# 8. Web 服务器-Apache

## 一. 讲在 Apache 之前

**HTML语言**：超文本标记语言，使用html语言编写的文本叫超文本，“超文本”就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素。

**HTTP协议**：超文本传输协议
HTTP使用统一资源标识符（URL）来建立连接和传输数据。是一个基于TCP/IP通信协议来传递数据的协议，属于应用层协议。

**URL**：统一资源定位符
统一资源定位符是对可以从互联网上得到的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址。
格式：

http://www.atguigu.com:80/image/a.jpg

知识拓展：
URI：统一资源标志符，URI与URL都是定位资源位置的，就是表示这个资源的位置信息，就像经纬度一样可以表示你在世界的哪个角落。URI是一种宽泛的含义更广的定义，而URL则是URI的一个子集，就是说URL是URI的一部分。

## 二. Apache 详解

### 1. 概述

Apache是世界使用排名第一的Web服务器软件。它可以运行在几乎所有广泛使用的计算机平台上，由于其跨平台和安全性被广泛使用，是最流行的Web服务器端软件之一。它快速、可靠并且可通过简单的API扩充，将Perl/Python/php等解释器编译到服务器中。

Apache有多种产品，可以支持SSL技术，支持多个虚拟主机。Apache是以进程为基础的结构，进程要比线程消耗更多的系统开支，不太适合于多处理器环境，因此，在一个Apache Web站点扩容时，通常是增加服务器或扩充群集节点而不是增加处理器。到目前为止Apache仍然是世界上用的最多的Web服务器，市场占有率达60%左右。

### 2. 工作模式

Apache一共有3种稳定的MPM模式(MPM：多进程处理模块)，它们分别是 prefork、worker、event

**prefork 工作模式**

Apache在启动之初，就预先fork一些子进程，然后等待请求进来。之所以这样做，是为了减少频繁创建和销毁进程的开销。每个子进程只有一个线程，在一个时间点内，只能处理一个请求。

优点：成熟稳定，兼容所有新老模块。同时，不需要担心线程安全的问题。

缺点：一个进程相对占用更多的系统资源，消耗更多的内存。而且，它并不擅长处理高并发请求。

**worker 工作模式**

使用了多进程和多线程的混合模式。它也预先fork了几个子进程(数量比较少)，然后每个子进程创建一些线程，同时包括一个监听线程。每个请求过来，会被分配到1个线程来服务。线程比起进程会更轻量，因为线程通常会共享父进程的内存空间，因此，内存的占用会减少一些。在高并发的场景下，因为比起prefork有更多的可用线程，表现会更优秀一些。

优点：占据更少的内存，高并发下表现更优秀。

缺点：必须考虑线程安全的问题。

**event 工作模式**

它和worker模式很像，最大的区别在于，它解决了keep-alive场景下，长期被占用的线程的资源浪费问题。event MPM中，会有一个专门的线程来管理这些keep-alive类型的线程，当有真实请求过来的时候，将请求传递给服务线程，执行完毕后，又允许它释放。这样增强了高并发场景下的请求处理能力。

HTTP采用keepalive方式减少TCP连接数量，但是由于需要与服务器线程或进程进行绑定，导致一个繁忙的服务器会消耗完所有的线程。Event MPM是解决这个问题的一种新模型，它把服务进程从连接中分离出来。在服务器处理速度很快，同时具有非常高的点击率时，可用的线程数量就是关键的资源限制，此时Event MPM方式是最有效的，但不能在HTTPS访问下工作。

