# 源码官方下，补丁及时打

正如我们在apache部署篇讲到的我们需要对下载的源码包进行验证一样，我们在挑选apache源码安装包的时候，一定要去官网，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。

另外为了apache的安全性和性能考虑，我们一定要多多关注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。

# 屏蔽apache版本等敏感信息

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@c64-web /]# grep "httpd-default.conf" /usr/local/apache/conf/httpd.conf

#Include conf/extra/httpd-default.conf

[root@c64-web /]# sed -i 's#\#Include conf/extra/httpd-default.conf#Include conf/extra/httpd-default.conf#g' /usr/local/apache/conf/httpd.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@jincon /]# grep Server conf/extra/httpd-default.conf|grep -v "#"

ServerTokens full

ServerSignature Off

修改为：

ServerTokens Prod

ServerSignature Off

经过上面的修改，当你在curl -I www.jincon.com的时候，还是会出现下面的信息

Server: Apache

彻底让版本等敏感信息消失

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

#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 #

上述列出的行，我已经给出了注释，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。

# 更改apache的默认用户

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。

[root@c64-web /]# useradd -M -s /sbin/nologin apache 创建apache用户

编辑apache配置文件，修改默认的用户。

[root@c64-web /]# vim /usr/local/apache/conf/httpd.conf

User apache #更改默认的daemon用户为apache用户

Group apache #更改默认的daemon用户为apache用户

# Apache目录及文件权限设置

在生产环境的网站架构中，我们应把资源文件，例如用户上传的图片及附件等和程序做好分离，最好是把上传程序也分离开来。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root，权限是755，文件的权限为644。

lrwxr-xr-x 1 root root 23 11月 5 02:04 apache -> /usr/local/apache-2.4.6

drwxr-xr-x 14 root root 4096 11月 5 12:37 apache-2.4.6

并且，在对日志的授权商，我们要将属主和属组都设置为root，权限设置为700。

drwx------ 2 root root 4096 11月 5 02:46 logs

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，所以这里设置700是不影响日志记录了。这也是日志记录的最安全的方法。

# 配置cronolog进行日志轮询

由于apache自带的日志轮询工具rotatelogs，据专家说在进行日志切割时容易丢日志，因此我们通常使用cronolog进行日志轮询。

1、下载并安装cronolog

[root@jincon /]# cd /server/tools/

[root@jincon tools]# wget http://cronolog.org/download/cronolog-1.6.2.tar.gz

[root@jincon tools]# tar zxf cronolog-1.6.2.tar.gz

[root@jincon tools]# cd cronolog-1.6.2

[root@jincon cronolog-1.6.2]# ./configure

[root@jincon cronolog-1.6.2]# make&&make install

2、配置apache使用cronolog

由于实验用的apache开启了虚拟主机功能，所以以下范例配置都在虚拟主机中进行。

[root@jincon cronolog-1.6.2]# vim /usr/local/apache/httpd/extra/httpd-vhosts.conf

将配置文件中的CustomLog和ErrorLog替换为下面的(由于我们要对多站点，所以这里要替换两个虚拟主机的，并且将日志存放名字做有效的区分才好)

CustomLog "|/usr/local/sbin/cronolog /app/logs/sunsky\_access\_%Y%m%d.log" combined

ErrorLog "|/usr/local/sbin/cronolog /app/logs/sunsky\_error\_%Y%m%d.log"

更多日志格式参考：

按天轮询(生产环境常见用法，推荐使用)：

CustomLog "|/usr/local/sbin/cronolog /app/logs/access\_www\_%Y%m%d.log" combined

按小时轮询(生产环境较常见用法)：

CustomLog "|/usr/local/sbin/cronolog /app/logs /access\_www\_ %Y%m%d%H.log" combined

# 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：

第一种：在apache的主配置文件httpd.conf中的标签内添加ErrorDocument配置。

[root@jincon /]# vi /usr/local/apache/conf/httpd.conf

<Directory "/www/html">

AllowOverride None

Options None

Require all granted

ErrorDocument 404 /404.html #将404错误跳转到/www/html下的404.html页面上

</Directory>

第二种方法：在apache的虚拟主机配置文件httpd-vhost.conf中的中添加ErrorDocument配置。

<VirtualHost \*:80>

ServerAdmin 274546888@qq.com

DocumentRoot "/www/html"

ServerName www.jincon.com

ServerAlias jincon.com

CustomLog "|/usr/local/sbin/cronolog /app/logs/www\_access\_%Y%m%d.log" combined

ErrorLog "|/usr/local/sbin/cronolog /app/logs/www\_error\_%Y%m%d.log"

