**第十五章 LAMP调优**

**本节所讲内容：**

**LAMP之apache调优**

**15.1 隐藏apache版本信息**

**15.2 设置错误页面-开启压缩和缓存功能**

**15.3 开启长连接功能**

**15.4 apache运行模式-prefork-worker运行模式介绍**

**15.6 prefork运行模式详解**

**15.7 worker运行模式详解**

**15.7 rewrite-禁止网站下某个目录执行PHP文件-apache调优总结**

**生产环境中，部署了apache之后，我们应该从安全还是性能角度，在apache服务上线之前，对其做诸多的优化调试才行。**

**实验环境：**

**操作系统： centos7.4**

**apache版本：httpd-2.4.10**

**PHP版本: php5.6.32**

**源码包存放位置：/usr/src**

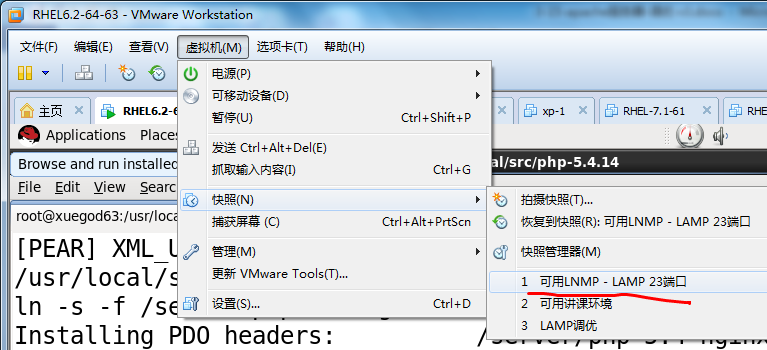
**源码包编译安装位置：**

**apache： /usr/local/apache/**

**php：/server/php-5.4**

**mysql：/server/mysql-5.5/**

**恢复实验环境：**



**源码官方下，补丁及时打**

**在官网上下载apache源码包。不要去乱七八糟的站点进行下载，防止源码包被别有用心的人动过手脚，导致后面对公司业务造成不必要损失。**

**关注apache的官网的补丁发布情况，一旦有新的补丁，我们一定要及早打上。特别是一些安全补丁，防止受到损失。**

**15.1 隐藏apache版本信息**

**15.1.1 屏蔽apache版本等敏感信息**

**查看apache版本信息：**

**[root@xuegod63 ~]# curl -I 192.168.1.63**

**HTTP/1.1 200 OK**

**Date: Thu, 25 Jan 2018 12:52:58 GMT**

**Server: Apache/2.4.10 (Unix)**

**Last-Modified: Mon, 11 Jun 2007 18:53:14 GMT**

**ETag: "2d-432a5e4a73a80"**

**Accept-Ranges: bytes**

**Content-Length: 45**

**Content-Type: text/html**

**[root@xuegod63 ~]# curl -I www.baidu.com**

**HTTP/1.1 200 OK**

**Server: bfe/1.0.8.18**

**Date: Thu, 25 Jan 2018 08:26:08 GMT**

**Content-Type: text/html**

**Content-Length: 277**

**Last-Modified: Mon, 13 Jun 2016 02:50:08 GMT**

**Connection: Keep-Alive**

**ETag: "575e1f60-115"**

**Cache-Control: private, no-cache, no-store, proxy-revalidate, no-transform**

**Pragma: no-cache**

**Accept-Ranges: bytes0**

**[root@xuegod63 ~]# curl -I www.tencent.com**

**HTTP/1.1 200 OK**

**Date: Tue, 17 Nov 2015 02:20:16 GMT**

**Server: Apache**

**Last-Modified: Thu, 18 Jun 2009 04:09:44 GMT**

**Accept-Ranges: bytes**

**Content-Length: 150**

**Cache-Control: max-age=0**

**Expires: Tue, 17 Nov 2015 02:20:16 GMT**

**Vary: Accept-Encoding**

**Connection: close**

**Content-Type: text/html**

**[root@xuegod63 ~]# curl -I www.taobao.com**

**HTTP/1.1 302 Found**

**Server: Tengine**

**Date: Tue, 17 Nov 2015 02:22:06 GMT**

**Content-Type: text/html**

**Content-Length: 258**

**Connection: keep-alive**

**Location: https://www.taobao.com/**

**Set-Cookie: thw=cn; Path=/; Domain=.taobao.com; Expires=Wed, 16-Nov-16 02:22:06 GMT;**

**1、我们在apache主配置文件httpd.conf中，找到包含httpd-default.conf的行，并解开注释**

**[root@xuegod63 ~]# vim /etc/conf/httpd.conf**

**475 #Include conf/extra/httpd-default.conf**

**为：**

**Include conf/extra/httpd-default.conf**

**注意：编译安装的情况下，只有此行解开注释了，后面的修改才能生效。**

**2）打开httpd-default.conf文件，修改如下两个地方**

**[root@xuegod63 ~]# vim /etc/httpd/extra/httpd-default.conf**

**改：**

**55 ServerTokens Full**

**65 ServerSignature On**

**为：**

**ServerTokens Prod**

**ServerSignature Off**

**注：**

**服务器标记产品**

**服务器签名**

**重启服务：**

**[root@xuegod63 ~]#systemctl restart httpd**

**测试：**

**[root@xuegod63 ~]# curl -I 192.168.1.63**

**HTTP/1.1 200 OK**

**Date: Sat, 29 Aug 2015 09:07:00 GMT**

**Server: Apache**

**X-Powered-By: PHP/5.4.14**

**Content-Type: text/html**

**注：还是会出现信息：Server: Apache**

**15.1.2 提升：彻底让版本等敏感信息消失**

**要彻底将版本之类的信息进行改头换面，需要在编译之前做准备或者进行重新编译了。在重新编译时，修改源码包下include目录下的ap\_release.h文件**

**[root@xuegod63 httpd-2.4.10]# pwd**

**/usr/src**

**[root@xuegod63 src]# rm -rf httpd-2.4.10**

**[root@xuegod63 src]# tar zxf httpd-2.4.10.tar.bz2**

**[root@xuegod63 src]# cd httpd-2.4.10**

**[root@xuegod63 httpd-2.4.10]# vim include/ap\_release.h**

**改：**

**40 #define AP\_SERVER\_BASEVENDOR "Apache Software Foundation"**

**41 #define AP\_SERVER\_BASEPROJECT "Apache HTTP Server"**

**42 #define AP\_SERVER\_BASEPRODUCT "Apache"**

**43**

**44 #define AP\_SERVER\_MAJORVERSION\_NUMBER 2**

**45 #define AP\_SERVER\_MINORVERSION\_NUMBER 4**

**46 #define AP\_SERVER\_PATCHLEVEL\_NUMBER 25**

**47 #define AP\_SERVER\_DEVBUILD\_BOOLEAN 0**

**为：**

**#define AP\_SERVER\_BASEVENDOR "xuegod"**

**#define AP\_SERVER\_BASEPROJECT "web server"**

**#define AP\_SERVER\_BASEPRODUCT "web"**

**#define AP\_SERVER\_MAJORVERSION\_NUMBER 8**

**#define AP\_SERVER\_MINORVERSION\_NUMBER 1**

**#define AP\_SERVER\_PATCHLEVEL\_NUMBER 2**

**#define AP\_SERVER\_DEVBUILD\_BOOLEAN 3**

**注释：**

**#define AP\_SERVER\_BASEVENDOR "Apache Software Foundation" #服务的供应商名称**

**#define AP\_SERVER\_BASEPROJECT "Apache HTTP Server" #服务的项目名称**

**#define AP\_SERVER\_BASEPRODUCT "Apache" #服务的产品名**

**#define AP\_SERVER\_MAJORVERSION\_NUMBER 2 #主要版本号**

**#define AP\_SERVER\_MINORVERSION\_NUMBER 4 #小版本号**

**#define AP\_SERVER\_PATCHLEVEL\_NUMBER 6 #补丁级别**

**#define AP\_SERVER\_DEVBUILD\_BOOLEAN 0 #**

**注：上述列出的行，大家可以修改成自己想要的，然后编译安装之后，再对httpd-default.conf文件进行修改，对方就彻底不知道你的版本号了。**

**15.1.3 源码编译安装apache**

**[root@xuegod63 httpd-2.4.10]# pwd**

**/usr/local/src/httpd-2.4.10**

**[root@xuegod63 httpd-2.2.11]# yum install openssl-devel pcre-devel**

**root@xuegod63 httpd-2.4.10]# ./configure --prefix=/usr/local/apache --sysconfdir=/etc/httpd --enable-so --enable-ssl --enable-cgi --enable-rewrite --with-zlib --with-pcre --with-apr=/usr/local/apr --enable-deflate --with-apr-util=/usr/local/apr-util --enable-modules=most --enable-mpms-shared=all --with-mpm=event**

