**[nginx](javascript:;" \t "_self)缓存cache的5种方案**

1、传统缓存之一（404）  
　　这个办法是把nginx的404错误定向到后端，然后用proxy\_store把后端返回的页面保存。  
　　配置：  
　　location / {  
　　 root /home/html/;#主目录

expires 1d;#网页的过期时间，1天

error\_page 404 =200 /fetch$request\_uri;#404定向到/fetch目录下，即又请求/fetch/XXX文件

}  
　　location /fetch/ {#404定向到这里  
　　 internal;#指明这个目录不能在外部直接访问到  
　　 expires 1d;#网页的过期时间

   alias /html/;  
    #proxy\_store关键字会将文件保存到这目录下  
　　 proxy\_pass<http://www.sudone.com/>;#后端upstream地址，/fetch同时是一个代理  
　　 proxy\_set\_header Accept-Encoding '';#让后端不要返回压缩（gzip或deflate）的内容，保存压缩后的内容会引发乱子。  
　　 proxy\_store on;#指定nginx将代理返回的内容以文件的形式保存  
　　 proxy\_temp\_path /home/tmp;#临时目录，这个目录要和/home/html在同一个硬盘分区内  
　　}  
  
　　使用的时候还有要注意是nginx要有权限往/home/tmp和/home/html下有写入文件的权限，在linux下nginx一般会配置成nobody用户运行，这样这两个目录就要chown nobody，设成nobody用户专用，当然也可以chmod 777，不过所有有经验的系统管理员都会建议不要随便使用777。  
　　2、传统缓存之二（!-e）  
　　原理和404跳转基本一致，但更简洁一些：  
　　location / {  
　　 root /home/html/;  
　　 proxy\_store on;#指定nginx将请求的结果以文件的形式来存储  
　　 proxy\_set\_header Accept-Encoding '';  
　　 proxy\_temp\_path /home/tmp;  
　　 if ( !-f $request\_filename )  
　　 {  
　　 proxy\_pass <http://www.sudone.com/>;  
　　 }  
　　}  
　　可以看到这个配置比404节约了不少代码，它是用!-f来判断请求的文件在文件系统上存不存在，不存在就proxy\_pass到后端，返回同样是用proxy\_store保存。  
　　两种传统缓存都有着基本一样的优点和缺点：  
　　缺点1：不支持带参数的动态链接，比如read.php?id=1，因为nginx只保存文件名，所以这个链接只在文件系统下保存为read.php，这样用户访问read.php?id=2时会返回不正确的结果。同时不支持[http://www.sudone.com/](http://www.sudone.com/" \t "_blank)这种形式的首页和二级目录[http://www.sudone.com/download/](http://www.sudone.com/download/" \t "_blank)，因为nginx非常老实，会将这样的请求照链接写入文件系统，而这个链接显然是一个目录，所以保存失败。这些情况都需要写rewrite才能正确保存。  
　　缺点2：nginx内部没有缓存过期和清理的任何机制，这些缓存的文件会永久性地保存在机器上，如果要缓存的东西非常多，那就会撑暴整个硬盘空间。为此可以使用一个**[shell](javascript:;" \t "_self)**脚本定期清理，同时可以撰写php等动态程序来做实时更新。  
　　缺点3：只能缓存200状态码，因此后端返回301/302/404等状态码都不会缓存，假如恰好有一个访问量很大的伪静态链接被删除，那就会不停穿透导致后端承载不小压力。  
　　缺点4：nginx不会自动选择内存或硬盘作为存储介质，一切由配置决定，当然在当前的**[操作系统](javascript:;" \t "_self)**里都会有操作系统级的文件缓存机制，所以存在硬盘上也不需要过分担心大并发读取造成的io性能问题。  
　　nginx传统缓存的缺点也是它和squid等缓存软件的不同之特色，所以也可看作其优点。在生产应用中它常常用作和squid的搭档，squid对于带?的链接往往无法阻挡，而nginx能将其访问拦住，例如：[http://sudone.com/?](http://sudone.com/?" \t "_blank)和[http://sudone.com/](http://sudone.com/" \t "_blank)在squid上会被当做两个链接，所以会造成两次穿透；而nginx只会保存一次，无论链接变成[http://sudone.com/?1](http://sudone.com/?1" \t "_blank)还是[http://sudone.com/?123](http://sudone.com/?123" \t "_blank)，均不能透过nginx缓存，从而有效地保护了后端主机。  
