

### 2.3.1 构建 memcached 服务

1) 使用 yum 安装软件包 memcached(缓存数据库软件,数据放在内存,重启全丢失)

```
proxy ~]# yum -y install memcached
```

```
proxy ~]# rpm -qa memcached
```

```
memcached-1.4.15-10.el7_3.1.x86_64
```



## 2) memcached 配置文件（查看即可，不需要修改）

```
proxy ~]# vim /usr/lib/systemd/system/memcached.service
```

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

```
proxy ~]# vim /etc/sysconfig/memcached
```

```
PORT="11211"
```

```
USER="memcached"
```

```
MAXCONN="1024"
```

```
CACHE_SIZE="64" #默认 M 为单位
```

```
OPTIONS=""
```

## 3) 启动服务并查看网络连接状态验证是否开启成功：

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，使用 ss 命令可以替代 netstat，功能与选项一样。

```
proxy ~]# systemctl start memcached
proxy ~]# systemctl status memcached
proxy ~]# netstat -anptu | grep memcached
tcp 0 0 0.0.0.0:11211 0.0.0.0:* LISTEN 2839/memcached
tcp 0 0 :::11211 :::* LISTEN 2839/memcached
udp 0 0 0.0.0.0:11211 0.0.0.0:* 2839/memcached
udp 0 0 :::11211 :::* 2839/memcached
proxy ~]# setenforce 0
proxy ~]# firewall-cmd --set-default-zone=trusted
```

### 2.3.2 使用 telnet 访问 memcached 服务器

#### 1) 使用 yum 安装 telnet

```
proxy ~]# yum -y install telnet
```

#### 2) 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

```
proxy ~]# telnet 192.168.4.5 11211
```

```
Trying 192.168.4.5...
```

```
.....
```

##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。

```
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, 不能修改值

set myname 0 180 10 #添加或替换变量, 能新建变量和替换变量的值

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

flush\_all #清空所有

quit #退出登录

### 三 案例 3: LNMP+memcached

#### 3.1 问题

沿用练习一和练习二, 部署 LNMP+memcached 网站平台, 通过 PHP 页面实现对 memcached 服务器的数据操作, 实现以下目标:

为 PHP 安装 memcache 扩展

创建 PHP 页面, 并编写 PHP 代码, 实现对 memcached 的数据操作

## 3.2 方案

如果希望使用 PHP 来操作 memcached，注意必须要为 PHP 安装 memcache 扩展（php-pecl-memcache），否则 PHP 无法解析连接 memcached 的指令。客户端测试时需要提前安装 telnet 远程工具。

## 3.3 步骤

实现此案例需要按照如下步骤进行。

### 3.3.1 创建 PHP 页面，使用 PHP 语言测试 memcached 服务

#### 1) 部署测试页面

创建 PHP 首页文档 /usr/local/nginx/html/mem.php，测试页面可以参考 lnmp\_soft/php\_scripts/mem.php。

注意：192.168.2.5 是 memcached 数据库。

```
web1 ~]# vim /usr/local/nginx/html/mem.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;

?>
```

## 2) 客户端测试 (结果会失败)

客户端使用浏览器访问服务器 PHP 首页文档, 检验对 memcached 的操作是否成功:

```
web1 ~]# firefox http://192.168.2.100/mem.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。

## 3) 为 PHP 添加 memcache 扩展

```
web1 ~]# yum -y install php-pecl-memcache
```

```
web1 ~]# systemctl restart php-fpm
```

## 4) 客户端再次测试 (结果会成功显示数据结果)

```
web1 ~]# firefox http://192.168.2.100/mem.php
```

# 四 案例 4: PHP 实现 session 共享

## 4.1 问题

沿用练习三, 通过修改 PHP-FPM 配置文件, 实现 session 会话共享:

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

## 4.2 方案

在练习三拓扑的基础上, 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。拓扑结构如图所示。



## 4.3 步骤

### 4.3.1 在后端 LNMP 服务器上部署 Session 共享

注意：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。

#### 1) 为 PHP 添加 memcache 扩展

注意，因为后端两台 web 服务器(web1,web2)都需要连接 memcached 数据库，所以两台主机都需要安装 PHP 扩展模块(下面也 web1 为例)。

```
web1 ~]# yum -y install php-pecl-memcache
```

#### 2) 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)。

```
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 Session 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）

```
web1 ~]# systemctl restart php-fpm
```

步骤三：客户端测试

客户端使用浏览器访问两台不同的 Web 服务器。

操作步骤参考练习一，最终可以获得相关的 Session ID 信息。