**注解：**

**--prefix=/usr/local/apache          //指定程序安装路径**

**--sysconfdir=/etc/httpd          //指定配置文件、或工作目录**

**--enable-so          //开启基于DSO动态装载模块**

**--enable-ssl          //开启支持ssl协议**

**--enable-cgi          //开启cgi机制**

**--enable-rewrite          //开启支持URL重写**

**--with-zlib               //zlib是网络上发送数据报文的通用压缩库的API，在apache调用压缩工具压缩发送数据时需要调用该库**

**--with-pcre               //支持PCRE，把pcre包含进程序中，（此处没指定pcre程序所在路径，默认会在PATH环境下查找）**

**--with-apr=/usr/local/apr          //指定apr位置**

**--with-apr-util=/usr/local/apr-util          //指定apr-util**

**--enable-modeles=most          //启动模块，all表示所有，most表示常用的**

**--enable-mpms-shared=all     //启动所有的MPM模块**

**--with-mpm=event          //指定默认使用event模块**

**编译：**

**[root@xuegod63 httpd-2.4.10]# make -j 4**

**安装：**

**[root@xuegod63 httpd-2.4.10]# make install**

**查看配置文件：**

**[root@xuegod63 httpd-2.4.10]# ls /etc/httpd/httpd.conf**

**/etc/httpd/httpd.conf**

**存放网站的根目录：**

**[root@xuegod63 httpd-2.4.10]# ls /usr/local/apache/htdocs/index.html**

**/usr/local/apache/htdocs/index.html**

**修改默认首页内容：**

**httpd-2.4.10**

**启动apache：**

**配置apache 可以开机启动并且可以使用systemctl 命令启动apache服务器**

**[root@xuegod63 httpd-2.4.10]# vim /usr/lib/systemd/system/httpd.service**

**[Unit]**

**Description=The Apache HTTP Server**

**After=network.target remote-fs.target nss-lookup.target**

**Documentation=man:httpd(8)**

**Documentation=man:apachectl(8)**

**[Service]**

**Type=forking**

**EnvironmentFile=/etc/httpd/httpd.conf**

**ExecStart=/usr/local/apache/bin/apachectl**

**ExecRestart=/usr/local/apache/bin/apachectl restart**

**ExecStop=/usr/local/apache/bin/apachectl stop**

**KillSignal=SIGCONT**

**PrivateTmp=true**

**[Install]**

**WantedBy=multi-user.target**

**重新加载unit文件：**

**[root@xuegod63 httpd-2.4.10]# systemctl daemon-reload**

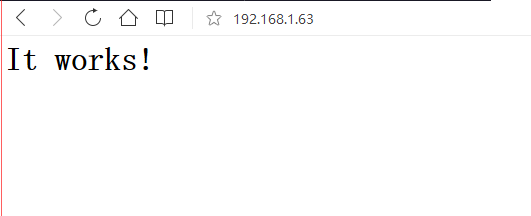
**设置开机自动启动：**

**[root@xuegod63 httpd-2.4.10]# systemctl enable httpd**

**启动apache：**

**[root@xuegod63 httpd-2.4.10]# systemctl start httpd**

**测试：**



**测试：**

**[root@xuegod63 ~]# curl -I 192.168.1.63 #看不到apache版本相关内容了**

**[root@xuegod63 httpd-2.4.10]# curl -I 192.168.1.63**

**HTTP/1.1 200 OK**

**Date: Thu, 25 Jan 2018 08:57:02 GMT**

**Server: web/8.1.2-dev (Unix)**

**Last-Modified: Thu, 25 Jan 2018 08:50:11 GMT**

**ETag: "e-56395de1913d7"**

**Accept-Ranges: bytes**

**Content-Length: 14**

**Content-Type: text/html**

**接下来再次修改：**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf**

**475 #Include conf/extra/httpd-default.conf**

**为：**

**Include conf/extra/httpd-default.conf**

**2、打开httpd-default.conf文件，修改如下两个地方**

**[root@xuegod63 ~]# vim /etc/httpd/extra/httpd-default.conf**

**改：**

**55 ServerTokens Full**

**65 ServerSignature On**

**为：**

**ServerTokens Prod**

**ServerSignature Off**

**重启服务：**

**[root@xuegod63 ~]# systemctl restart httpd**

**测试：**

**[root@xuegod63 ~]# curl -I http://192.168.1.63/**

**HTTP/1.1 200 OK**

**Date: Sat, 29 Aug 2015 09:55:31 GMT**

**Server: web**

**Last-Modified: Sat, 29 Aug 2015 09:37:36 GMT**

**ETag: "6d086-3a-51e6ff35dba19"**

**Accept-Ranges: bytes**

**Content-Length: 58**

**Content-Type: text/html**

**15.1.5 查看运行apache的默认用户**

**我们通过更改apache的默认用户，可以提升apache的安全性。这样，即使apache服务被攻破，黑客拿到apache普通用户也不会对系统和其他应用造成破坏。这里创建的apache用户，将用于对子进程和线程的控制。**

**注：默认使用daemon用户是安全的。**

**[root@xuegod63 ~]# ps -axu | grep httpd**

**[root@xuegod63 ~]# id daemon**

**uid=2(daemon) gid=2(daemon) groups=2(daemon),1(bin),4(adm),7(lp)**

**[root@xuegod63 ~]# grep daemon /etc/passwd**

**daemon:x:2:2:daemon:/sbin:/sbin/nologin**

**[root@xuegod63 ~]# useradd -M -s /sbin/nologin apache**

**－M：不要自动建立用户的登入目录。**

**编辑apache配置文件，修改默认的用户。**

**[root@c64-web /]# vim /etc/httpd/httpd.conf**

**161 User daemon**

**162 Group daemon**

**为：**

**User apache**

**Group apache**

**[root@xuegod63 httpd-2.4.10]# lsof -i :80**

**COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME**

**httpd 54066 root 4u IPv6 64673 0t0 TCP \*:http (LISTEN)**

**httpd 54083 apache 4u IPv6 64673 0t0 TCP \*:http (LISTEN)**

**httpd 54084 apache 4u IPv6 64673 0t0 TCP \*:http (LISTEN)**

**httpd 54085 apache 4u IPv6 64673 0t0 TCP \*:http (LISTEN)**

**15.1.6 apache目录及文件权限设置， 不能给777权限**

**在生产环境的网站架构中，我们应把资源文件的权限做好配置。**

**例：用户上传的图片及附件等和程序做好分离。这样才能更方便我们做好授权，保证apache服务和整个服务器安全。**

**这里我们设置apache的网站目录属主和属组是daemon:**

**[root@xuegod63 ~]# ll -sd /usr/local/apache/htdocs/**

1. **drwxr-xr-x 2 root root 24 1月 25 15:53 /usr/local/apache/htdocs/**

**[root@xuegod63 ~]# chown daemon. /usr/local/apache/htdocs/ -R**

**[root@xuegod63 ~]# ll -sd /usr/local/apache/htdocs/**

1. **drwxr-xr-x 2 daemon daemon 24 1月 25 15:53 /usr/local/apache/htdocs/**

**15.1.7 保护apache日志：设置好apache日志文件权限**

**对日志的授权，我们要将属主和属组都设置为root：**

**[root@xuegod63 ~]# ll -sd /usr/local/apache/logs/**

1. **drwxr-xr-x 2 root root 58 1月 25 17:29 /usr/local/apache/logs/**

**[root@xuegod63 ~]# ll /usr/local/apache/logs/**

**总用量 16**

**-rw-r--r-- 1 root root 912 1月 25 17:29 access\_log**

**-rw-r--r-- 1 root root 6648 1月 25 17:29 error\_log**

**-rw-r--r-- 1 root root 7 1月 25 17:29 httpd.pid**

**注：由于apache日志的记录是由apache的主进程进行操作的，而apache的主进程又是root用户启动的，所以这样不影响日志的输出。这也是日志记录的最安全的方法。**

**15.2 设置错误页面-开启压缩和缓存功能**

**15.2.1 错误页面优雅显示**

**为了提升网站的用户体验，避免404，403之类的丑陋的默认错误提示出现。我们需要对错误页面进行优化，让他们变的漂亮一点。错误页面不仅在于告诉用户访问出现了问题，而且需要引导用户到正确的页面。**

**错误页面优雅化显示的实现方式主要有两种,下面我们主要以404错误为例：**

**方法：**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf #在根目录的标签中添加以下红色标记内容。**

**217 <Directory "/usr/local/apache2.2-xuegod/htdocs">**

**。。。**

**243 ErrorDocument 404 /404.html**

**244 </Directory>**

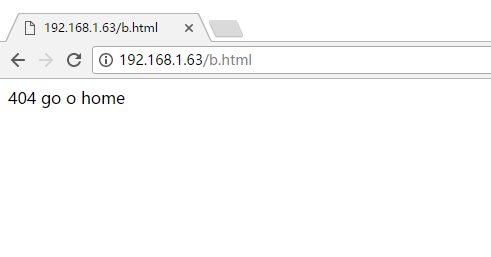
**注：#将404错误跳转到/usr/local/apache/htdocs下的404.html页面上**

**创建404测试页：**

**[root@xuegod63 ~]# echo "404 go to home" > /usr/local/apache/htdocs/404.html**

**重启：**

**[root@xuegod63 ~]#systemctl restart httpd**



**总结：ErrorDocument的命令格式如下：**

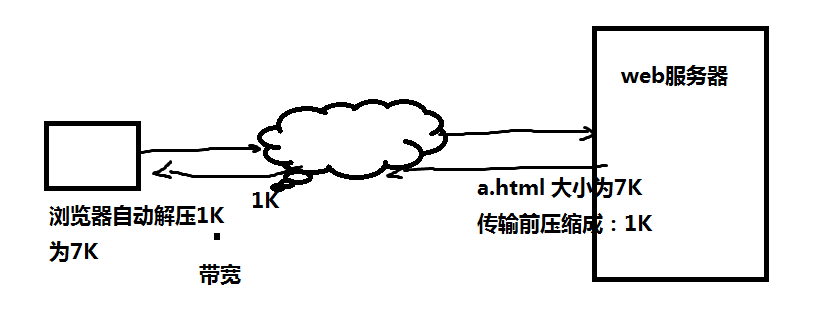
**ErrorDocument 错误代码 跳转到的页面或文件**

**另外这里需要注意，你若设置跳转到文件，必须要有这个文件才行。另外文件必须在站点目录内，不然会报错。**

**在跳转到文件的测试中，我用全路径和别名路径进行测试，当把404错误页面跳转文件放到其他目录的时候，不报错，但是页面跳转不过去。**

**15.2.2 启用压缩模块mod\_deflate**

**网站随着用户访问量的增加和内容量的增加，网站的带宽会不断的增加，随之就是网站成本的增加。并且当内容量增大的时候，客户端如果带宽小，就会影响用户的体验。因此从这两方面考虑，网站的某些内容必须经过压缩之后再传给用户，然后在用户客户端进行解压，来实现双方共赢的效果。**

****

**apache的压缩要用到mod\_deflate模块，该模块提供了DEFLATE输出过滤器，允许服务器在将输出内容发送到客户端以前进行压缩，以节约带宽。它的核心思想就是把文件先在服务器进行压缩，然后再进行传输，这样可以显著减少文件传输的大小。当传输完毕后，客户端浏览器会重新对压缩过的内容进行解压缩。如果没特殊情况的话，所有的文本内容都应该能被gzip压缩，例如：html（php），js，css，xml，txt等。**

1. **mod\_deflate模块检查及安装**

**检查：**

**root@xuegod63 httpd-2.4.10]# /usr/local/apache/bin/apachectl -M | grep deflate**

**Syntax OK #出现这个说明没有安装。或没任何输出，也是说明没有安装**

**如果安装了，将弹出以下内容：**

**[root@xuegod63 src]# /usr/local/apache/bin/apachectl -M | grep deflate**

**deflate\_module (static) #弹出此种结果，则为编译安装时装的**

**[root@xuegod63 src]# /usr/local/apache/bin/apachectl -M | grep deflate**

**deflate\_module (shared) #弹出此种结果，则为DSO方式安装的**

**安装了的话，就可以直接进行压缩配置了，如果没有安装，下面为安装方法**

**a）编译时安装方法**

**编译的时候跟上--enable-deflate即可实现安装**

**b）DSO方式安装。**

**扩展：DSO： Dynamic shared object动态共享对象 。DSO模块可以在编译服务器之后编译，也可以用Apache扩展工具([apxs](http://linux.chinaitlab.com/manual/ApacheManual/programs/apxs.html))编译并增加**