**查看方式：**

```
# httpd -V | grep -i "server mpm"
```

**指定方式：**

在编译时，在选项中指定，--with-mpm=xxx

# 3. 相关文件保存位置

**配置文件位置：**

源码包安装：　PREFIX/etc/httpd.conf（主配置文件）
　　PREFIX/etc/extra/*.conf（子配置文件）

rpm包安装：　/etc/httpd/conf/httpd.conf

**网页文件位置：**

源码包安装：　PREFIX/htdocs/

rpm包安装：　/var/www/html/

**日志文件位置：**

源码包安装：　PREFIX/logs/

rpm包安装：　/var/log/httpd/

# 4. 配置文件详解

注意：apache配置文件严格区分大小写

**针对主机环境的基本配置参数**

```
ServerRoot      /usr/local/apache2  #apache主目录
Listen          :80                 #监听端口
LoadModule      php7                #加载的相关模块
User
Group                       #用户和组
ServerAdmin                 #管理员邮箱
ServerName                  #服务器名（没有域名解析时，使用临时解析。默认不开启）
ErrorLog "logs/error_log                        #服务器错误日志
CustomLog "logs/access_log" common              #访问记录日志
DirectoryIndex index.html index.php             #默认网页文件名,优先级顺序
Include  etc/extra/httpd-vhosts.conf            #子配置文件中内容也会加载生效
```

**主页目录及权限**

```
DocumentRoot "/usr/local/apache2/htdocs"
#网页文件存放目录（默认）
<Directory "/usr/local/apache2/htdocs">
#定义指定目录的权限
    Options Indexes FollowSymLinks
        None                #没有任何额外权限
        All                 #所有权限（除去MultiViews以外）
        Indexes             #浏览权限（当此目录下没有默认网页文件时，显示目录内容）

        FollowSymLinks      #准许软连接到其他目录
        MultiViews          #准许文件名泛匹配（需要手动开启模块才有效negotiation）
    AllowOverride None
    #定义是否允许目录下.htaccess文件中的权限生效
        None                        #.htaccess中权限不生效
        All                         #文件中所有权限都生效
        AuthConfig                  #文件中，只有网页认证的权限生效
    Require all granted（denied）   #访问控制列表
</Directory>

<IfModule dir_module>           #此标签用来指定访问到指定目录时自动加载哪个页面文件
        DirectoryIndex index.php index.html     #可以写多个，但是有优先级之分
</IfModule>
```

# 5. Apache 实验

**实验环境：**建议使用之前搭建好的 lamp 环境进行试验测试

## 1) Apache 的目录别名

当 apache 接受请求时，在默认情况下会将 DocumentRoot 目录中的文件送到客户端，如果想将某一不在 DocumentRoot 目录中的文件共享到网站上，并希望将它们留在本来位置而不需要进行移动的话，处理这种情况可以通过建立别名的方式将 URL 指向特定的目录

**1、编辑主配置文件**

```
# vim /usr/local/apache2/conf/httpd.conf
  Include etc/extra/httpd-autoindex.conf            #去掉注释，开启调用子配置文件
```

**2、编辑子配置文件**

```
# vim /usr/local/apache2/conf/extra/httpd-autoindex.conf
  alias /icons/ "/usr/local/apache2/icons/"
  结构：别名 "真实目录"   #真实目录的结尾要有/，否则报错
  <Directory "/usr/local/apache2/icons">
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
  </Directory>
可以根据模板编写一个自己需要的目录别名
```

## 2) Apache 的用户认证

有时候，我们需要给一些特殊的访问设置一个用户认证机制，增加安全。比如我们的个人网站，一般都是有一个管理后台的，虽然管理后台本身就有密码，但我们为了更加安全，可以再设置一层用户身份认证。

**1、编辑配置文件**

```
# vim /usr/local/apache2/etc/httpd.conf
```

在需要进行登录认证的目录标签中加入如下配置：

```
<Directory "/usr/local/apache2/htdocs/admin"> #声明被保护目录
        Options Indexes FollowSymLinks
        AllowOverride All                    #开启权限认证文件.htaccess
        Require all granted
</Directory>
```