ErrorDocument 404 http://www.jincon.com ; #将404错误跳转到http://www.jincon.com 页面上

</VirtualHost>

从上面的两种解决方案，我们可以看出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。若跳转为未链接，则不影响。

# 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，所以的文本内容都应该能被gzip压缩，例如：html(php)，js，css，xml，txt等。特殊情况就是像一些首页上有很多广告投放的js代码，由于需要每次加载进而进行来访信息统计，所以这些广告代码拥有者网站的js不会经过gzip压缩，

mod\_deflate模块，在部署的时候已经编译安装进去了。

那么如果没有安装，如何安装呢?并且如果不知道到是否安装了，如何查看呢?

* 1、mod\_deflate模块检查及安装

由于apache2.2.x和2.4.x这两个版本在检查mod\_deflate模块是否安装方法方法是不同的。最后，我无奈做了很多测试，下面建议大家这样来检查。

[root@c64-web /]# /usr/local/apache/bin/apachectl -M | grep deflate

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

[root@c64-web /]# /usr/local/apache/bin/apachectl -M | grep deflate

deflate\_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

[root@c64-web /]# cd /server/tools/httpd-2.4.6/modules/filters/ #切到apache源码包mod\_deflate所在的目录下

[root@c64-web filters]# /usr/local/apache/bin/apxs -c -i -a mod\_deflate.c #以dso的方式编译安装到apache中

[root@c64-web filters]# ll /usr/local/apache/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

-rwxr-xr-x 1 root root 76697 11月 5 07:50 /usr/local/apache/modules/mod\_deflate.so

扩展：使用DSO方式安装，apxs后跟的参数详解

1. -c 此选项表明需要执行编译操作。它首先会编译c源程序(.c)files为对应的目标代码文件(.c)，然后连接这些目标代码和files中其余的目标代码文件(.c和.a)，以生成动态共享对象dsofile。如果没有指定-s选项，则此输出文件名由files中的第一个文件名推测得出，也就是默认 mod\_name.so。

2. -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。

3. -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。

4. 压缩模块配置

* 2、配置压缩参数

注意如果我们是编译安装时已经编译进去的，此时我们需要先将httpd.conf主配置文件中

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。

我们需要在虚拟机的中添加如下配置即可实现压缩：

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

SetOutputFilter DEFLATE #启用压缩

AddOutputFilterByType DEFLATE text/html text/plain text/xml #仅压缩限制特定的MIME类型文件：

AddOutputFilterByType DEFLATE application/javascript

AddOutputFilterByType DEFLATE text/css

AddOutputFilterByType DEFLATE image/gif image/png image/jpe image/swf image/jpeg image/bmp

#DeflateFilterNote ratio #在日志中放置压缩率标记，下面是记录日志的，这个功能一般不用

#LogFormat '"%r" %{outstream}n/%{instream}n (%{ratio}n%%)' deflate

#CustmLog logs/deflate\_log.log deflate

下面是朋友公司的另外一种压缩配置，有兴趣的朋友们也可以操作操作。

DeflateCompressionLevel 9

SetOutputFilter DEFLATE

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

#LogFormat '"%r" %{outstream}n/%{instream}n (%{ratio}n%%)' deflate #声明日志类型

#CustomLog logs/deflate\_log.log deflate #声明日志类型

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

执行上面的添加后，我们对apache进行优雅重启：

/usr/local/apache/bin/apachectl graceful

下面访问我们的网站，然后用火狐或者google游览器的YSLOW插件进行压缩效果的查看

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。

下面给出大家几乎是所有的AddOutputFilterByTypeDEFLATE后跟的压缩文件类型，大家可以参照选择。

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

# mod\_expires缓存功能

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。

* 1、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@c64-web /]# /usr/local/apache/bin/apachectl -M | grep deflate

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

[root@c64-web /]# /usr/local/apache/bin/apachectl -M | grep deflate

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

安装了的话，就可以直接进行压缩配置了，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@c64-web /]# cd /server/tools/httpd-2.4.6/modules/filters/ #切到apache源码包mod\_expires所在的目录下

[root@c64-web filters]# /usr/local/apache/bin/apxs -c -i -a mod\_expires.c #以dso的方式编译安装到apache中

[root@c64-web filters]# ll /usr/local/apache/modules/mod\_expires.so #检查mod\_deflate是否安装，成功安装这里会显示出该文件

