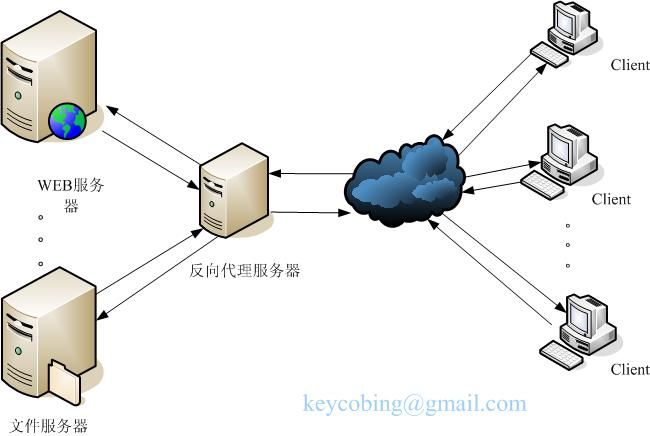
**一、什么是反向代理**

**反向代理**（Reverse Proxy）方式是指以代理服务器来接受Internet上的连接请求，然后将请求转发给内部网络上的服务器；并将从服务器上得到的结果返回给Internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

　　通常的代理服务器，只用于代理内部网络对Internet外部网络的连接请求，客户机必须指定代理服务器，并将本来要直接发送到Web服务器上的 http请求发送到代理服务器中。不支持外部网络对内部网络的连接请求，因为内部网络对外部网络是不可见的。当一个代理服务器能够代理外部网络上的主机， 访问内部网络时，这种代理服务的方式称为反向代理服务。此时代理服务器对外就表现为一个Web服务器，外部网络就可以简单把它当作一个标准的Web服务器 而不需要特定的配置。不同之处在于，这个服务器没有保存任何网页的真实数据，所有的静态网页或者CGI程序，都保存在内部的Web服务器上。因此对反向代 理服务器的攻击并不会使得网页信息遭到破坏，这样就增强了Web服务器的安全性。

　　反向代理就是通常所说的web服务器加速，它是一种通过在繁忙的web服务器和外部网络之间增加一个高速的web缓冲服务器来降低实际的web服务 器的负载的一种技术。反向代理是针对web服务器提高加速功能，作为代理缓存，它并不是针对浏览器用户，而针对一台或多台特定的web服务器，它可以代理 外部网络对内部网络的访问请求。

　　反向代理服务器会强制将外部网络对要代理的服务器的访问经过它，这样反向代理服务器负责接收客户端的请求，然后到源服务器上获取内容，把内容返回给 用户，并把内容保存到本地，以便日后再收到同样的信息请求时，它会把本地缓存里的内容直接发给用户，以减少后端web服务器的压力，提高响应速度。



**二 、反向代理服务器的工作原理**

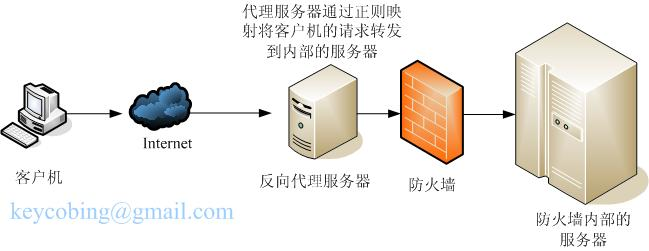
　　反向代理服务器通常有两种模型，它可以作为内容服务器的替身，也可以作为内容服务器集群的负载均衡器。

**1，作内容服务器的替身**

　　如果您的内容服务器具有必须保持安全的敏感信息，如信用卡号数据库，可在防火墙外部设置一个代理服务器作为内容服务器的替身。当外部客户机尝试访问内容服务器时，会将其送到代理服务器。实际内容位于内容服务器上，在防火墙内部受到安全保护。代理服务器位于防火墙外部，在客户机看来就像是内容服务器。

　　当客户机向站点提出请求时，请求将转到代理服务器。然后，代理服务器通过防火墙中的特定通路，将客户机的请求发送到内容服务器。内容服务器再通过该通道将结果回传给代理服务器。代理服务器将检索到的信息发送给客户机，好像代理服务器就是实际的内容服务器（参见图 2）。如果内容服务器返回错误消息，代理服务器会先行截取该消息并更改标头中列出的任何 URL，然后再将消息发送给客户机。如此可防止外部客户机获取内部内容服务器的重定向 URL。

　　这样，代理服务器就在安全数据库和可能的恶意攻击之间提供了又一道屏障。与有权访问整个数据库的情况相对比，就算是侥幸攻击成功，作恶者充其量也仅限于访问单个事务中所涉及的信息。未经授权的用户无法访问到真正的内容服务器，因为防火墙通路只允许代理服务器有权进行访问。