**使用DSO方式安装，/usr/local/apache/bin/apxs后跟的参数详解**

**-c 此选项表明需要执行编译操作。**

**-i 此选项表示需要执行安装操作，以安装一个或多个动态共享对象到服务器的modules目录。**

**-a 此选项自动增加一个LoadModule行到httpd.conf文件中，以激活此模块，或者，如果此行已经存在，则启用之。**

**[root@xuegod63 filters]#cd /usr/src/httpd-2.4.10/modules/filters**

**#切到apache源码包mod\_deflate所在的目录下**

**[root@xuegod63 filters]# /usr/local/apache/bin/apxs -c -i -a /usr/src/httpd-2.4.10/modules/filters/mod\_deflate.c**

**#以dso的方式编译安装到apache中**

**[root@xuegod63 filters]# ll /usr/local/apache/modules/mod\_deflate.so**

**-rwxr-xr-x 1 root root 61912 Aug 31 09:05 /usr/local/apache2.2-xuegod/modules/mod\_deflate.so**

**#检查mod\_deflate是否安装，成功安装这里会显示出该文件**

**15.2.4 压缩模块配置**

**1、配置压缩参数**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf #确认下面这一行是打开的。默认是打开的。**

**103 LoadModule deflate\_module modules/mod\_deflate.so**

**#此行解锁后再进行下面操作，不然下面的操作会报错。**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf #在以下内容后，插入：**

**57 <IfModule !mpm\_netware\_module>**

**58 <IfModule !mpm\_winnt\_module>**

**<ifmodule mod\_deflate.c>**

**DeflateCompressionLevel 9**

**SetOutputFilter DEFLATE**

**DeflateFilterNote Input instream**

**DeflateFilterNote Output outstream**

**DeflateFilterNote Ratio ratio**

**AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css application/javascript**

**</ifmodule>**

**注释：**

**<ifmodule mod\_deflate.c>**

**DeflateCompressionLevel 9 #压缩等级，越大效率越高，消耗CPU也越高。**

**DeflateCompressionLevel 9是指压缩程度的等级，从1到9，9是最高等级。据了解，这样做最高可以减少8成大小的传输量（看档案内容而定），最少也能够节省一半。  
DeflateCompressionLevel 预设可以采用 6 这个数值，以维持耗用处理器效能与网页压缩质量的平衡。**

**注：一般压缩等级使用6或8**

**SetOutputFilter DEFLATE #启用压缩**

**DeflateFilterNote Input instream #声明输入流的byte数量**

**DeflateFilterNote Output outstream #声明输出流的byte数量**

**DeflateFilterNote Ratio ratio #声明压缩的百分比**

**AddOutputFilterByType DEFLATE text/html text/plain text/xml text/css application/javascript #仅压缩，限制特定的MIME类型文件**

**</ifmodule>**

**注：**

**1、如果是虚拟主机，需要在<VirtualHost\*:80></VirtualHost>中添加配置即可实现压缩**

**2、图片和视频本身就是压缩格式，一般不需要压缩的。有些小图片和视频压缩后还会变大。**

**总结：我们在企业生产环境中时，在启用mod\_deflate时，一定要注意，对于太小的文件和某些格式的图片不要对它们进行压缩，有可能越压越大。**

**扩展：AddOutputFilterByTypeDEFLATE后跟的所有的压缩文件类型，后期可以参照选择。**

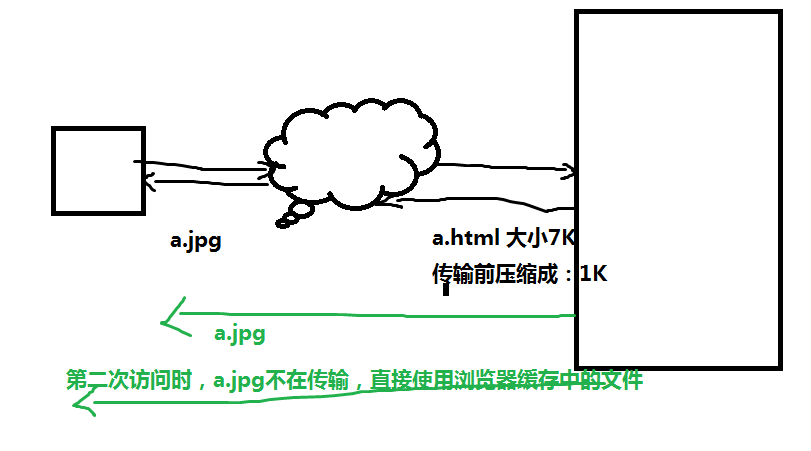
**text/plain text/html text/php text/xml text/css text/javascript**

**application/xhtml+xml application/xml application/rss+xml application/atom\_xml application/x-javascript application/x-httpd-php image/svg+xml image/gif image/png image/jpe image/swf image/jpeg image/bmp**

**15.2.5 mod\_expires： 设置网页缓存时间**

**虽然我们上面通过mod\_deflate模块启用了压缩，从很大程度上节约了企业带宽，降低了企业成本。可是由于现在越来越多的图片、脚本、css和flash被嵌入到页面中，当客户访问站点势必会做很多次的http请求，因此我们还可以通过mod\_expires缓存模块来设置ExpiresHeader来缓存这些文件。**

**Expires是通过header报文来指定特定类型的文件在游览器中的缓存时间的。平时，我们大多数的图片，flash在发布之后都是不需要经常修改的，因此做了缓存之后，游览器第一次从服务器下载之后，就不需要再从服务器下载这些文件而是直接从游览器缓存中读取了。这样客户访问页面的速度就会大大加快，企业的带宽压力也得到了缓解。**

****

**15.2.6 mod\_expires模块检查及安装**

**检查mod\_expires模块是否安装的方法如下：**

**[root@xuegod63 ~]# /usr/local/apache/bin/apachectl -M |grep expires**

**expires\_module (static) #此种结果为编译安装时装的**

**[root@xuegod63 ~]# /usr/local/apache/bin/apachectl -M |grep expires**

**expires\_module (shared) #此种结果为DSO方式安装的**

**安装了的话，就可以直接进行缓存配置了，如果没有安装，下面为安装方法**

**a）编译方式安装**

**编译的时候跟上--enable-expires即可实现安装**

**b）DSO方式安装**

**[root@xuegod63 ~]# cd /usr/src/httpd-2.4.10/modules/metadata/ #切到apache源码包mod\_expires所在的目录下**

**[root@xuegod63 metadata]# ls mod\_expires.c**

**mod\_expires.c**

**以dso的方式编译安装到apache中**

**[root@xuegod63 metadata]# /usr/local/apache/bin/apxs -c -i -a /usr/src/httpd-2.4.10/modules/metadata/mod\_expires.c**

**注意如果我们是编译安装时已经编译进去的，要先解锁以下行，再进行下面操作，不然会报错。**

**[root@xuegod63 ~]# vim /etc/httpd/conf/httpd.conf #确认这一行是开启的。**

**118 LoadModule expires\_module modules/mod\_expires.so**

**存的用法有3种，分别问对全局，对目录，对虚拟主机。**

**1、对全局**

**对全局的配置就是在apache主配置文件httpd.conf的末尾加入如下参数即可**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf #在最后添加以下内容：**

**<IfModule mod\_expires.c>**

**ExpiresActive on**

**ExpiresDefault "access plus 12 month"**

**ExpiresByType text/html "access plus 12 months"**

**ExpiresByType text/css "access plus 12 months"**

**ExpiresByType image/gif "access plus 12 months"**

**ExpiresByType image/jpeg "access plus 12 months"**

**ExpiresByType image/jpg "access plus 12 months"**

**ExpiresByType image/png "access plus 12 months"**

**EXpiresByType application/x-shockwave-flash "access plus 12 months"**

**EXpiresByType application/x-javascript "access plus 12 months"**

**ExpiresByType video/x-flv "access plus 12 months"**

**</IfModule>**

**重启服务：**

**[root@xuegod63 ~]# systemctl restart httpd**

**2、对目录**

**对目录的配置就是在apache主配置文件中<Directory></Directory>标签内，最后加入如下参数即可**

**[root@xuegod63 ~]# vim /usr/local/apache2.2-xuegod/conf/httpd.conf**

**133 <Directory "/usr/local/apache/htdocs">**

**<IfModule mod\_expires.c>**

**ExpiresActive on**

**ExpiresDefault "access plus 12 month"**

**ExpiresByType text/html "access plus 12 months"**

**ExpiresByType text/css "access plus 12 months"**

**ExpiresByType image/gif "access plus 12 months"**

**ExpiresByType image/jpeg "access plus 12 months"**

**ExpiresByType image/jpg "access plus 12 months"**

**ExpiresByType image/png "access plus 12 months"**

**EXpiresByType application/x-shockwave-flash "access plus 12 months"**

**EXpiresByType application/x-javascript "access plus 12 months"**

**ExpiresByType video/x-flv "access plus 12 months"**

**</IfModule>**

**</Directory>**

**c）对虚拟主机**

**对虚拟主机的配置就是在apache的虚拟主机配置文件httpd-vhost.conf中添加如下参数即可**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf**

**修改：**

**216 DocumentRoot "/usr/local/apache/htdocs"**

**改为：**

**# DocumentRoot "/usr/local/apache/htdocs"**

**修改：**

**467 # Include /etc/httpd/extra/httpd-vhosts.conf**

**改为：**

**Include /etc/httpd/extra/httpd-vhosts.conf**

**[root@xuegod63 ~]# vim /etc/httpd/extra/httpd-vhosts.conf**

**<VirtualHost \*:80>**

**ServerAdmin 888@qq.com**

**DocumentRoot "/www/html"**

**ServerName www.xuegod.cn**

**ServerAlias xuegod.cn**

**ErrorLog "logs/dummy-host.example.com-error\_log"**

**CustomLog "logs/dummy-host.example.com-access\_log" common**

**<Directory "/www/html">**

**Options None**

**Require all granted**

**</Directory>**

**<IfModule mod\_expires.c>**

**ExpiresActive on**

**ExpiresDefault "access plus 12 month"**

**ExpiresByType text/html "access plus 12 months"**

**ExpiresByType text/css "access plus 12 months"**

**ExpiresByType image/gif "access plus 12 months"**

**ExpiresByType image/jpeg "access plus 12 months"**

**ExpiresByType image/jpg "access plus 12 months"**

**ExpiresByType image/png "access plus 12 months"**

**EXpiresByType application/x-shockwave-flash "access plus 12 months"**

**EXpiresByType application/x-javascript "access plus 12 months"**

**ExpiresByType video/x-flv "access plus 12 months"**

**</IfModule>**

**</VirtualHost>**

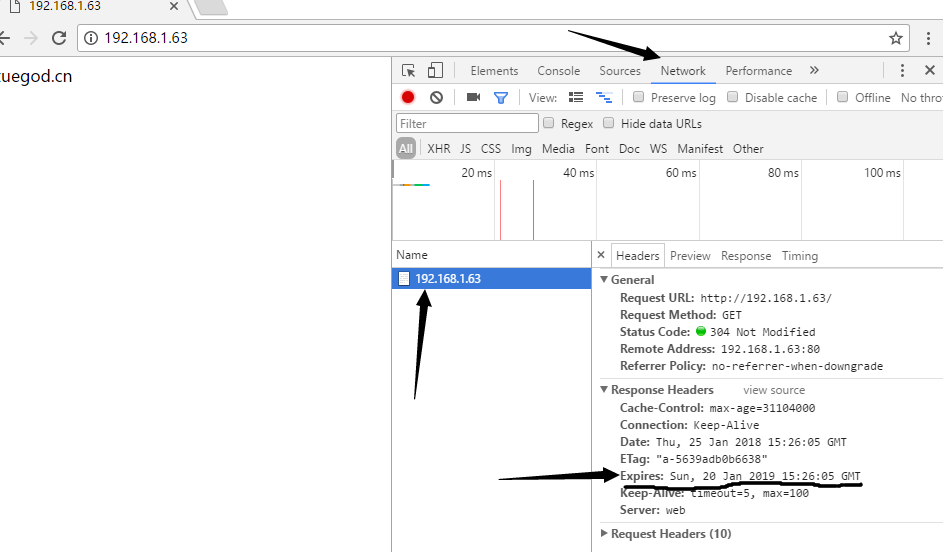
**[root@xuegod63 ~]# mkdir -pv /www/html**