-rwxr-xr-x 1 root root 37439 11月 5 02:02 /usr/local/apache/modules/mod\_expires.so

* 2、配置缓存参数

注意如果我们是编译安装时已经编译进去的，此时我们需要先将httpd.conf主配置文件中：

LoadModule expires\_module modules/mod\_expires.so

此行解锁后再进行下面操作，不然会报错。

缓存的用法有3种，分别对全局，对目录，对虚拟主机。

a)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：

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 plus12 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"

b)对目录

对目录的配置就是在apache主配置文件中标签内加入如下参数即可：

AllowOverride None

Options None

Require all granted

ErrorDocument 404 /404.html

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 plus12 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"

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：

ServerAdmin 274546888@qq.com

DocumentRoot "/www/html"

ServerName www.jincon.com

ServerAlias jincon.com

CustomLog "|/usr/local/sbin/cronolog /app/logs/www\_access\_%Y%m%d.log" combined

ErrorLog "|/usr/local/sbin/cronolog /app/logs/www\_error\_%Y%m%d.log"

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 plus12 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"

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YSlow插件进行缓存效果查看。

* 扩展：expires模块的语法

上面已经的操作就是实现apache的缓存功能。其中，expires后面的参数你也可以根据需求自行更改，缓存时间也是一样可以更改的。

这里我对expires的模块语法进行一些细说

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 " [plus] {}"

其中的参数有3个：access，now(等价于‘access’)，modification

plus关键字是可选的。

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)

参数类型：years，months，weeks，days，hours，minutes，seconds

下面用几个例子带大家理解下

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“”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"

注意：如果你使用基于最后修改日期的设置，“Expires”头将不会被添加到那些并非来自于硬盘文件的内容，这是因为这些内容并不存在“最后修改时间”的属性。

基准时刻可以是源文件的最后修改时刻或者客户端对源文件的访问时刻，至于使用那一个则由指定。“M”表示源文件的最后修改时刻，“A”表示客户端对源文件的访问时刻，需要注意的是和seconds之间没有空格。

这两个基准的差别是很微妙的。如果使用“M”，所有当前缓存中的文档副本都将在同一时刻过期，这个可能对定期更新的URL(比如位于同一位置的每周通告)很有好处。如果使用“A”，则每个客户端所得到的有效期是不一样的，这个可能对那些几乎不更新的图片很有好处，特别是对于一组都引用相同图片的相关文档。

缓存优化对企业来说是至关重要了的，不同的公司类型，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。

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?

1.第一种：缩短缓存时间例如：1天，不彻底牺牲性能

2.第二种：对缓存的对象改名。

# 启用worker模式，提升并发数(可以达到2000-5000)

apache有两个模式，默认的模式是prefork模式。那么我们为什么要使用worker模式呢?这两种模式又有什么区别呢?

* 1、prefork模式

prefork使用的是多个子进程，而每个子进程只有一个线程，每个进程在某个确定的时间只能维持一个连接。

工作原理：控制进程最初建立若干个子进程，为了不在请求到来时再生成子进程，所以要根据需求不断的创建新的子进程，最大可以达到每秒32个直到满足需求为止。

安装方法：在编译的过程中，加入参数--with-mpm=frefork，不加也可以，因为默认的话，会采用prefork模式。

优点：效率高，稳定，安全。对于线程调试困难的平台来说，调试更加容易些。

缺点：与worker模式相比消耗资源多。

配置参数说明：

</IFModule mpm\_prefork\_module>

StartServers 5 #最初建立的子进程

MinSpareServers 5 #最小空闲进程数，如果空闲的进程小于设定值，Apache会自动建立进程，如果服务器并发及负载大的话，可以考虑加大。

MaxSpareServers 10 #最大空闲进程数，如果空闲的进程大于设定值，Apache会自动kill掉多余的进程，如果服务器负载大的话，可以考虑加大。

MaxClients 150 #设定的是apache可以同时处理的请求，是对apache性能影响最大的参数，就是apache可以同时处理的请求数，就是说，如果有150个用户在访问，那么第151个用户就要等之前的访问结束后才能访问。

MaxRequestsPerChild 0 # 每个子进程可处理的请求数，每个子进程在处理了“MaxRequestsPerChild”个请求后将自动销毁。0以为着无限，即子进程永不销毁。虽然缺省设为0可以使每个子进程处理更多的请求，但如果设置成非0值也有两点重要的好处。(1)可防止意外的内存泄露(2)在服务器负载下载的时候会自动减少子进程数。