**2、在指定目录下创建权限文件**

切换到/usr/local/apache2/htdocs/admin，创建 .htaccess 文件，并添加下面的内容

```
# vi .htaccess
AuthName "Welcome to atguigu"
#提示信息
AuthType basic
#加密类型
AuthUserFile /usr/local/apache2/htdocs/admin/apache.passwd
#密码文件，文件名自定义。（使用绝对路径）
require valid-user
#允许密码文件中所有用户访问
```

**3、建立密码文件，加入允许访问的用户。（此用户和系统用户无关）**

```
# htpasswd    -c    /usr/local/apache2/htdocs/admin/apache.passwd    test1
-c    建立密码文件，只有添加第一个用户时，才能-c
# htpasswd    -m    /usr/local/apache2/htdocs/admin/apache.passwd    test2
-m    再添加更多用户时，使用-m 参数
注意：  htpasswd  该命令是 httpd 的命令，需要绝对路径
```

**4、重启 apache 服务**

```
/usr/local/apache2/bin/apachectl -t
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start
```

先检查配置是否正确，然后通过浏览器输入要访问的资源时就会提示输入密码了。

# 3) 虚拟主机（重点）

虚拟主机，也叫“网站空间”，就是把一台运行在互联网上的物理服务器划分成多个“虚拟”服务器。虚拟主机技术极大的促进了网络技术的应用和普及。同时虚拟主机的租用服务也成了网络时代的一种新型经济形式。

**虚拟主机的分类：**

基于 IP 的虚拟主机：一台服务器，多个 ip，搭建多个网站
基于端口的虚拟主机：一台服务器，一个 ip，搭建多个网站，每个网络使用不同端口访问
**基于域名的虚拟主机：一台服务器，一个 ip，搭建多个网站，每个网站使用不同域名访问**

**实验准备：**

**1. 域名解析：准备两个域名**

www.sohu.com
www.sina.com
#使用本地 hosts 文件进行解析

**2. 网站主页目录规划**

在/htdocs/目录下分别创建 sohu 和 sina 两个目录，并在新建目录内创建 index.html 文件（分别写

不一样的内容）

**试验步骤：**

**1. 修改主配置文件开启文件关联**

```
# vi /usr/local/apache2/etc/httpd.conf
  Include etc//extra/httpd-vhosts.conf       #此行取消注释
```

**2. 编辑子配置文件，编写虚拟主机标签**

```
# vi  /usr/local/apache2/etc/extra/httpd-vhosts.conf
  添加下方内容，有几个虚拟主机就写几组（添加之前先把原先存在的示例删除掉）
  <Directory "/usr/local/apache2/htdocs/sina">
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
  </Directory>
  #目录权限标签根据需要自行添加
  <VirtualHost 192.168.88.10:80>                          #虚拟主机标签
    ServerAdmin webmaster@sina.com                        #管理员邮箱
    DocumentRoot "/usr/local/apache2/htdocs/sina"         #网站主目录
    ServerName www.sina.com                               #完整域名
    ErrorLog "logs/sina-error_log"                        #错误日志
    CustomLog "logs/sina-access_log" common               #访问日志
  </VirtualHost>
```

**3. 重启服务，验证结果**

Windows 下：浏览器下输入两个不同的域名验证网页内容（提前修改 windows 的 hosts 文件）

Linux 下：通过 elinks/curl 命令验证：elinks/curl URL 地址（提前修改 windows 的 hosts 文件）

# 4) 域名跳转

一个站点难免会有多个域名，而多个域名总得有一个主次，比如我的网站可以用两个域名访问：www.sina.com 和 www.sohu.cn 但大家发现不管我用哪个域名访问，最终都会跳转到 www.sina.com 上来。这个行为就叫做域名跳转，状态码：301 是永久跳转，302 是临时跳转，网站上一定要设置为 301，这样对搜索引擎是比较友好的。

**实验条件：**

1. 虚拟主机能正常访问
2. **打开主配置文件开启重写模块**

```
  LoadModule rewrite_module modules/mod_rewrite.so      #取消注释
```

