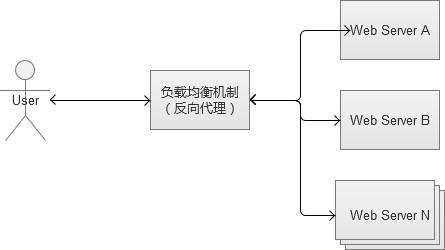
亿级Web系统搭建——单机到分布式集群

<https://segmentfault.com/p/1210000008562600/read>

**Web负载均衡**

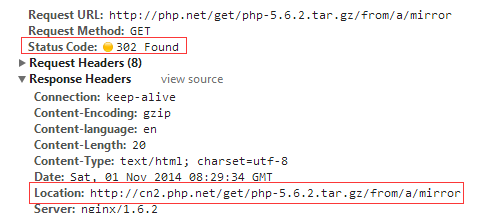
Web负载均衡（Load Balancing），简单地说就是给我们的服务器集群分配“工作任务”，而采用恰当的分配方式，对于保护处于后端的Web服务器来说，非常重要。



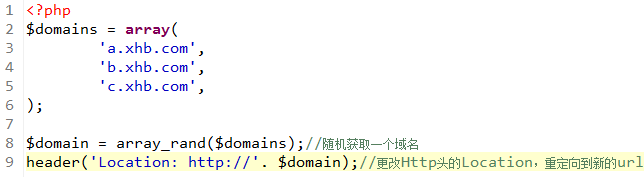
负载均衡的策略有很多，我们从简单的讲起哈。

**1. HTTP重定向**

当用户发来请求的时候，Web服务器通过修改HTTP响应头中的Location标记来返回一个新的url，然后浏览器再继续请求这个新url，实际上就是页面重定向。通过重定向，来达到“负载均衡”的目标。例如，我们在下载PHP源码包的时候，点击下载链接时，为了解决不同国家和地域下载速度的问题，它会返回一个离我们近的下载地址。重定向的HTTP返回码是302，如下图：



如果使用PHP代码来实现这个功能，方式如下：



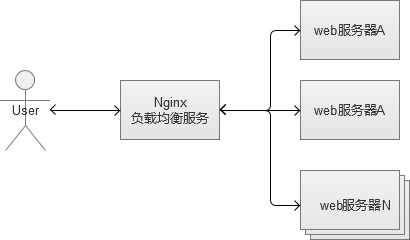
这个重定向非常容易实现，并且可以自定义各种策略。但是，它在大规模访问量下，性能不佳。而且，给用户的体验也不好，实际请求发生重定向，增加了网络延时。

(http状态码301和302详解及区别——辛酸的探索之路

<http://blog.csdn.net/grandPang/article/details/47448395>)

**2. 反向代理负载均衡**

反向代理服务的核心工作主要是转发HTTP请求，扮演了浏览器端和后台Web服务器中转的角色。因为它工作在HTTP层（应用层），也就是网络七层结构中的第七层，因此也被称为“七层负载均衡”。可以做反向代理的软件很多，比较常见的一种是Nginx。



Nginx是一种非常灵活的反向代理软件，可以自由定制化转发策略，分配服务器流量的权重等。反向代理中，常见的一个问题，就是Web服务器存储的session数据，因为一般负载均衡的策略都是随机分配请求的。同一个登录用户的请求，无法保证一定分配到相同的Web机器上，会导致无法找到session的问题。

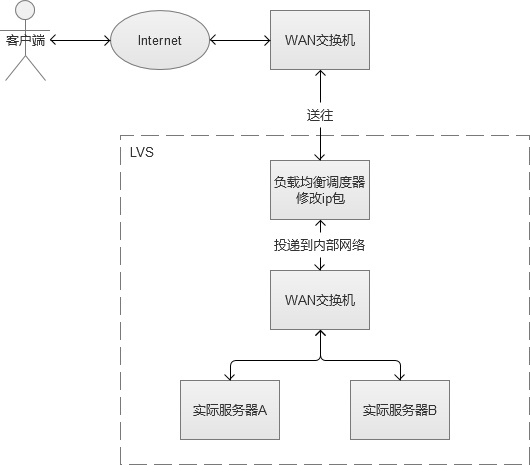
解决方案主要有两种：

* 配置反向代理的转发规则，让同一个用户的请求一定落到同一台机器上（通过分析cookie），复杂的转发规则将会消耗更多的CPU，也增加了代理服务器的负担。
* 将session这类的信息，专门用某个独立服务来存储，例如redis/memchache，这个方案是比较推荐的。

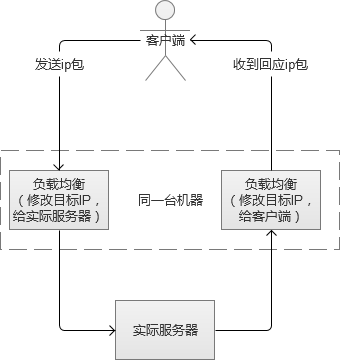
反向代理服务，也是可以开启缓存的，如果开启了，会增加反向代理的负担，需要谨慎使用。（？） 这种负载均衡策略实现和部署非常简单，而且性能表现也比较好。但是，它有“单点故障”的问题，如果挂了，会带来很多的麻烦。而且，到了后期Web服务器继续增加，它本身可能成为系统的瓶颈。