</IFModule>

生产环境配置实例1：

</IFModule mpm\_prefork\_module>

StartServers 10

MinSpareServers 10

MaxSpareServers 15

ServerLimit 2000

MaxClients 1000

MaxRequestsPerChild 5000

</IFModule>

* 2、worker模式

worker模式是apche2.x新引进来的模式，是线程与进程的结合，在worker模式下会有多个子进程，每个子进程又会有多个线程。每个线程在某个确定的时间只能维持一个连接。

工作原理：由主控制进程生成若干个子进程，而每个子进程中又包含固定的线程数，各个线程独立处理请求，同样为了不在请求到来时再生成线程，在配置文件中设置了最小和最大的空闲线程数及所有子进程中的线程总数，如果现有子进程中的线程总数不能满足并发及负载，控制进程将派生新的子进程。

安装方法：在配置编译的工程中，加入参数--with-mpm=worker，如果不加的话系统会采用默认prefork模式。

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任何线程一起“死掉”。由于线程共享内存空间，所以一个程式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。

配置参数说明：

StartServers 2 #最初建立的子进程

MaxClients 150 #所有子进程中的线程总数。如果现有子进程中的线程总数不能满足负载，控制进程将派生新的子进程。

MinSpareThreads 25 #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 #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。这时要显式使用threadlimit指令，它的最大缺省值是20000.

MaxRequestsPerChild 0 #功能同prefork模式

常用配置参考：

生产环境配置实例1：

StartServers 3

MaxClients 2000

ServerLimit 25

MinSpareThreads 50

MaxSpareThreads 200

ThreadLimit 200

ThreadsPerChild 100

MaxRequestsPerChild 0

生产环境配置实例2：

StartServers 5

MaxClients 9600

ServerLimit 64

MinSpareThreads 25

MaxSpareThreads 500

ThreadLimit 200

ThreadsPerChild 150

MaxRequestsPerChild 0

生产场景配置实例3：

StartServers 2

MaxClients 500

ServerLimit 25

MinSpareThreads 25

MaxSpareThreads 75

ThreadsPerChild 25

MaxRequestsPerChild 0

生产场景配置实例4：

StartServers 3

MaxClients 1600

ServerLimit 25

MinSpareThreads 50

MaxSpareThreads 200

ThreadLimit 200

ThreadsPerChild 64

worker 模式下所能同时处理的请求总数是由子进程总数乘以Threadsperchild值决定的，应该大于等于maxclients。如果负载很大，现有的子进程数不能满足时，控制进程会派生新的子进程。默认最大的子进程总数是16，如需加大时也需要显示声明serverlimit的值(最大值是20000)

特别说明：如果显示声明了ServerLimit，那么它乘以ThreadsPerChild的值必须大于等于MaxClients，而且 MaxClients必须是ThreadsPerChild的整数倍，否则apache将会自动调节到一个相应值(可能是个非期望值)。

数学表达：

MaxClient<=总的进程数(ServerLimit)x线程数(ThreadsPerChild)

MaxClient%ThreadsPerChild=0

生产环境中我们需要开启httpd-mpm.conf配置行，这样才能实现并发链接数的增加。

1、开启httpd-mpm.conf配置行

这里我们要apache的主配置文件httpd.conf，找到包含httpd-mpm.conf的行，并解开注释

[root@c64-web /]# grep "httpd-mpm.conf" /usr/local/apache/conf/httpd.conf

#Include conf/extra/httpd-mpm.conf

[root@c64-web /]# sed -i 's#\#Include conf/extra/httpd-mpm.conf#Include conf/extra/httpd-mpm.conf#g' /usr/local/apache/conf/httpd.conf

注意：编译安装的，只有此行解开注释了，后面的修改才能生效。

2、修改httpd-mpm.conf配置

1)如果你的apache服务为worker模式，那么生产环境中可以选择上面worker模式的生产环境配置实例1的配置。

2)如果你的apache服务为prefork模式，那么生产环境中可以参考上面prefork模式的生产环境配置实例1的配置。

生产环境中，这里我建议大家选择worker模式。

# 开启防盗链

一些小网站为了盈利，通过盗链来实现对自己网站内容的丰富，这无疑加大了企业的空间和流量的成本，因此我们需要对apache进行防盗链的配置。

由于我们要用到mod\_rewrite模块，因此我们需要先检测该模块是否安装，检测和安装方法与上面mod\_deflate和mod\_expires的一样，这里不多讲了。