**实验步骤：**

**1. 修改虚拟主机配置文件**

```
# vi  */extra/httpd-vhosts.conf
  <Directory "/usr/local/apache2/htdocs/sohu">
        Options Indexes FollowSymLinks
```

```
        AllowOverride All
        Require all granted
  </Directory>
```

**2. 创建规则匹配文件**

```
# vi  */.htaccess
  在指定的网站目录下创建文件，并添加以下内容
  RewriteEngine on
  # 开启rewrite功能
  RewriteCond %{HTTP_HOST} ^www.sohu.com
  # 把以www.sina.com    开头的内容赋值给HTTP_HOST变量
  RewriteRule  ^(.*)$   http://www.sina.com/$1 [R=permanent,L]
  # ^(.*)$ 指代客户端要访问的资源
  # $1 把 .* 所指代的内容赋值到$1变量中
  # R=permanent 永久重定向 = 301
  # L 指定该规则为最后一条生效的规则，以后的不再生效
```

**3. 重启服务器并测试**

```
/usr/local/apache2/bin/apachectl -t
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start
```

通过上述测试，发现无论是 sina 或 sohu 最终都是访问到 www.sina.com 域名上来则试验成功

## 5) Apache+openssl 实现 https（重点）

HTTPS（全称：Hypertext Transfer Protocol Secure，超文本传输安全协议），是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。即 HTTP 下加入 SSL 层，用于安全的 HTTP 数据传输。这个系统被内置于浏览器中，提供了身份验证与加密通讯方法。现在它被广泛用于万维网上安全敏感的通讯，例如交易支付方面。

**1. 准备工作：**

检查 Apache 是否支持 SSL，检查相应模块是否安装，若安装则将模块启用

**模块存放目录：**/usr/local/apache2/modules

**检查模块是否启用：**apachectl -M

**2. CA 证书申请：**

a. openssl genrsa -out ca.key 1024

#建立服务器私钥，生成 RSA 密钥

b. openssl req -new -key ca.key -out atguigu.csr

#需要依次输入国家，地区，城市，组织，组织单位，Email 等信息。最重要的是有一个 common name，可以写你的名字或者域名。如果为了 https 申请，这个必须和域名吻合，否则会引发浏览器警报。生成的 csr 文件交给 CA 签名后形成服务端自己的证书

c. openssl x509 -req -days 365 -sha256 -in atguigu.csr -signkey ca.key -out atguigu.crt

#使用 CA 服务器签发证书，设置证书的有效期等信息

注意 1：生成完秘钥和证书文件后，将文件存放在 Apache 的安装目录下的 cert 目录下

注意 2：在生产环境中必须要在 https 证书厂商注册（否则浏览器不识别）

**3. 配置文件修改：**

**a. 调用 ssl 模块，并启用 ssl 独立配置文件**

```
# vim /usr/local/apache2/etc/httpd.conf
   LoadModule ssl_module modules/mod_ssl.so        #取消注释
   Include etc/extra/httpd-ssl.conf                #取消注释
```

**b. 修改 conf/extra/httpd-ssl.conf 配置文件，调用证书等文件**

```
   #添加 SSL 协议支持协议，去掉不安全的协议
   SSLProtocol all -SSLv2 -SSLv3
   #修改加密套件如下
   SSLCipherSuite HIGH:!RC4:!MD5:!aNULL:!eNULL:!NULL:!DH:!EDH:!EXP:+MEDIUM
   SSLHonorCipherOrder on
   #证书公钥配置（签字的）
   SSLCertificateFile cert/atguigu.crt
   #证书私钥配置
   SSLCertificateKeyFile cert/ca.key
```

**c. 修改主配置文件，添加虚拟主机**

```
<VirtualHost _default_:443>
    # DocumentRoot 目录位置要和 httpd.conf 里面的一致
    DocumentRoot "/usr/local/apache2/htdocs"
    ServerName localhost:443
    SSLCertificateFile cert/atguigu.crt
    SSLCertificateKeyFile cert/ca.key
    SSLCertificateChainFile cert/atguigu.crt
</VirtualHost>
```

**4. 结果验证：**

```
# apachectl -t #检查配置文件语法
# apachectl restart #重启 apache，并测试是否可以使用 https 访问
```

**报错提示：** AH00526: Syntax error on line 78 of /usr/local/apache2/etc/extra/httpd-ssl.conf: SSLSessionCache: 'shmcb' session cache not supported (known names: ). Maybe you need to load the appropriate socache module (mod_socache_shmcb?).