　　nginx会非常老实地将链接形式保存到文件系统中，这样对于一个链接，可以很方便地查阅它在缓存机器上的缓存状态和内容，也可以很方便地和别的文件管理器如rsync等配合使用，它完完全全就是一个文件系统结构。  
　　这两种传统缓存都可以在linux下将文件保存到/dev/shm里，一般我也是这么做的，这样可以利用系统内存来做缓存，利用内存的话，清理过期内容速度就会快得多。使用/dev/shm/时除了要把tmp目录也指向到/dev/shm这个分区外，如果有大量小文件和目录，还要修改一下这个内存分区的inode数量和最大容量：  
　　mount -o size=2500M -o nr\_inodes=480000 -o noatime,nodiratime -o remount /dev/shm  
　　上面的命令在一台有3G内存的机器上使用，因为/dev/shm默认最大内存是系统内存的一半就是1500M，这条命令将其调大成2500M，同时shm系统inode数量默认情况下可能是不够用的，但有趣的是它可以随意调节，这里调节为480000保守了点，但也基本够用了。  
　　3、基于memcached的缓存  
　　nginx对memcached有所支持，但是功能并不是特别之强，性能上还是非常之优秀。  
　　location /mem/ {  
　　if ( $uri ~ "^/mem/([0-9A-Za-z\_]\*)$" )  
　　{  
　　set $memcached\_key "$1";  
　　memcached\_pass     192.168.1.2:11211;  
　　}  
　　expires 70;  
　　}  
　　这个配置会将[http://sudone.com/mem/abc](http://sudone.com/mem/abc" \t "_blank)指明到memcached的abc这个key去取数据。  
　　nginx目前没有写入memcached的任何机制，所以要往memcached里写入数据得用后台的动态语言完成，可以利用404定向到后端去写入数据。  
　　4、基于第三方插件ncache  
　　ncache是新浪兄弟开发的一个不错的项目，它利用nginx和memcached实现了一部分类似squid缓存的功能，我并没有使用这个插件的经验，可以参考：  
　　<http://code.google.com/p/ncache/>  
　　5、nginx新开发的proxy\_cache功能  
　　从nginx-0.7.44版开始，nginx支持了类似squid较为正规的cache功能，目前还处于开发阶段，支持相当有限，这个缓存是把链接用md5编码hash后保存，所以它可以支持任意链接，同时也支持404/301/302这样的非200状态。  
　　配置：  
　　首先配置一个cache空间：  
　　proxy\_cache\_path /path/to/cache levels=1:2 keys\_zone=NAME:10m inactive=5m max\_size=2m clean\_time=1m;  
　　注意这个配置是在**[server](javascript:;" \t "_self)**标签外，levels指定该缓存空间有两层hash目录，第一层目录是1个字母，第二层为2个字母，保存的文件名就会类似/path/to/cache/c/29/b7f54b2df7773722d382f4809d65029c；keys\_zone为这个空间起个名字，10m指空间大小为10MB；inactive的5m指缓存默认时长5分钟；max\_size的2m是指单个文件超过2m的就不缓存；clean\_time指定一分钟清理一次缓存。  
　　location / {  
　　proxy\_pass<http://www.sudone.com/>;  
　　proxy\_cache NAME;#使用NAME这个keys\_zone  
　　proxy\_cache\_valid 200 302 1h;#200和302状态码保存1小时  
　　proxy\_cache\_valid 301 1d;#301状态码保存一天  
　　proxy\_cache\_valid any 1m;#其它的保存一分钟  
　　}  
　　ps：支持cache的0.7.44到0.7.51这几个版本的稳定性均有问题，访问有些链接会出现错误，所以这几个版本最好不要在生产环境中使用。nginx-0.7下目前所知较为稳定的版本是0.7.39。稳定版0.6.36版也是近期更新，如果在配置里没有使用到0.7的一些新标签新功能，也可以使用0.6.36版。