防盗链配置：

一般我们都是将防盗链的配置，配置在虚拟主机里面，配置如下：

ServerAdmin jincon@qq.com

DocumentRoot "/www/html"

ServerName www.jincon.com

ServerAlias jincon.com

CustomLog "|/usr/local/sbin/cronolog /app/logs/www\_access\_%Y%m%d.log" combined

ErrorLog "|/usr/local/sbin/cronolog /app/logs/www\_error\_%Y%m%d.log"

RewriteEngine On

RewriteCond %{HTTP\_REFERER} !^http://jincon.com/.\*$ [NC]

RewriteCond %{HTTP\_REFERER} !^http://jincon.com$ [NC]

RewriteCond %{HTTP\_REFERER} !^http://www.jincon.com/.\*$ [NC]

RewriteCond %{HTTP\_REFERER} !^http://www.jincon.com$ [NC]

RewriteRule .\*\.(gif|jpg|swf)$ http://www.jincon.com [R,NC]

如上配置之后，我们重启apache服务即可生效。

# 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。我的http://down.jincon.com就是范例。

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

实现方法有三种：

第一种：

AllowOverride None

Options FollowSymLinks #不填写Indexes项，默认即为关闭。

Require all granted

第二种：

AllowOverride None

Options -Indexes #通过在Indexes前面加-来禁用此功能

Require all granted

第三种：

AllowOverride None

Options None #禁用所有选项

Require all granted

# 禁用AllowOverride选项，关闭.htaccess文件使用

首先是性能考虑，如果AllowOverride启用了.htaccess文件，则apache需要在每个目录中查找.htaccess文件。因此，无论是否真正用到，启用.htaccess都会导致服务器性能的下降。另外，对于每一个请求，都需要读取一次.htaccess文件。

其次是安全考虑，这样会允许用户自己修改服务器的配置，这可能会导致某些意想不到的修改，所以请认真考虑是否应当给予用户这样的特权。

* 1、禁用AllowOverride选项

我们通过修改apache主配置文件httpd.conf中的标签内的AllowOverride选项参数来实现禁用目录浏览。

Options none

AllowOverride None #禁止该选项，防止用户重复载入

Require all granted

通过该设置加快了服务器响应速度，因为它不再让每个请求去寻求每个目录的访问控制文件(.htaccess)

* 2、关闭.htaccess文件使用

默认在Unix平台下能够使用.htaccess来对目录权限进行规则定义，但是这是不安全的，建议关闭，默认的选项：

AccessFileName .htaccess

这里我们将它关闭注释掉

#AccessFileName .htaccess

全部目录权限定义使用httpd.conf中的定义，不使用.htaccess进行定义。

# 关闭自带CGI功能

生产环境中，一般我们不使用apache自带的cgi的功能，如果使用的话，可以用mod\_perl模块来替代。因此我们这里需要将apache主配置文件httpd.conf中有关cgi的内容删除掉。

删除部分：

ScriptAlias /cgi-bin/ "/usr/local/apache2.4.6/cgi-bin/"

<Directory "/usr/local/apache2.4.6/cgi-bin">

AllowOverride None

Options None

Require all granted

</Directory>

# 禁止PHP解析指定站点的目录

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。

# apache的安全模块

我们需要对apache站点增加一些扩展的安全模块，如：mod\_evasive20防DDOS、mod\_limitipconn(针对单站点)配置、mod\_security2防SQL注入等等之类的工具。

另外，下面是推荐给大家的几个工具。

makejail http://www.floc.net/makejail/

它是一个自动把建立jail所需要的程序放到jail内的软件，使用python编写，他有debian和openbsd的版本。

mod-security http://www.modsecurity.org/

它是apche的一个模块，它不仅可以实现过滤请求和日志审计等功能，而且可以防止SQLInjection和跨站脚本攻击等是个很不错的安全模块。

# 使用tmpfs文件系统替代频繁访问的目录

tmpfs 是一种基于内存的文件系统。它最主要的两个优势就是动态文件系统大小和速度快。因为典型的tmpfs文件系统会完全驻留在RAM中，读写几乎可以是瞬间的。即使用了一些交换分区，性能仍然是卓越的。因此对于那些频繁访问的目录，建议使用tmpfs文件系统进行替代。