**解决方案：** 要么不调用此模块，要么让调用的模块加载上

**5. 强制跳转 https：**

**有些时候为了安全，网站不允许使用 http 访问，仅允许使用 https 访问，目的是为了更加安全**

在 http 部分的目录权限标签中添加一下内容

```
<Directory "/usr/local/apache2/htdocs">
    ……
    RewriteEngine on                                #开启转发规则
    RewriteCond %{SERVER_PORT} !^443$               #检查访问端口只要目标不是443的
    RewriteRule ^(.*)?$ https://%{SERVER_NAME}/$1 [R=301,L] #全都使用https重新访问
</Directory>
```

**在做后面实验时为了更加方便理解，我们可以先把 https 关闭掉**

需要关闭：跳转&虚拟主机&ssl 配置文件调用

## 6) Apache 日志切割

我们每访问一次网站，那么就会记录若干条日志。如果日志不去管理，时间长了日志文件会越来越大，如何避免产生大的日志文件？其实 apache 有相关的配置，使日志按照我们的需求进行归档，比如每天一个新日志，或者每小时一个新的日志。

1.首先简单设置日志的路径名称

```
# vim /usr/local/apache2/etc/httpd.conf
```

编辑添加内容如下：

```
ErrorLog "logs/error.log"
CustomLog "logs/access.log" combined
```

指定了日志存放在/usr/local/apache2/logs 目录下分别为 error.log 和 access.log，combined 为日志显示的格式，日志格式可以参考配置文件 httpd.conf 中格式的指定，如下：

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

2.设置 apache 日志分割

同样编辑配置文件 httpd.conf

```
ErrorLog "|/usr/local/apache2/bin/rotatelogs -l /usr/local/apache2/logs/error_%Y%m%d.log
86400"
CustomLog "|/usr/local/apache2/bin/rotatelogs -l
/usr/local/apache2/logs/access_%Y%m%d.log 86400" combined
```

**注意 1：以上仅为两条命令（一条错误日志，一条访问日志），路径太长写不开**

**注意 2：若开启了 https，则需要修改 http-ssl.conf 配置文件中的日志记录条目**

ErrorLog 是错误日志，CustomLog 是访问日志。|就是管道符，意思是把产生的日志交给 rotatelog 这个工具，而这个工具就是 apache 自带的切割日志的工具。-l 的作用是校准时区为 UTC，也就是北京时间。86400，单位是秒，正好是一天，那么日志会每天切割一次。而最后面的 combined 为日志的格式，在 httpd.conf 中有定义。

## 7) 不记录指定文件类型的日志

如果一个网站访问量特别大，那么访问日志就会很多，但有一些访问日志我们其实是可以忽略掉的，比如网站的一些图片，还有 js、css 等静态对象。而这些文件的访问往往是巨量的，而且即使记录这些日志也没有什么用，那么如何忽略不记录这些日志呢？

1、配置日志不记录图片的访问

```
# vim /usr/local/apache2/conf/httpd.conf
```

相关配置为：

```
SetEnvIf Request_URI ".*\.gif$" image-request
SetEnvIf Request_URI ".*\.jpg$" image-request
SetEnvIf Request_URI ".*\.png$" image-request
SetEnvIf Request_URI ".*\.bmp$" image-request
SetEnvIf Request_URI ".*\.swf$" image-request
SetEnvIf Request_URI ".*\.js$"  image-request
SetEnvIf Request_URI ".*\.css$" image-request
CustomLog "|/usr/local ... _%Y%m%d.log 86400" combined env=!image-request
```