**[root@xuegod63 ~]# echo “xuegod.cn”> /www/html/index.html**

**以上三种配置任何一种配置之后，对apache服务器进行优化重启，然后用火狐或者google游览器的**

**测试：**

**在浏览器中访问192.168.1.63，按下f12，点击network，再次刷新站点**



**由以上测试可见，缓存为1年**

**15.2.7 扩展：expires模块的语法**

**expires模块用到了ExpiresDefault和EXpiresByType两个指令，下面是这两个指令的语法。**

**ExpiresDefault “<base> [plus] {<num><type>}\*”**

**EXpiresByType type/encoding "<base> [plus] {<num><type>}"**

**其中<base>的参数有3个：access，now（等价于‘access’），modification**

**modification [ˌmɒdɪfɪˈkeɪʃn] 改性，修正**

**plus关键字是可选的。**

**plus [plʌs] 加上**

**<num>必须是整数，确保可以atoi（）所接收。（atoi可以把字符串转换成长整型数）**

**<type>参数类型：years，months，weeks，days，hours，minutes，seconds**

**例如：下面3个指令都表示文档默认的有效期是一个月**

**ExpiresDefault "access plus 1 month"**

**ExpiresDefault "access plus 4 weeks"**

**ExpiresDefault "access plus 30 days"**

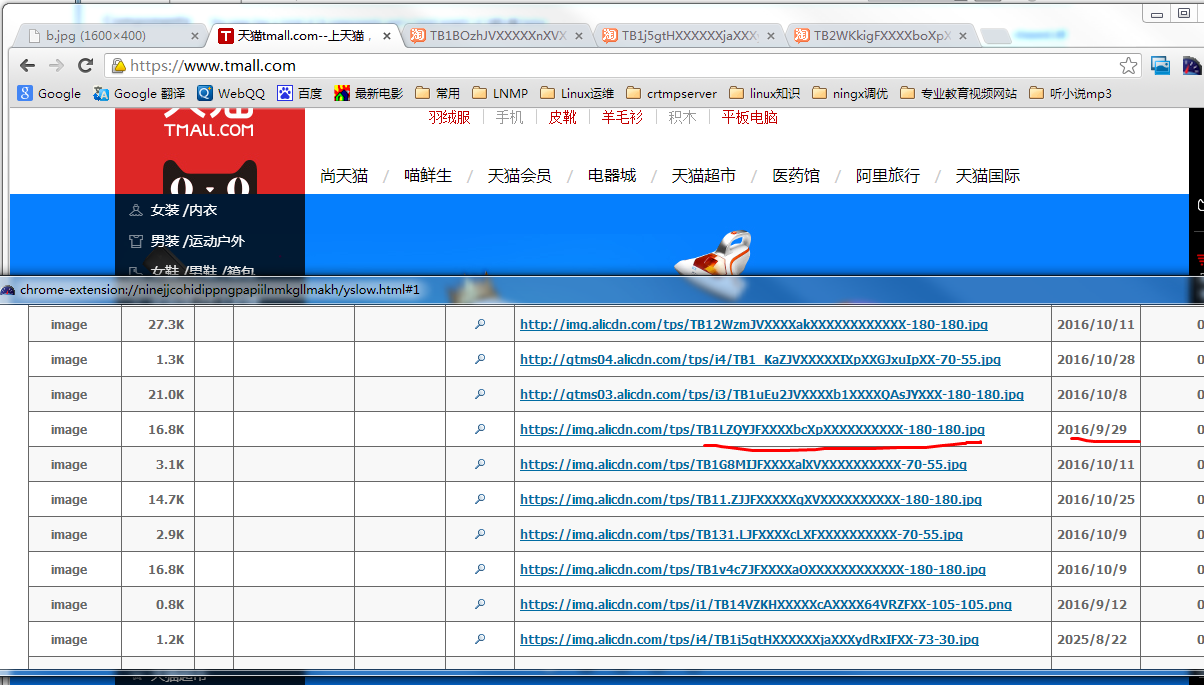
**有效期可以通过增加“<num><type>”子句进一步调整**

**ExpiresByType text/html "access plus 1 month 15 days 2 hours"**

**ExpiresByType image/gif "modification plus 5 hours 3 minutes"**

**缓存优化对企业来说是至关重要了的，不同的业务类型，对于缓存的类型和时间又各不相同，大家具体可以用YSlow对淘宝，京东，新浪，谷歌等站点进行测试，看看他们的缓存时间是多少，并思考下为什么那样设置。**

**查看天猫缓存时间：**



**注：天猫解决被缓存文件及时更新的方法：对缓存的对象改名**

**虽然缓存的设置有如下优点**

**1.缩短服务的响应时间**

**2.减轻服务器负担**

**3.减少网络带宽使用量，降低企业成本**

**但是他的缺点也是显而易见的，由于使用了缓存设置，导致被缓存的内容更新了，但是客户看到的却还是旧的。**

**如何解决被缓存文件及时更新这个问题呢？**

**第一种：缩短缓存时间例如：1天，不彻底牺牲性能**

**第二种：对缓存的对象改名。**

**15.3 开启长连接功能**

**15.3.1 apache的keepalive和keepalivetimeout  。打开长链接**

**KeepAlive Off/On 保持连接，会减少三次握手，但是会消耗内存，是否打开，取决于单位时间内是否进行多次连接(三次握手)，一个连接有多次请求的，建议打开,并适当调整KeepAliveTimeout时间**

**在APACHE的httpd.conf中，KeepAlive指的是保持连接活跃，如果将KeepAlive设置为On，那么来自同一客户端的请求就不需要再一次连接，避免每次请求都要新建一个连接而加重服务器的负担。**

**KeepAlive的连接活跃时间当然是受KeepAliveTimeOut限制的。如果第二次请求和第一次请求之间超过KeepAliveTimeOut的时间的话，第一次连接就会中断，再新建第二个连接。**

**所以，一般情况下，图片较多的网站应该把KeepAlive设为On。但是KeepAliveTimeOut应该设置为多少秒就是一个值得讨论的问题了。**

**如果KeepAliveTimeOut设置的时间过短，例如设置为1秒，那么APACHE就会频繁的建立新连接，当然会耗费不少的资源；反过来，如果KeepAliveTimeOut设置的时间过长，例如设置为300秒，那么APACHE中肯定有很多无用的连接会占用服务器的资源，也不是一件好事。**

**所以，到底要把KeepAliveTimeOut设置为多少，要看网站的流量、服务器的配置而定。**

**以下是我的配置：**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf**

**改： #Include /etc/httpd/extra/httpd-default.conf**

**为：Include /etc/httpd/extra/httpd-default.conf**

**# vim /etc/httpd/extra/httpd-default.conf**

**修改：**

**KeepAlive On**

**KeepAliveTimeout 5**

**注：如果配置文件中这两项，就自己修改一下。源码编译的有时没有这两个参数。**

**考虑到我的网站上有不少的图片，所以将KeepAlive设为On，一般的页面两次请求间隔不会超过30秒，所以这样设置，至尽运行状况良好。如果是艺术品网站， 需要看的时间长，那么就设置长一些。**

**MaxKeepAliveRequests**

**默认：100**

**一个建立好的Keep-Alive连接，允许发送的请求的个数。一旦建立连接，要么就是个数达到了断开，要么就是等KeepAliveTimeout时间到了断开连接。**

**MaxKeepAliveRequests指令限制了当启用KeepAlive时，每个连接允许的请求数量。如果将此值设为"0"，将不限制请求的数目。我们建议最好将此值设为一个比较大的值，以确保最优的服务器性能。"**

**这个数字的设置，必须考虑在一个时间段内，同一个用户访问你的服务会发多少请求。要结合KeepAliveTimeout参数来考虑。**

**举个例子，用户需要间隔时间不大于KeepAliveTimeout的时间内，连续请求10个文件，那么这个参数就应该设置成10，如果用户在连续时间里不断请求访问，则这个数值得设置得更多。否则就重新建立连接下载。一旦用户连续进行了10个请求后，并且这个用户肯定在完成这些请求后的5秒内不会再请求，甚至要在之后的很长时间后请求，那么这个KeepAliveTimeout时间就可以设置得很短，以便尽早断开这种用户，把资源让个其他用户。**

**TimeOut**

**默认：60 秒**

**"TimeOut指令用于设置Apache等待以下三种事件的时间长度：**

**1. 接受一个GET请求耗费的总时间。**

**2. POST或PUT请求时，接受两个TCP包之间的时间。**

**3. 应答时TCP包传输中两个ACK包之间的时间。**

**比如：apache需要把jsp文件传给后端tomcat服务器，而tomcat服务器关了，这时这个链接需要等待的超时时间，由TimeOut控制。**

**注：计时器在1.2版本之前的默认值为1200，而现在已经设置为300了，但对于绝大多数情况来说仍是足够的。**

**5.3.2 案例分析：**

**假设 KeepAlive 的超时时间为 10 秒钟，服务器每秒处理 50 个独立用户访问，那么系统中Apache的总进程数就是 10 \* 50 ＝ 500 个，如果一个进程占用 4M 内存，那么总共会消耗2G内存，所以可以看出，在这种配置中，相当消耗内存，但好处是系统只处理了 50次 TCP 的握手和关闭操作。**