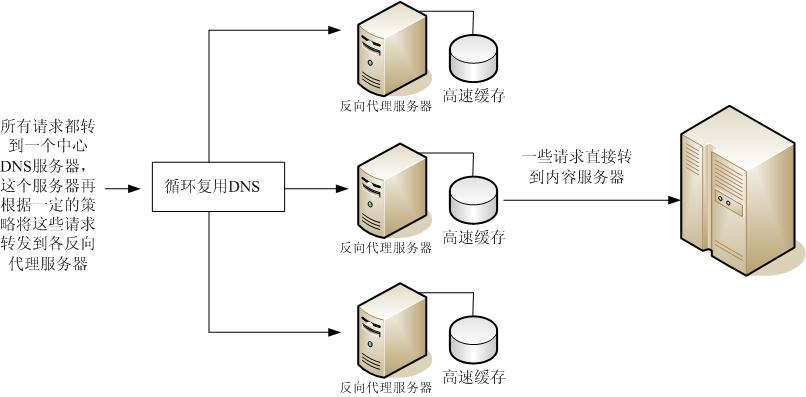


**2，作为内容服务器的负载均衡器**

　　可以在一个组织内使用多个代理服务器来平衡各 Web 服务器间的网络负载。在此模型中，可以利用代理服务器的高速缓存特性，创建一个用于负载平衡的服务器池。此时，代理服务器可以位于防火墙的任意一侧。如果 Web 服务器每天都会接收大量的请求，则可以使用代理服务器分担 Web 服务器的负载并提高网络访问效率。

　　对于客户机发往真正服务器的请求，代理服务器起着中间调停者的作用。代理服务器会将所请求的文档存入高速缓存。如果有不止一个代理服务器，DNS 可以采用“循环复用法”选择其 IP 地址，随机地为请求选择路由。客户机每次都使用同一个 URL，但请求所采取的路由每次都可能经过不同的代理服务器。

　　可以使用多个代理服务器来处理对一个高用量内容服务器的请求，这样做的好处是内容服务器可以处理更高的负载，并且比其独自工作时更有效率。在初始启动期间，代理服务器首次从内容服务器检索文档，此后，对内容服务器的请求数会大大下降。



**三、反向代理的好处**

1、 解决了网站服务器对外可见的问题；

2、节约了有限的IP地址资源，企业内所有的网站共享一个在internet中注册的IP地址，这些服务器分配私有地址，采用虚拟主机的方式对外提供服务；

3、 保护了真实的web服务器，web服务器对外不可见，外网只能看到反向代理服务器，而反向代理服务器上并没有真实数据，因此，保证了web服务器的资源安全；

4、 加速了对网站访问速度，减轻web服务器的负担，反向代理具有缓存网页的功能，如果用户需要的内容在缓存中，则可以直接从代理服务其中获取，减轻了web服务器的负荷，同时也加快了用户的访问速度。

**四、Nginx作为反向代理实现负载均衡的示例**

因为nginx在处理并发方面的优势，现在这个应用非常常见。当然了Apache的 mod\_proxy和mod\_cache结合使用也可以实现对多台app server的反向代理和负载均衡，但是在并发处理方面apache还是没有nginx擅长。

1)环境：

　　a. 我们本地是Windows系统，然后使用VirutalBox安装一个虚拟的Linux系统。在本地的Windows系统上分别安装nginx(侦听 8080端口)和apache(侦听80端口)。在虚拟的Linux系统上安装apache(侦听80端口)。这样我们相当于拥有了1台nginx在前端 作为反向代理服务器；后面有2台apache作为应用程序服务器(可以看作是小型的server cluster。;-) )；

　　b. nginx用来作为反向代理服务器，放置到两台apache之前，作为用户访问的入口；nginx仅仅处理静态页面，动态的页面(php请求)统统都交付给后台的两台apache来处理。也就是说，可以把我们网站的静态页面或者文件放置到nginx的目录下；动态的页面和数据库访问都保留到后台的apache服务器上。

　　c. 如下介绍两种方法实现server cluster的负载均衡。

　　我们假设前端nginx(为127.0.0.1:80)仅仅包含一个静态页面index.html；后台的两个apache服务器(分别为localhost:80和158.37.70.143:80)，一台根目录放置phpMyAdmin文件夹 和test.php(里面测试代码为print "server1";)，另一台根目录仅仅放置一个test.php(里面测试代码为print "server2";)。

2)针对不同请求的负载均衡：

　　a. 在最简单地构建反向代理的时候(nginx仅仅处理静态不处理动态内容，动态内容交给后台的apache server来处理)，我们具体的设置为：在nginx.conf中修改： location ~ /.php$ { proxy\_pass 158.37.70.143:80 ; }