说明：在原来的访问日志配置基础上，增加了一些 image-request 的定义，比如把 gif、jpg、bmp、swf、js、css 等结尾的全标记为 image-request，然后在配置日志后加一个标记 env=!image-request，表示取反。

## 8) Apache 配置静态缓存

所说的静态文件指的是图片、js、css 等文件，用户访问一个站点，其实大多数元素都是图片、js、css 等，这些静态文件其实是会被客户端的浏览器缓存到本地电脑上的，目的就是为了下次再请求时不再去服务器上下载，这样就加快了速度，提高了用户体验。但这些静态文件总不能一直缓存，它总有一些时效性，那么就得设置这个过期时间。

1、配置静态缓存

```
# vim /usr/local/apache2/conf/httpd.conf
<IfModule mod_expires.c>            #此模块默认未启用，请手动启用
    ExpiresActive on
    ExpiresByType image/gif "access plus 1 days"
    ExpiresByType image/jpeg "access plus 24 hours"
    ExpiresByType image/png "access plus 24 hours"
    ExpiresByType text/css "now plus 2 hours"
    ExpiresByType application/x-javascript "now plus 2 hours"
    ExpiresByType application/javascript "now plus 2 hours"
    ExpiresByType application/x-shockwave-flash "now plus 2 hours"
    ExpiresDefault "now plus 0 min"
</IfModule>
```

或者使用 mod_headers 模块实现：该模块默认启用

```
<IfModule mod_headers.c>
    # htm,html,txt 类的文件缓存一个小时
    <filesmatch "\.(html|htm|txt)$">
```

```
        header set cache-control "max-age=3600"
    </filesmatch>

    # css, js, swf 类的文件缓存一个星期
    <filesmatch "\.(css|js|swf)$">
        header set cache-control "max-age=604800"
    </filesmatch>

    # jpg,gif,jpeg,png,ico,flv,pdf 等文件缓存一年
    <filesmatch "\.(ico|gif|jpg|jpeg|png|flv|pdf)$">
        header set cache-control "max-age=29030400"
    </filesmatch>
</IfModule>
```

说明：这里的时间单位可以 days、hours 甚至是 min，两种不同的方法，上面使用的是 mod_expires，而下面用的是 mod_headers，要想使用这些模块，必须要事先已经支持。如何查看是否支持，使用命令：

```
# /usr/local/apache2/bin/apachectl -M
```

2、重启服务器并验证

```
/usr/local/apache2/bin/apachectl -t
/usr/local/apache2/bin/apachectl stop
/usr/local/apache2/bin/apachectl start
```

验证：

```
# curl -x127.0.0.1:80 'http://www.sohu.com/image/a.jpg' -I
HTTP/1.1 200 OK
Date: Wed, 26 Oct 2016 03:51:26 GMT
Server: Apache/2.2.31 (Unix) PHP/5.5.38
Last-Modified: Tue, 31 May 2016 03:08:36 GMT
ETag: "46891b-16b-5341ab0597500"
Accept-Ranges: bytes
Content-Length: 363
Cache-Control: max-age=86400
Expires: Thu, 27 Oct 2016 03:51:26 GMT
Content-Type: image/jpg
```

# 9) 禁止解析 PHP

某个目录下禁止解析 PHP，这个很有作用，我们做网站安全的时候，这个用的很多，比如某些目录可以上传文件，为了避免上传的文件有木马，所以我们禁止这个目录下面的访问解析 PHP。

**配置禁止解析 php**

```
<Directory /usr/local/apache2/htdocs/data>
    php_admin_flag engine off
    <filesmatch "(.*)php">
        Order deny,allow
        Deny from all
    </filesmatch>
</Directory>
```