**如果关闭 KeepAlive，如果还是每秒50个用户访问，如果用户每秒的并发请求数为3个，那么 Apache 的总进程数就是 50 \* 3 = 150 个，如果还是每个进程占用 4M 内存，那么总的内存消耗为600M，这种配置能节省大量内存，但是，系统处理了 150 次 TCP 的握手和关闭的操作，因此又会多消耗一些 CPU 资源。**

**总结：**

**1、如果内存和CPU都足够，开启和关闭KeepAlive对性能影响不大。**

**2、如果考虑服务器压力，如果同一个客户端对服务器会经常访问，建议开启KeepAlive。**

**参考数值：**

**KeepAlive On**

**KeepAliveTimeout 30**

**MaxKeepAliveRequests 100 #这个值一般不需要配置。**

**15.4 apache运行模式-prefork-worker运行模式介绍**

**15.4.1 apache不同运行模式调优。**

**Web服务器Apache目前一共有三种稳定的MPM（Multi-Processing Module，多进程处理模块）模式。**

**Prefork：进程模式**

**worker：线程模式**

**Event : 事件模式（2.4版本后开始稳定）**

**15.6 prefork运行模式详解**

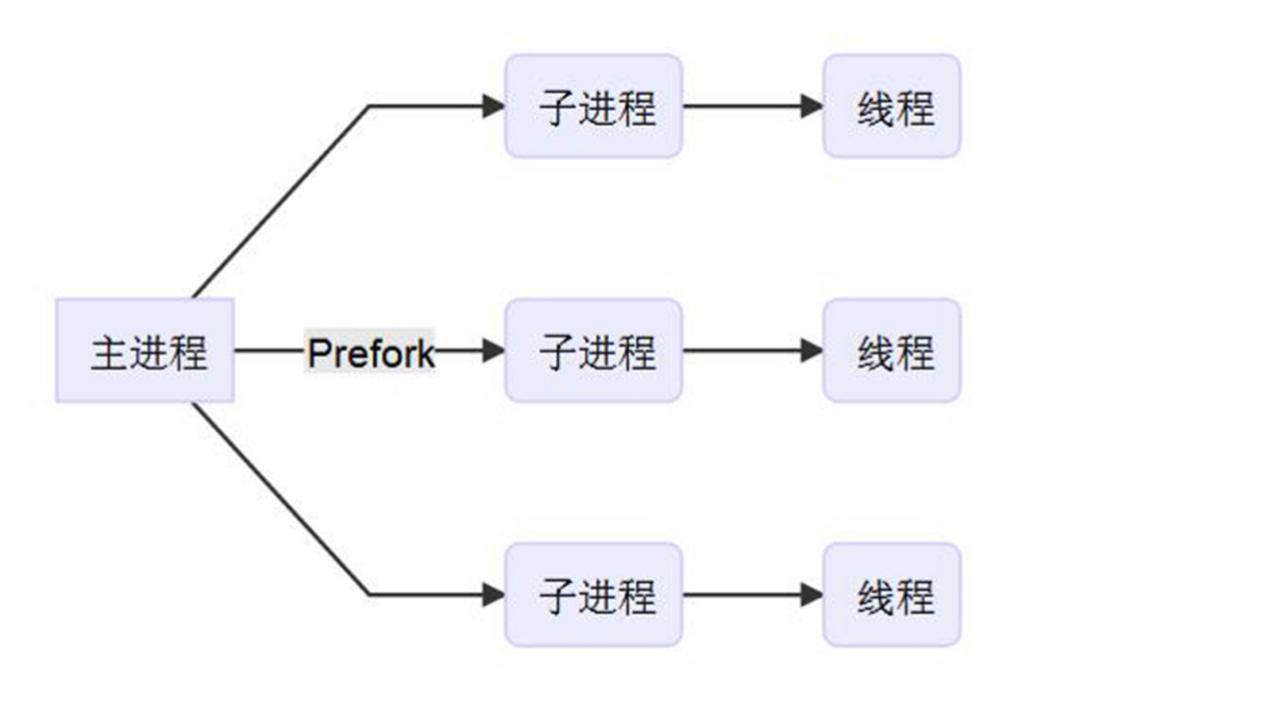
1. **Prefork MPM : Prefork MPM实现了一个非线程的、预派生的web服务器。它在Apache启动之初，就先预派生一些子进程，然后等待连接；可以减少频繁创建和销毁进程的开销，每个子进程只有一个线程，在一个时间点内，只能处理一个请求。这是一个成熟稳定，可以兼容新老模块，也不需要担心线程安全问题，但是一个进程相对占用资源，消耗大量内存，不擅长处理高并发的场景。**

**最重要的是将MaxRequestWorkers设置为一个足够大的数值以处理潜在的请求高峰，同时又不能太大，以致需要使用的内存超出物理内存的大小。**

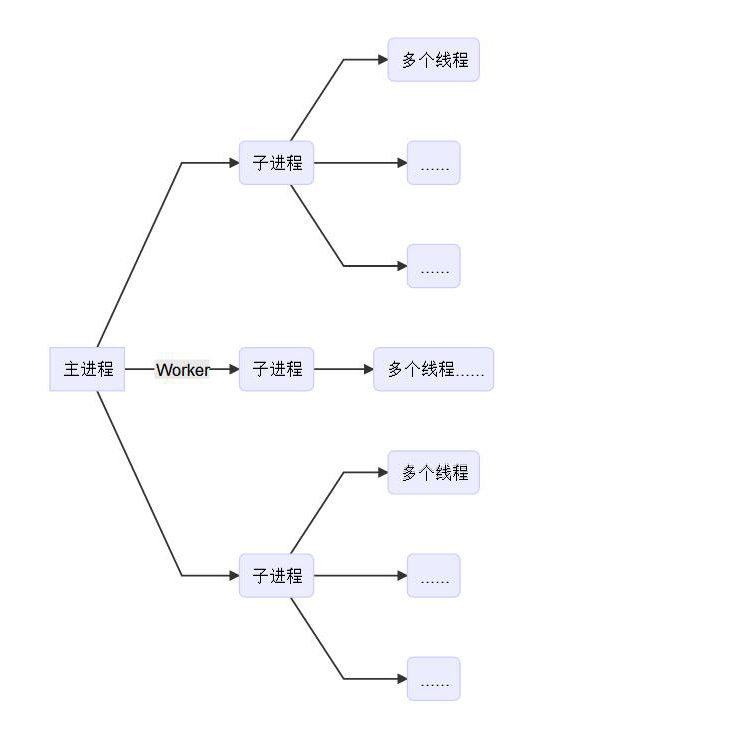
**注： Prefork  是基于多进程的模式。**

**优点：因为每个进程使用独立的内存空间，所以比较安全。一个进程坏了，不会影响其他进程。**

**缺点：占用的内存比较大。**



**2、Worker MPM :  和prefork模式相比，worker使用了多进程和多线程的混合模式，worker模式也同样会先预派生一些子进程，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一个线程来服务。线程比起进程会更轻量，因为线程是通过共享父进程的内存空间，因此，内存的占用会减少一些，在高并发的场景下会比prefork有更多可用的线程，表现会更优秀一些；另外，如果一个线程出现了问题也会导致同一进程下的线程出现问题，如果是多个线程出现问题，也只是影响Apache的一部分，而不是全部。由于用到多进程多线程，需要考虑到线程的安全了，在使用keep-alive长连接的时候，某个线程会一直被占用，即使中间没有请求，需要等待到超时才会被释放（该问题在prefork模式下也存在）。**



**注： Worker MPM**

**优点：可以处理海量请求，而系统资源的开销小。原因：一个进程中包括多个线程。多个线程之间可以共享内存，所以占用的内存资源比较少。如图：**

**缺点：不太安全。如果一个线程坏了。 整个进程都要坏了。另外存在keep-alive长连接占用资源时间过长**

**如何避免进程中某个线程坏了? 一个进程中所有线程完成一定数量的请求后，自动关闭，再重打开。就可以避免内存溢出等问题。 就像一个电脑开机时间长了，会卡，需要重启一下。**

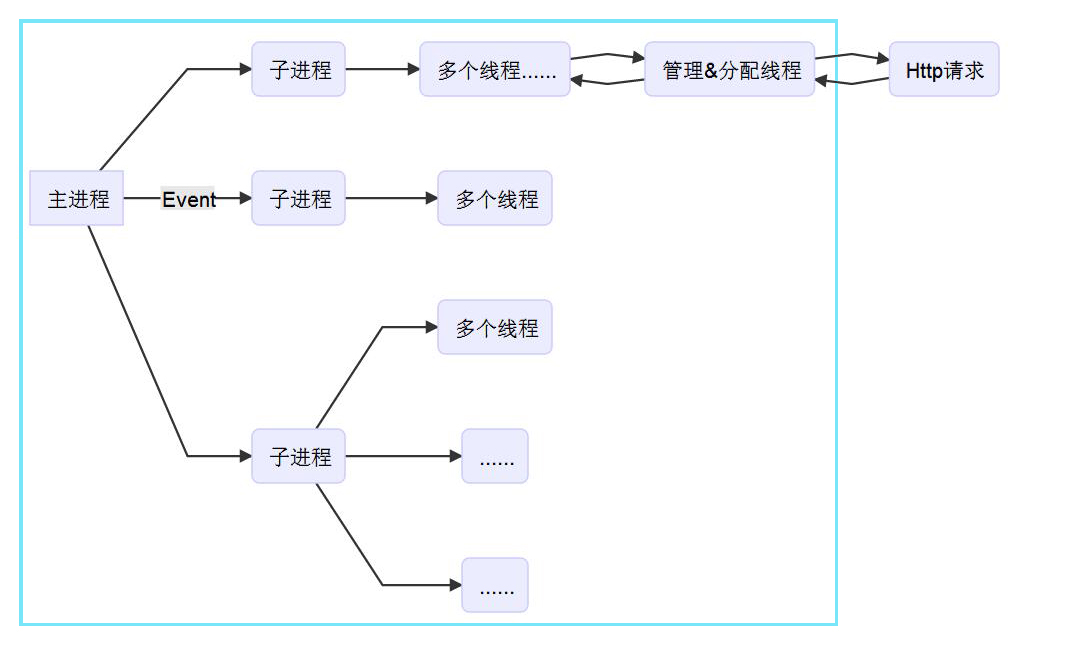
**进程包括线程的。 一个进程可以有一个或多个线程。**

**总结：　不管是Worker模式或是Prefork 模式，Apache总是试图保持一些备用的(spare)或者是空闲的子进程（空闲的服务线程池）用于迎接即将到来的请求。这样客户端就不需要在得到服务前等候子进程的产生。**

**这就是：预先派生进程或线程。**

**3、Event MPM：event模式是在2.4版本中才稳定发布的模式。这是Apache最新的工作模式，它和worker模式很像，不同的是在于它解决了keep-alive长连接的时候占用线程资源被浪费的问题，在event工作模式中，会有一些专门的线程用来管理这些keep-alive类型的线程，当有真实请求过来的时候，将请求传递给服务器的线程，执行完毕后，又允许它释放。这增强了在高并发场景下的请求处理。**

