反向代理及 Nginx 示例

1 反向代理的概念

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。

方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。

2 反向代理服务器与内容服务器

代理服务器充当服务器的替身，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。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。文章来源：一起遛书网 www.176book.com。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。

3 反向代理服务器的工作流程

1） 用户通过域名发出访问 web 服务器的请求，该域名被 DNS 服务器解析为反向代理服务器的 IP 地址；

2） 反向代理服务器接受用户的请求；

3） 反向代理服务器在本地缓存中查找请求的内容，找到后直接把内容发送给用户；

4） 如果本地缓存里没有用户所请求的信息内容，反向代理服务器会代替用户向源服务器请求同样的信息内容，并把信息内容发给用户，如果信息内容是缓存的还会把它保存到缓存中。

4 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。

5 Nginx 作为反向代理实现负载均衡的示例

我们介绍了 nginx 这个轻量级的高性能 server 主要可以干的两件事情：

直接作为 http server(代替 apache，对 PHP 需要 FastCGI 处理器支持，这个我们之后介绍)；

另外一个功能就是作为反向代理服务器实现负载均衡 (如下我们就来举例说明实际中如何使用 nginx 实现负载均衡)。

因为 nginx 在处理并发方面的优势，现在这个应用非常常见。文章来源：一起遛书网 www.176book.com。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。