**3. IP负载均衡**

IP负载均衡服务是工作在网络层（修改IP）和传输层（修改端口，第四层），比起工作在应用层（第七层）性能要高出非常多。原理是，他是对IP层的数据包的IP地址和端口信息进行修改，达到负载均衡的目的。这种方式，也被称为“四层负载均衡”。常见的负载均衡方式，是LVS（Linux Virtual Server，Linux虚拟服务），通过IPVS（IP Virtual Server，IP虚拟服务）来实现。



在负载均衡服务器收到客户端的IP包的时候，会修改IP包的目标IP地址或端口，然后原封不动地投递到内部网络中，数据包会流入到实际Web服务器。实际服务器处理完成后，又会将数据包投递回给负载均衡服务器，它再修改目标IP地址为用户IP地址，最终回到客户端。

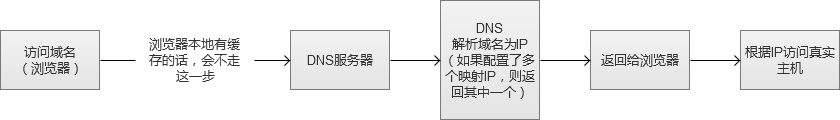


上述的方式叫LVS-NAT，除此之外，还有LVS-RD（直接路由），LVS-TUN（IP隧道），三者之间都属于LVS的方式，但是有一定的区别，篇幅问题，不赘叙。

IP负载均衡的性能要高出Nginx的反向代理很多，它只处理到传输层为止的数据包，并不做进一步的组包，然后直接转发给实际服务器。不过，它的配置和搭建比较复杂。

**4. DNS负载均衡**

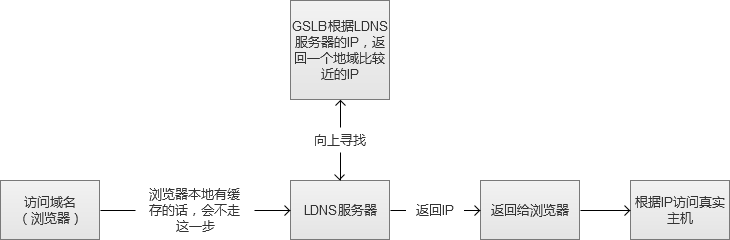
DNS（Domain Name System）负责域名解析的服务，域名url实际上是服务器的别名，实际映射是一个IP地址，解析过程，就是DNS完成域名到IP的映射。而一个域名是可以配置成对应多个IP的。因此，DNS也就可以作为负载均衡服务。



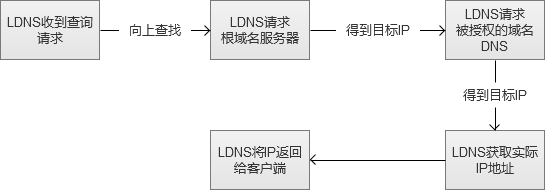
这种负载均衡策略，配置简单，性能极佳。但是，不能自由定义规则，而且，变更被映射的IP或者机器故障时很麻烦，还存在DNS生效延迟的问题。

**5. DNS/GSLB负载均衡**

我们常用的CDN（Content Delivery Network，内容分发网络）实现方式，其实就是在同一个域名映射为多IP的基础上更进一步，通过GSLB（Global Server Load Balance，全局负载均衡）按照指定规则映射域名的IP。一般情况下都是按照地理位置，将离用户近的IP返回给用户，减少网络传输中的路由节点之间的跳跃消耗。

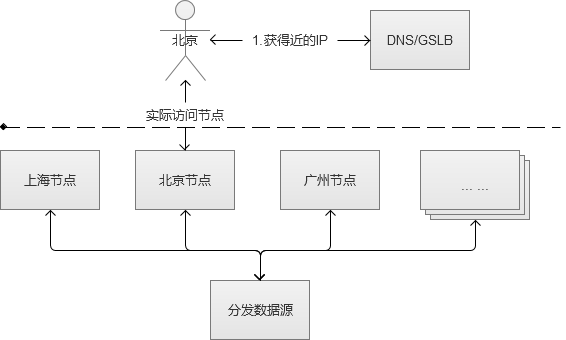


图中的“向上寻找”，实际过程是LDNS（Local DNS）先向根域名服务（Root Name Server）获取到顶级根的Name Server（例如.com的），然后得到指定域名的授权DNS，然后再获得实际服务器IP。



CDN在Web系统中，一般情况下是用来解决大小较大的静态资源（html/Js/Css/图片等）的加载问题，让这些比较依赖网络下载的内容，尽可能离用户更近，提升用户体验。

例如，我访问了一张imgcache.gtimg.cn上的图片（腾讯的自建CDN，不使用qq.com域名的原因是防止http请求的时候，带上了多余的cookie信息），我获得的IP是183.60.217.90。



这种方式，和前面的DNS负载均衡一样，不仅性能极佳，而且支持配置多种策略。但是，搭建和维护成本非常高。互联网一线公司，会自建CDN服务，中小型公司一般使用第三方提供的CDN。