**当某个连接没有请求时，会主动关闭连接，在work模式下，必须等keep-alive超时，才可以释放。**



**在configure配置编译参数的时候，可以使用 --with-mpm=prefork|worker|event 来指定编译为那一种MPM，当然也可以用编译为三种都支持：--enable-mpms-shared=all，这样在编译的时候会在modules目录下自动编译出三个MPM文件的so，然后通过修改httpd.conf配置文件更改MPM**

**15.4.2 实战2： Apache prefork调优**

**案例场景： 当用户访问网站时，在客户端浏览器输入网址后长时间无响应，而一旦连接上之后，页面很快就打开了（因为配置了持久连接）。**

**排查：**

**登录上linux服务器后，使用netstat观察最大连接数稳定在257，查看apache配置文件中，prefork模式中，MaxRequestWorkers为 257， 这因为apache连接数明显不够用。**

**先查看apache的运行模式**

**查看命令：**

**[root@xuegod63 ~]# /usr/local/apache/bin/httpd -M | grep event**

**mpm\_event\_module (shared) #看到这个，说明是prefork多进程模式**

**注：查看模块：**

**httpd -l          //查看MPM模块**

**httpd -M          //查看DSO模块，由于mod\_mpm\_event.so，mod\_mpm\_prefork.so，mod\_mpm\_worker.so，此三个模块被做成DSO模块，所以使用httpd -M查看**

**或：**

**[root@xuegod63 src]# /usr/local/apache/bin/httpd -V**

**Server version: web/8.1.2-dev (Unix)**

**Server built: Feb 8 2018 21:29:56**

**Server's Module Magic Number: 20120211:36**

**Server loaded: APR 1.5.2, APR-UTIL 1.5.2**

**Compiled using: APR 1.5.2, APR-UTIL 1.5.2**

**Architecture: 64-bit**

**Server MPM: event**

**注：如果使用rpm安装的httpd，直接执行：**

**[root@xuegod63 ~]# httpd -M |grep prefork**

**mpm\_prefork\_module (shared)**

**[root@xuegod63 ~]# rpm -qf `which httpd` #httpd命令是apache服务器软件包安装的**

**httpd-2.4.10-15.el7.x86\_64**

**总结：**

**RHEL6/7系统自带的apache默认采用的是prefork进程模型；在编译apache源码时，如果不用--with-mpm显式指定某种MPM，prefork就是缺省的MPM；**

**httpd2.4 新特性**

**1）MPM支持在运行时装载**

**指定启用：**

**--enable-mpms-shared=all --with-mpm=event      //把所有支持的MPM都编译进来，但启用默认的是event**

**2) 支持event**

**3）异步读写**

**4) 在每模块及每目录上指定日志级别**

**5）每请求配置：<If> <Elseif>**

**6) 增强版的表达式分析器**

**7) 毫秒级的keepalive timeout ，使用ms指定为毫秒**

**8）支持主机名的虚拟主机不在需要NameVirtualHost指令**

**9) 支持使用自定义变量**

**新增一些模块：mod\_proxy\_fcgi（基于fcgi方式调用执行环境）**

**mod\_ratelimit（用于做速率限定）**

**mod\_request（对请求方法做限定）**

**mod\_remoteip（对远端IP做限定）**

**对于基于IP的访问做了修改，不在使用order,allow,deny这些机制；而是统一使用require进行**

**调整MPM模块则直接修改主配置文件中的LoadModule指定即可**

**#LoadModule mpm\_event\_module modules/mod\_mpm\_event.so**

**LoadModule mpm\_prefork\_module modules/mod\_mpm\_prefork.so**

**#LoadModule mpm\_worker\_module modules/mod\_mpm\_worker.so**

**2，对prefork模式进行优化。 修改apache 的httpd-mpm.conf 配置.**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf**

**改：449 #Include conf/extra/httpd-mpm.conf**

**为：Include conf/extra/httpd-mpm.conf**

**[root@xuegod63 ~]# vim /etc/httpd/extra/httpd-mpm.conf #第一次打开的时候默认配置是这样的。**

**改：**

**28 <IfModule mpm\_prefork\_module>**

**29 StartServers 5**

**30 MinSpareServers 5**

**31 MaxSpareServers 10**

**32 MaxRequestWorkers 250**

**33 MaxRequestsPerChild 0**

**34 </IfModule>**

**注：spare[英][speə(r)] 备用**

**为：**

**<IfModule mpm\_prefork\_module>**

**ServerLimit 3000**

**StartServers 50**

**MinSpareServers 50**

**MaxSpareServers 100**

**MaxRequestWorkers 3000**

**MaxRequestsPerChild 1000**

**</IfModule>**

**[root@xuegod63 ~]#systemctl restart httpd #重启服务**

**刚才配置文件中参数详解：**

**首先来看看在prefork模式下，apache各个参数的意义：**

**ServerLimit 是最大的进程数**

**MaxRequestWorkers 是最大的请求并发。**

**注：所以他们的关系是MaxRequestWorkers=ServerLimit\*进程的线程数。因为，在prefork模式下一个进程只有一个线程，并且一个进程对应一个连接。**

**所以这里配置成：MaxRequestWorkers=ServerLimit，MaxRequestWorkers不得大于ServerLimit参数。**

**如果做5000并发的web，需要多少内存？ 5000\*2M/0.8/1024(转G)=12.2G 。 服务器大概需要14G -16G 内存。**

**ServerLimit的大小，取决于你系统的资源，每个apache进程默认大约占用2M内存，基本可以按照这个公式来计算：最大内存\*80%/2M=ServerLimit。**

**注：一个apache进程实际使用内存大小和处理的请求数有关。即和MaxRequestsPerChild 这个值有关。（MaxRequestsPerChild，每个子进程在其生存期内允许处理的最大请求数量）。**

**15.4.3 StartServers 50 启动时默认启动的 进程数**

**这个参数默认是5，因为apache会通过自动启动新进程来增加响应服务的进程数，这个值不做调整的也是可以的，会由默认的5增加到满足服务的进程数，但是会出现开始启动后，突然后有大并发访问时，因为进程数太小，出现卡住的现象。**

**例：查看进程数：**

**[root@xuegod63 ~]# ps -axu | grep httpd | wc -l**

**Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ**

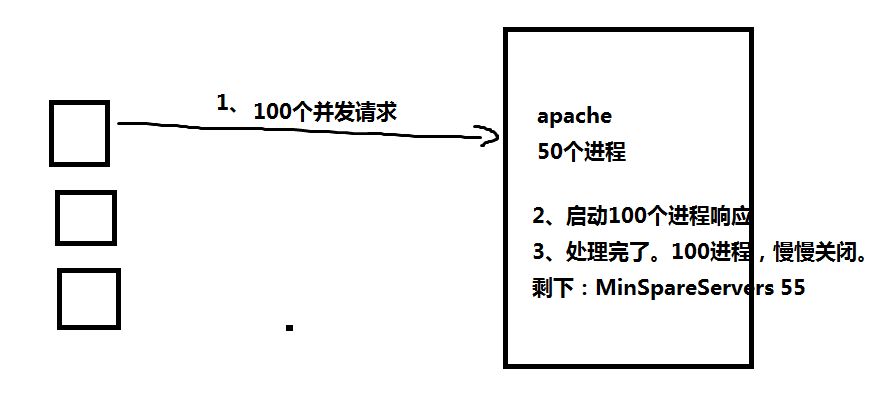
**52**

**注：所以不要一次启动太多的apache进程，只启动足够用的进程即可。其他增加的流量，apache会自动调整进程数，直到MaxRequestWorkers参数限定的范围。**

**15.4.4 MinSpareServers 55 最小空闲进程**

**Spare ：[speə(r)] 备用**

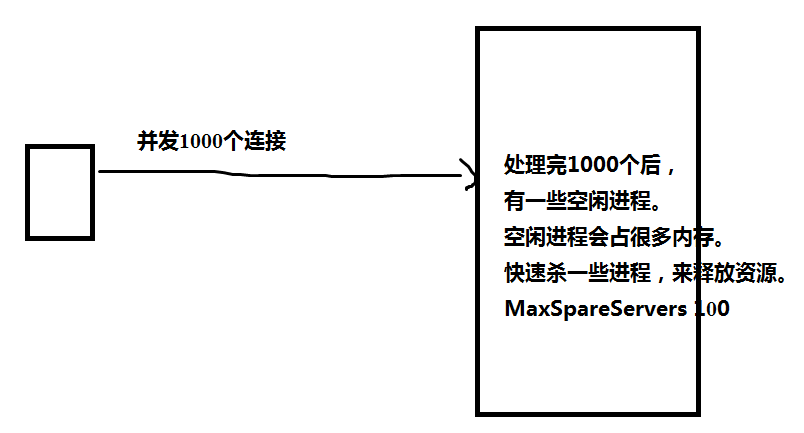
**MinSpareServers指令设置空闲子进程的最小数量。**

****

**MinSpareServers指令设置空闲子进程的最小数量。所谓空闲子进程是指没有正在处理请求的子进程。如果当前空闲子进程数少于MinSpareServers ，那么Apache将以第一秒一个，第二秒两个，第三秒四个，按指数递增个数的速度产生新的子进程。如此按指数级增加创建的进程数，最多达到每秒32个，直到满足 MinSpareServers设置的值为止；这就是预派生（prefork）的由来；这种模式可以不必在请求到来时再产生新的进程，从而减小了系统开销以增加性能；**

**15.4.5 MaxSpareServers 100 最大空闲进程**

**MaxSpareServers指令设置空闲子进程的最大数量。所谓空闲子进程是指没有正在处理请求的子进程。如果当前有超过MaxSpareServers数量的空闲子进程，那么父进程将杀死多余的子进程。**

****

**可以调整MinSpareServers 和MaxSpareServers这两个参数，但是这两个参数的值不能设得太大，否则apache进程太多，会导致内存占用太多。**

**注：在一台压力大(并发访问2000)的服务器上，MaxSpareServers这个值设置的是200。保留最大并发数的10分之一。**

**设置了这个值的好处是不会有太多的空闲的进程在消耗资源，关闭空闲apache进程的同时，会释放内存，从而减少系统资源消耗。**

**15.4.6 MaxRequestsPerChild 1000**

**"MaxRequestsPerChild指令设置每个子进程在其生存期内允许处理的最大请求数量。到达MaxRequestsPerChild的限制后，子进程将会结束。如果MaxRequestsPerChild为"0"，子进程将永远不会结束。**