# 优化系统内核参数

对系统内核的优化参数，我们在系统的基础优化这边博文已经讲过了。系统内核优化最主要的目的就是减少TIME-WAIT的TCP连接数。

由于在基础优化篇我们已经对apache进行了内核参数的优化，因此这里我就不再操作了。这里我将为大家对优化的参数进行讲解。

net.ipv4.tcp\_fin\_timeout #表示如果套接字由本端要求关闭，这个参数决定了它保持在FIN-WAIT-2状态的时间，默认值是60秒。

net.ipv4.tcp\_tw\_reuse #表示开启重用。允许将TIME-WAIT sockets重新用于新的TCP连接，默认值为0，表示关闭。该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_tw\_reuse 0

net.ipv4.tcp\_tw\_recycle # 表示开启TCP连接中TIME-WAIT sockets的快速回收该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_tw\_recycle提示：reuse和recycle 两个参数为了防止生产环境下web，squid等time\_wait过多设置的。

net.ipv4.tcp\_syncookies # 表示开启SYN Cookies功能。当出现SYN等待队列溢出时，启用cookies来处理，可防范少量SYN攻击，Centos5系列默认值为1，表示开启。因此这个参数也可以不添加。该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_syncookies,默认为1

net.ipv4.tcp\_keepalive\_time #表示当keepalive起用的时候，TCP发送keepalive消息的频度。缺省是2小时，改为10分钟。该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_keepalive\_time，默认为7200秒。

net.ipv4.ip\_local\_port\_range #选项用来设定允许系统打开的端口范围。即用于向外连接的端口范围。该参数对应系统路径为：/proc/sys/net/ipv4/ip\_local\_port\_range 32768 61000

net.ipv4.tcp\_max\_syn\_backlog #表示SYN队列的长度，默认为1024，加大队列长度为8192，可以容纳更多等待连接的网络连接数。选项为服务器端用来记录那些尚未收到客户端确认信息的连接请求的最大值。

该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_max\_syn\_backlog

net.ipv4.tcp\_max\_tw\_buckets # 表示系统同时保持TIME\_WAIT套接字的最大数量，如果超过这个数字，TIME\_WAIT套接字将立刻被清楚并打印警告信息。默认为180000，对于Apache、Nginx等服务器来说可以调整低一点，如改为5000-30000，不同业务的服务器也可以给大一旦，比如lvs，squid。上几行的参数可以很好地减少TIME\_WAIT套接字数量，但是对于Squid效果却不大。此项参数可以控制TIME\_WAIT套接字的最大数量，避免 Squid服务器被大量的TIME\_WAIT套接字拖死。此参数对应系统路径为：/proc/sys/net/ipv4 /tcp\_max\_tw\_buckets

net.ipv4.tcp\_synack\_retries #参数的值决定了内核放弃连续之前发送SYN+ACK包的数量。该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_synack\_retries默认值为5

net.ipv4.tcp\_syn\_retries #表示在内核放弃建立连接之前发送SYN包的数量。该参数对应系统路径为：/proc/sys/net/ipv4/tcp\_syn\_retries 5

net.ipv4.tcp\_max\_orphans # 选项用于设定系统中最多有多少个TCP套接字不被关联到任何一个用户文件句柄上。如果超过这个数字，孤立连接将立即被复位并打印出警告信息。这个限制这是为了防止简单的Dos攻击，不能过分依靠这个限制甚至认为减少这个值，更多的情况是增加这个值。该参数对应系统路径为：/proc/sys/net /ipv4/tcp\_max\_orphans 65536

# 尽可能减少HTTP请求数

http 请求是要开销的，想办法减少请求数自然可以提高网页速度。常用的方法，合并css，js(将一个页面中的css和js文件分别合并)以及 Imagemaps和csssprites等。当然或许将css，js文件拆分成多个是因为css结构，共用等方面的考虑。阿里巴巴中文站当时的做法是开发时依然分开开发，然后在后台对js，css进行合并，这样对于游览器来说依然是一个请求，但是开发时仍然能还原成多个，方便管理和重复引用。yahoo 甚至建议将首页的css和js直接写在页面文件里面，而不是外部引用。因为首页的访问量太大了，这么做可以减少两个请求数。而事实上国内的很多门户都是这么做的。

其中csssprites是指只用将页面上的背景图合并成一张，然后通过css的background-position属性定义不过的值来取他的背景。淘宝和阿里巴巴中文站目前都是这样做的。http://www.csssprites.com/这是个工具网站，它可以自动将你上传的图片合并，并给出对应的background-position坐标。并将结果以png和gif的格式输出。

# 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。

国内有名的CND公司：网宿，蓝汛(chinacache)，快网

# 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。

那么如何实现分离呢?

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.jincon.com/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.jincon.com/upload/index.php 就抛给上传服务器(CDN最好)，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web服务器。