　　这样当客户端访问localhost:8080/index.html的时候，前端的nginx会自动进行响应；

　　当用户访问localhost:8080/test.php的时候(这个时候nginx目录下根本就没有该文件)，但是通过上面的设置 location ~ /.php$(表示正则表达式匹配以.php结尾的文件，详情参看location是如何定义和匹配的http://wiki.nginx.org /NginxHttpCoreModule) ，nginx服务器会自动pass给158.37.70.143的apache服务器了。该服务器下的test.php就会被自动解析，然后将html的 结果页面返回给nginx，然后nginx进行显示(如果nginx使用memcached模块或者squid还可以支持缓存)，输出结果为打印 server2。

　　如上是最为简单的使用nginx做为反向代理服务器的例子；

　　b. 我们现在对如上例子进行扩展，使其支持如上的两台服务器。

　　我们设置nginx.conf的server模块部分，将对应部分修改为：

　　location ^~ /phpMyAdmin/ { proxy\_pass 127.0.0.1:80 ; }location ~ /.php$ { proxy\_pass 158.37.70.143:80 ; }

　　上面第一个部分location ^~ /phpMyAdmin/，表示不使用正则表达式匹配(^~)，而是直接匹配，也就是如果客户端访问的URL是以http: //localhost:8080/phpMyAdmin/ 开头的话(本地的nginx目录下根本没有phpMyAdmin目录)，nginx会自动pass到127.0.0.1:80 的Apache服务器，该服务器对phpMyAdmin目录下的页面进行解析，然后将结果发送给nginx，后者显示；

　　如果客户端访问URL是http://localhost/test.php 的话，则会被pass到158.37.70.143:80 的apache进行处理。

　　因此综上，我们实现了针对不同请求的负载均衡。

　　如果用户访问静态页面index.html，最前端的nginx直接进行响应；

　　如果用户访问test.php页面的话，158.37.70.143:80 的Apache进行响应；

　　如果用户访问目录phpMyAdmin下的页面的话，127.0.0.1:80 的Apache进行响应；

3)访问同一页面的负载均衡：

　　即用户访问http://localhost:8080/test.php 这个同一页面的时候，我们实现两台服务器的负载均衡(实际情况中，这两个服务器上的数据要求同步一致，这里我们分别定义了打印server1和server2是为了进行辨认区别)。

　　a. 现在我们的情况是在windows下nginx是localhost侦听8080端口；

　　两台apache，一台是127.0.0.1:80(包含test.php页面但是打印server1)，另一台是虚拟机的158.37.70.143:80(包含test.php页面但是打印server2)。

　　b. 因此重新配置nginx.conf为：

　　首先在nginx的配置文件nginx.conf的http模块中添加，服务器集群server cluster(我们这里是两台)的定义： upstream myCluster { server 127.0.0.1:80 ; server 158.37.70.143:80 ; }表示这个server cluster包含2台服务器>然后在server模块中定义，负载均衡： location ~ /.php$ { proxy\_pass http://myCluster ; #这里的名字和上面的cluster的名字相同proxy\_redirect off; proxy\_set\_header Host $host; proxy\_set\_header X-Real-IP $remote\_addr; proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for; }这样的话，如果访问http://localhost:8080/test.php 页面的话，nginx目录下根本没有该文件，但是它会自动将其pass到myCluster定义的服务区机群中，分别由127.0.0.1:80;或者 158.37.70.143:80;来做处理。上面在定义upstream的时候每个server之后没有定义权重，表示两者均衡；如果希望某个更多响应 的话例如： upstream myCluster { server 127.0.0.1:80 weight=5;server 158.37.70.143:80 ; }这样表示5/6的几率访问第一个server,1/6访问第二个。另外还可以定义max\_fails和fail\_timeout等参数。

　　综上，我们使用nginx的反向代理服务器reverse proxy server的功能，将其布置到多台apache server的前端。

　　nginx仅仅用来处理静态页面响应和动态请求的代理pass，后台的apache server作为app server来对前台pass过来的动态页面进行处理并返回给nginx。

　　通过以上的架构，我们可以实现nginx和多台apache构成的机群cluster的负载均衡。两种均衡：

　　1)可以在nginx中定义访问不同的内容，代理到不同的后台server；如上例子中的访问phpMyAdmin目录代理到第一台server上；访问test.php代理到第二台server上；

　　2)可以在nginx中定义访问同一页面，均衡(当然如果服务器性能不同可以定义权重来均衡)地代理到不同的后台server上。如上的例子访问test.php页面，会均衡地代理到server1或者server2上。

　　实际应用中，server1和server2上分别保留相同的app程序和数据，需要考虑两者的数据同步。

http://www.cnblogs.com/smallrookie/p/5634738.html