**将MaxRequestsPerChild设置成非零值有两个好处：**

**\* 可以防止(偶然的)内存泄漏无限进行，从而耗尽内存。**

**\* 给进程一个有限寿命，从而有助于当服务器负载减轻的时候减少活动进程的数量。**

**注：当KeepAlive 为On， 开启长链接时，发送的请求，在MaxRequestsPerChild里面只算一个，不管这个连接发送了多少个请求。**

**注意：对于KeepAlive链接，只有第一个请求会被计数。事实上，它改变了每个子进程限制最大链接数量的行为。**

**也就是说实际上这个时候子进程最大连接数=MaxRequestsPerChild\*MaxKeepAliveRequests**

**例：**

**MaxRequestsPerChild 1000**

**MaxKeepAliveRequests 200 #一个长连接可以处理请求数为200**

**每子进程最大连接数=1000\*200=20万次请求。 注：这个进程处理的数量太大了。**

**改为：**

**MaxRequestsPerChild 100**

**MaxKeepAliveRequests 20 #一个长连接可以处理请求数为20**

**每子进程最大连接数=100\*20=2千次请求。**

**注：所以在开启KeepAlive后，需要同时设置MaxRequestsPerChild和MaxKeepAliveRequests，确保每个apache进程在服务一定请求数后会关闭，重新开启新的子进程，避免apache进程异常导致的内存泄露和资源占用。**

**15.4.7 实战：如何跟据硬件来配置连接数？**

**连接数理论上是越大越好，但是得根据硬件，服务器的CPU，内存，带宽等因素，查看当前的apache连接数。**

**[root@xuegod63 ~]# ps -aux | grep httpd | wc -l**

**52 注：总进程数为52-2**

**查看httpd 占用内存的平均数: 使用ps查看RSS列，每个进程占用的内存。**

**[root@xuegod63 ~]# ps -axu | grep http | awk '{print $6}' #单位是K，现在一个使用1.6M左右**

**1636**

**1636**

**1636**

**1636**

**1636**

**1636**

**1636**

**1636**

**1636**

**或：**

**[root@xuegod63 ~]# ps aux | grep http |awk '{sum += $6;n++};END{print sum/n}'**

**1638.92**

**注：一个进程使用1.6M左右内存。**

**15.4.8 如何设置最大链接数：**

**计算后要减去服务器系统本身所需要的资源。比如内存2G，减去500M留给服务器，还有1.5G，那么可得到最大连接数：1500/1.6=930左右。**

**根据情况修改后的http-mpm.conf的prefork的配置后为：**

**<IfModule mpm\_prefork\_module>**

**StartServers 50**

**MinSpareServers 50**

**MaxSpareServers 100**

**ServerLimit 1000**

**MaxRequestWorkers 1000**

**MaxRequestsPerChild 100**

**</IfModule>**

**注：这里重点介绍下ServerLimit，必须放到MaxRequestWorkers前，值要等于MaxRequestWorkers。**

**重启apache，再打开网站看看是否还会有慢的问题了。**

**[root@xuegod63 ~]# systemctl restart httpd**

**测试：**

**动态观察apache的最大连接数:**

**[root@xuegod63 ~]# watch -n 1 "pgrep httpd|wc -l"**

**Every 1.0s: pgrep httpd|wc -l Wed Sep 2 15:24:08 2015**

**51 #有51个进程**

**15.4.9 生产环境配置实例1：物理是内存4G**

**<IFModule mpm\_prefork\_module>**

**StartServers 20**

**MinSpareServers 20**

**MaxSpareServers 30**

**ServerLimit 2000**

**MaxRequestWorkers 2000**

**MaxRequestsPerChild 5000 #没有开keepalive是5000，如果开启keepalive就配置成500**

**</IFModule>**

**15.4.10 实战2： apache worker模拟性能优化**

**Apache2.0的性能方面的改进最明显的变化就在于worker；**

**优点：内存占用比prefork模式低，适合高并发高流量HTTP服务。**

**缺点：假如一个线程崩溃，整个进程就会连同其他任何线程一起“死掉”。由于线程共享内存空间，所以一个程序在运行时必须被系统识别为“每个线程都是安全的”。服务稳定性不如prefork模式。**

**[root@xuegod63 ~]# /usr/local/apache/bin/httpd -M |grep worker //查看是否使用worker模块，没有输出，则说明没有使用**

**[root@xuegod63 ~]# ls /usr/local/apache/modules/\* |grep worker //查模块目录，模块已存在**

**/usr/local/apache/modules/mod\_mpm\_worker.so**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf //修改apache配置文件**

**#LoadModule mpm\_event\_module modules/mod\_mpm\_event.so**

**#LoadModule mpm\_prefork\_module modules/mod\_mpm\_prefork.so**

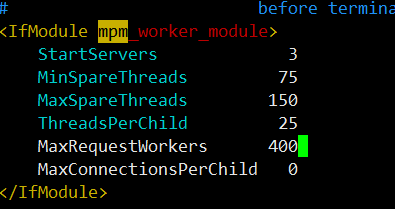
**LoadModule mpm\_worker\_module modules/mod\_mpm\_worker.so //只开启worker项，如此项没有，则手动添加即可**

**[root@xuegod63 ~]# systemctl restart httpd //重启apache**

**[root@xuegod63 ~]# vim /etc/httpd/extra/httpd-mpm.conf**

**[root@xuegod63 ~]# vim /etc/httpd/extra/httpd-mpm.conf**

**改：**



**StartServers 2**

**MaxRequestWorkers 150000**

**MinSpareThreads 25**

**MaxSpareThreads 75**

**ThreadsPerChild 25**

**MaxRequestsPerChild 0**

**[root@xuegod63 ~]# systemctl restart httpd //重启apache**

**配置参数说明：**

**<IFModule mpm\_worker\_module>**

**StartServers 2 #最初建立的子进程**

**MaxRequestWorkers 150000 # MaxRequestWorkers，apache同时最多能支持150000个并发访问，超过的要进入队列等待，其大小由ServerLimit和ThreadsPerChild的乘积决定。这个150000指的是所有子进程中的线程总数。一般要把这个值配置的大一些**

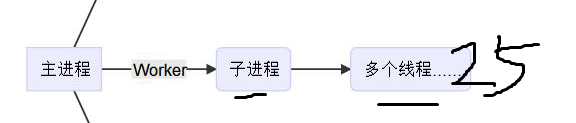
**MinSpareThreads 25 #基于整个服务器监视的最小空闲线程数，如果空闲的线程小于设定值，apache会自动建立线程，如果服务器负载大的话，可以考虑加大此参考值。**

**MaxSpareThreads 75 #基于整个服务器监视的最大空闲线程数，如果空闲的线程大于设定值，apache会自动kill掉多余的线程，如果服务器负载大的话，可以考虑加大此参考值。**

**ThreadsPerChild 25 #每个子进程包含固定的线程数，此参数在worker模式中，是影响最大的参数，ThreadsPerChild的最大缺省值是64，如果负载较大，64是不够的.**

**MaxRequestsPerChild 0 #每个子进程可以支持的请求数，这要设置为0，因为一个进程关闭，所有的线程也都关了。**

**注：child 子进程**

****

**ServerLimit服务器允许配置进程数的上限。  
 ThreadLimit每个子进程可以配置的线程上限**

**常用配置参考：**

**生产环境配置实例：**

**<IFModule mpm\_worker\_module>**

**StartServers 5**

**MaxRequestWorkers 9600**

**ServerLimit 64**

**MinSpareThreads 25**

**MaxSpareThreads 500**

**ThreadLimit 200**

**ThreadsPerChild 150**

**MaxRequestsPerChild 0**

**</IFModule>**

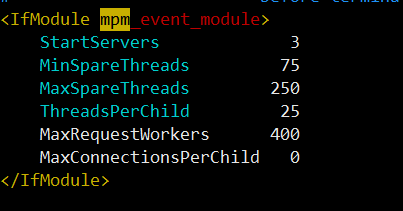
**此服务器配置： 最多进程数：64个； 最多线程数（最大并发数） 64\*150=9600 ；不可能超过64\*200=12800**

**案例：**

**如果MaxRequestsPerChild 30000 这样配置会有什么结果？**

**结果：一但到30000次请求 ， 那150个线程都关了，会出大量闪断情况。**

**event模式：了解一下**



**<IfModule mpm\_event\_module>   
StartServers 3   
MinSpareThreads 75   
MaxSpareThreads 250   
ThreadsPerChild 25   
MaxRequestWorkers 400   
MaxConnectionsPerChild 1000   
</IfModule>**

**总结：**

**Apache httpd 能更好的为有特殊要求的站点定制。例如，要求 更高伸缩性的站点可以选择使用线程的 MPM，即 worker 或 event； 需要可靠性或者与旧软件兼容的站点可以使用 prefork。**

**15.7 rewrite-禁止网站下某个目录执行PHP文件-apache调优总结**

**15.7.1 Rewrite规则简介：**

**Rewirte主要的功能就是实现URL的跳转，它的正则表达式是基于Perl语言。可基于服务器级的(httpd.conf)和目录级的 (.htaccess)两种方式。如果要想用到rewrite模块，必须先安装或加载rewrite模块。**

**安装Rewirte模块两种方式：**

**方法一：是编译apache的时候就直接 安装rewrite模块。**

**方法二：编译apache时以DSO模式安装apache,然后再利用源码和apxs来安装rewrite模块。**

**15.7.2 基于服务器级的(httpd.conf)有两种方法：**

**方法1：在httpd.conf的全局下 直接利用RewriteEngine on来打开rewrite功能;**

**方法2：在局部里利用RewriteEngine on来打开rewrite功能,下面将会举例说明，需要注意的是,必须在每个virtualhost里用RewriteEngine on来打开rewrite功能。否则virtualhost里没有RewriteEngine on它里面的规则也不会生效。**

**基于目录级的(.htaccess),要注意一点那就是必须打开此目录的FollowSymLinks属性且在.htaccess里要声明RewriteEngine on。**

**15.7.3 Apache mod\_rewrite规则重写的标志一览:**

**1) R[=code](force redirect) 强制外部重定向**

**强制在替代字符串加上http://thishost[:thisport]/前缀重定向到外部的URL.如果code不指定，将用缺省的302 HTTP状态码。**

**2) F(force URL to be forbidden)禁用URL,返回403HTTP状态码。**

**3) G(force URL to be gone) 强制URL为GONE，返回410HTTP状态码。**

**4) P(force proxy) 强制使用代理转发。**

**5) L(last rule) 表明当前规则是最后一条规则，停止分析以后规则的重写。**

**6) N(next round) 重新从第一条规则开始运行重写过程。**

**7) C(chained with next rule) 与下一条规则关联**

**如果规则匹配则正常处理，该标志无效，如果不匹配，那么下面所有关联的规则都跳过。**

**8) T=MIME-type(force MIME type) 强制MIME类型**

**9) NS (used only if no internal sub-request) 只用于不是内部子请求**

**10) NC(no case) 不区分大小写**

**11) QSA(query string append) 追加请求字符串**

**12) NE(no URI escaping of output) 不在输出转义特殊字符**

**例如：RewriteRule /foo/(.\*) /bar?arg=P1\%3d$1 [R,NE] 将能正确的将/foo/zoo转换成/bar?arg=P1=zed**

**13) PT(pass through to next handler) 传递给下一个处理**

**例如：**

**RewriteRule ^/abc(.\*) /def$1 [PT] # 将会交给/def规则处理**

**Alias /def /ghi**

**14) S=num(skip next rule(s)) 跳过num条规则**

**15) E=VAR:VAL(set environment variable) 设置环境变量**

**15.7.4 mod\_rewrite模块检查及安装**

**[root@xuegod63 ~]# /usr/local/apache/bin/httpd -M |grep rewrite //没有加载rewrite模块**

**[root@xuegod63 ~]# ls /usr/local/apache/modules/\* |grep rewrite**

**/usr/local/apache/modules/mod\_rewrite.so**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf**

**修改：**

**161 #LoadModule rewrite\_module modules/mod\_rewrite.so**

**改为：**

**LoadModule rewrite\_module modules/mod\_rewrite.so**

**[root@xuegod63 ~]# systemctl restart httpd**

**[root@xuegod63 ~]# /usr/local/apache/bin/httpd -M |grep rewrite**

**rewrite\_module (shared)**

**15.7.5 实战举例：**

**实现client请求的主机前缀不是www.xuegod.com和192.168.1.63都跳转到主机前缀为http://www.xuegod.com.cn 。例如当用户在地址栏写入http://xuegod.com.cn和bbs.xuegod.com直接跳转到http://www.xuegod.com.cn登录网站。**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf**

**在：**

**172 User daemon**

**173 Group daemon**

**之后插入:**

**RewriteEngine on**

**RewriteCond %{HTTP\_HOST} !^www.xuegod.com [NC]**

**RewriteCond %{HTTP\_HOST} !^192.168.1.63 [NC]**

**RewriteCond %{HTTP\_HOST} !^$**

**RewriteRule ^/(.\*) http://www.xuegod.com.cn/ [L]**

**注释：**

**RewriteEngine on #打开rewirte功能**

**RewriteCond %{HTTP\_HOST} !^www.xuegod.com [NC] #声明Client请求的主机中前缀不是www.xuegod.com,[NC]的意思是忽略大小写**

**RewriteCond %{HTTP\_HOST} !^192.168.1.63 [NC] #声明Client请求的主机中前缀不是192.168.1.63,[NC]的意思是忽略大小写**

**RewriteCond %{HTTP\_HOST} !^$ #声明Client请求的主机中前缀不为空,[NC]的意思是忽略大小写**

**RewriteRule ^/(.\*) http://www.xuegod.com.cn/ [L]**

**#含义是如果Client请求的主机中的前缀符合上述条件，则直接进行跳转到http://www.xuegod.com.cn/,[L]意味着立即停止重写操作，并不再应用其他重写规则。这里的.\*是指匹配所有URL中不包含换行字符，()括号的功能是把所有的字符做一个标记，以便于后面的应用. 就是引用前面里的(.\*)字符。**

**添加hosts记录文件：**

**[root@xuegod63 ~]# vim /etc/hosts**

**192.168.1.63 www.xuegod.com**

**192.168.1.63 www.xuegod.com.cn**

**192.168.1.63 xuegod.com.cn**

**192.168.1.63 xuegod.com**

**192.168.1.63 bbs.xuegod.com**

**重启服务:**

**[root@xuegod63 ~]# systemctl restart httpd**

**测试：**

**http://www.xuegod.com/ #可以正常访问**

**http://192.168.1.63/ #可以正常访问**

**http://xuegod.com.cn/ #直接跳转到http://www.xuegod.com.cn/**

**http://bbs.xuegod.com/ #直接跳转到http://www.xuegod.com.cn/**

**可以实现跳转说明成功。**

**15.7.6 实例二.将输入 bbs.test.com 的域名时跳转到www.test.com**

**root@xuegod63 ~]# vim /etc/httpd/httpd.conf**

**改：**

**RewriteEngine on**

**RewriteCond %{HTTP\_HOST} !^www.xuegod.com [NC]**

**RewriteCond %{HTTP\_HOST} !^192.168.1.63 [NC]**

**RewriteCond %{HTTP\_HOST} !^$**

**RewriteRule ^/(.\*) http://www.xuegod.com.cn/ [L]**

**为：**

**RewriteEngine on**

**RewriteCond %{HTTP\_HOST} ^bbs.test.com [NC]**

**RewriteRule ^/(.\*) http://www.test.com/ [L]**

**[root@xuegod63 ~]# vim /etc/hosts #hosts文件中追加**

**192.168.1.63 bbs.test.com**

**192.168.1.63 www.test.com**

**[root@xuegod63 ~]# systemctl restart httpd**

**测试：**

**http://bbs.test.com/ 直接转到http://www.test.com/**

**15.7.7 禁止目录浏览**

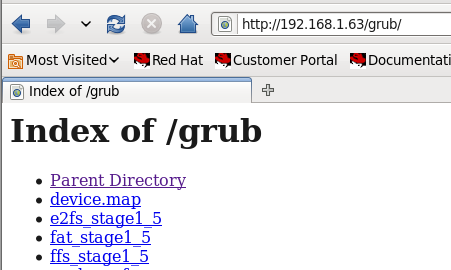
**由于开启目录浏览会让我们整个目录下的内容全部都暴露到外面，因此我们必须要禁止目录浏览功能。当然一些目录开放给客户做下载的，可以忽略此项优化。**

**我们通过修改apache主配置文件httpd.conf中的<Directory></Directory>标签内的Options选项参数来实现禁用目录浏览。**

**先复制一些目录：**

**[root@xuegod63 ~]# cp -r /boot/grub/ /usr/local/apache/htdocs/**

**测试：http://192.168.1.63/grub/**



**现在禁止访问：**

**[root@xuegod63 ~]# vim /etc/httpd/httpd.conf #找到根目录中的**

**<Directory "/usr/local/apache/htdocs">**

**。。。**

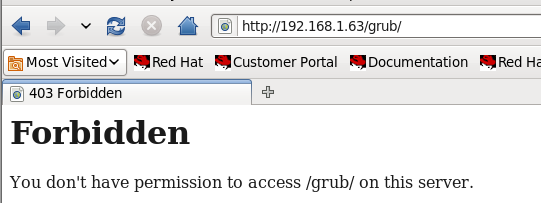
**改： Options Indexes FollowSymLinks**

**为： Options FollowSymLinks**

**注：直接删除Indexes 此选项即可**

**[root@xuegod63 ~]# systemctl restart httpd**

**测试：http://192.168.1.63/grub/**



**15.7.10 禁止PHP解析网站中某个目录中的php文件**

**企业的站点有时会提供用户进行上传操作，而用户上传文件的存放目录，我们是不能给php的解析权限的，否则会对apache服务和系统造成危害。**

**实例：**

**[root@xuegod63 ~]# mkdir /usr/local/apache/htdocs/data #data目录是我们需要保护目录**

**[root@xuegod63 ~]# vim /usr/local/apache/htdocs/data/a.php**

**<?php**

**phpinfo();**

**?>**

**测试：**

**http://192.168.1.63/data/a.php #可以正常解析访问**

**现在配置：**

**[root@xuegod63 ~]# vim /etc/httpd/conf/httpd.conf**

**在：**

**210 <Directory />**

**211 AllowOverride none**

**212 Require all denied**

**213 </Directory>**

**之后，添加以下内容：**

**<Directory "/usr/local/apache/htdocs/data" >**

**<Files ~ ".php">**

**AllowOverride none**

**Require all denied**

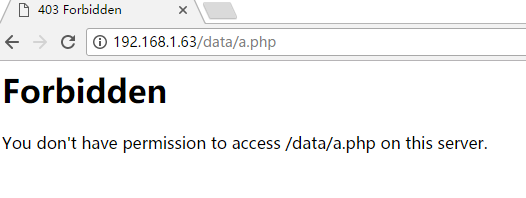
**</Files>**

**</Directory>**

**[root@xuegod63 ~]# systemctl restart httpd**

**测试：**

**http://192.168.1.63/data/a.php #发现执行不成了**



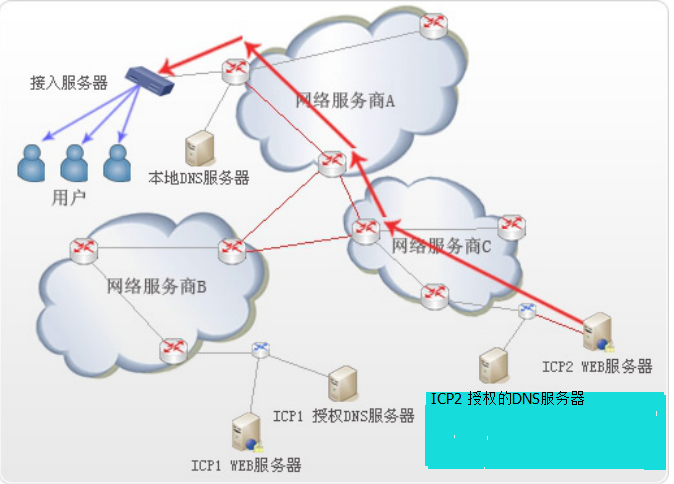
**15.7.13 优化系统内核参数。 参考系统调优**

**15.7.14 优化mysql数据库。 参考mysql调优**

**15.7.15 使用CDN做网站加速**

**简单地说，就是通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到最接近用户的缓存服务器内。通过DNS负载均衡的技术，判断用户来源就近访问cache服务器取得所需的内容，杭州的用户访问接近杭州服务器上的内容，北京访问接近北京服务器上的内容。这样可以有效减少数据在网络上传输的事件，提高速度。把静态内容发布到CDN减少了用户影响时间20%或更多。**

**CDN技术示意图：**



**国内有名的CND公司：网宿，蓝汛（chinacache），快网**

**15.7.16 apache网站架构优化**

**好的网站架构是网站性能强大关键，更是网站安全的关键。**

**在生产环境中建议将程序页面服务器、图片附件服务器和上传服务器三者的功能尽量分离。**

**分离方法：**

**1、分离最佳方式是分别使用独立的服务器（需要程序支持）**

**2、次选方案在前端负载均衡器通过haproxy/nginx来根据用户请求的目录或扩展名来对后端的服务器发出请求。**

**例如：请求http://www.xuegod.cn/a/b.jpg就抛给图片服务器（CDN最好），这里是根据扩展名.jpg分发**

**请求http:// /www.xuegod.cn /upload/login.php就抛给apache服务器，这里是根据URL路径分发。**

**均不符合上面两个要求的，默认就都是抛给主web服务器。**