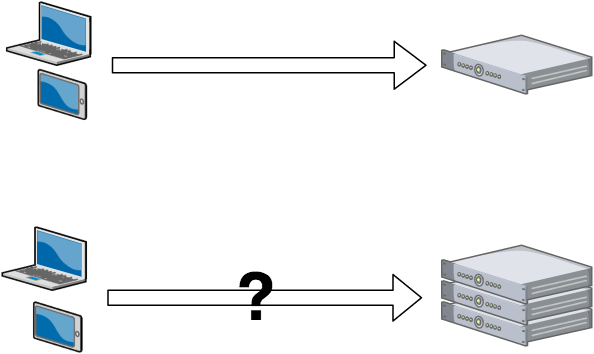
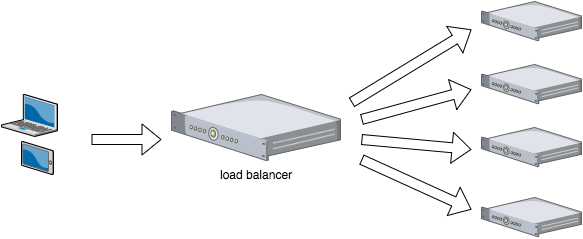
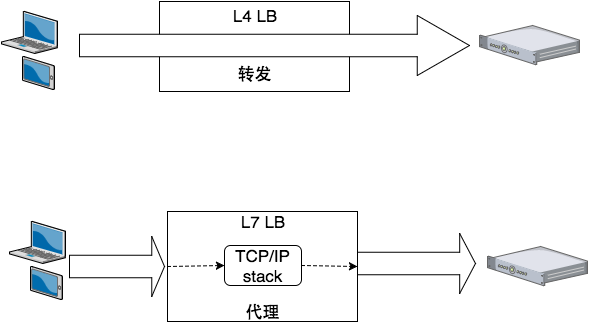
**[负载均衡基础知识](http://www.cnblogs.com/danbing/p/7459224.html)**

**一、什么是负载均衡？**

　互联网早期，业务流量比较小并且业务逻辑比较简单，单台服务器便可以满足基本的需求；但随着互联网的发展，业务流量越来越大并且业务逻辑也越来越复杂，单台机器的性能问题以及单点问题凸显了出来，因此需要多台机器来进行性能的水平扩展以及避免单点故障。但是要如何将不同的用户的流量分发到不同的服务器上面呢？  


　早期的方法是使用DNS做负载，通过给客户端解析不同的IP地址，让客户端的流量直接到达各个服务器。但是这种方法有一个很大的缺点就是延时性问题，在做出调度策略改变以后，由于DNS各级节点的缓存并不会及时的在客户端生效，而且DNS负载的调度策略比较简单，无法满足业务需求，因此就出现了负载均衡。  


**客户端的流量首先会到达负载均衡服务器**，由负载均衡服务器**通过一定的调度算法将流量分发到不同的应用服务器上面**，同时负载均衡服务器也会**对应用服务器做周期性的健康检查**，当**发现故障节点**时便动态的将节点从应用服务器集群中**剔除**，以此来保证应用的高可用。  


　负载均衡又分为四层负载均衡和七层负载均衡。四层负载均衡工作在OSI模型的**传输层，主要工作是转发**，它在接收到客户端的流量以后通过**修改数据包的地址信息将流量转发到应用服务器。**

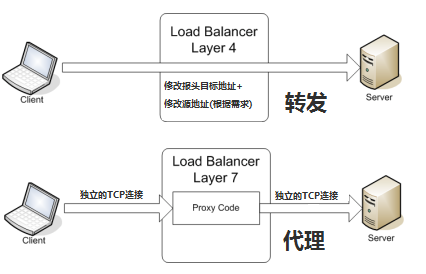
　七层负载均衡工作在OSI模型的应用层，因为它需要解析应用层流量，所以七层负载均衡在接到客户端的流量以后，**还需要一个完整的TCP/IP协议栈**。**七层负载均衡会与客户端建立一条完整的连接并将应用层的请求流量解析出来，再按照调度算法选择一个应用服务器**，并与应用服务器建立另外一条连接将请求发送过去，**因此七层负载均衡的主要工作就是代理。**

**二、四层和七层负载均衡的区别？**

**2.1 - 技术原理上的区别。**

　所谓四层负载均衡，也就是**主要通过报文中的目标地址和端口，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。**

　以常见的TCP为例，负载均衡设备在接收到第一个来自客户端的SYN 请求时，即通过上述方式**选择一个最佳的服务器，并对报文中目标IP地址进行修改(改为后端服务器IP），直接转发给该服务器**。TCP的连接建立，即三次握手是客户端和服务器直接建立的，**负载均衡设备只是起到一个类似路由器的转发动作**。在某些部署情况下，为保证服务器回包可以正确返回给负载均衡设备，在转发报文的同时可能还会对报文原来的源地址进行修改。



　所谓七层负载均衡，也称为“内容交换”，也就是主要通过**报文中的真正有意义的应用层内容**，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。

　以常见的TCP为例，负载均衡设备如果要根据真正的应用层内容再选择服务器，只能**先代理最终的服务器和客户端建立连接(三次握手)后，才可能接受到客户端发送的真正应用层内容的报文**，然后再根据该报文中的特定字段，再加上负载均衡设备设置的服务器选择方式，决定最终选择的内部服务器。  
　  
　负载均衡设备在这种情况下，更类似于一个代理服务器。负载均衡和前端的客户端以及后端的服务器会分别建立TCP连接。所以从这个技术原理上来看，七层负载均衡明显的对负载均衡设备的要求更高，处理七层的能力也必然会低于四层模式的部署方式。那么，为什么还需要七层负载均衡呢？

**2.2 - 应用场景的需求。**

　七层应用负载的好处，是使得整个网络更**"智能化"**, 参考我们之前的[另外一篇专门针对HTTP应用的优化的介绍](http://virtualadc.blog.51cto.com/3027116/580832)，就可以基本上了解这种方式的优势所在。例如访问一个网站的用户流量，可以通过七层的方式，将对图片类的请求转发到特定的图片服务器并可以使用缓存技术；将对文字类的请求可以转发到特定的文字服务器并可以使用压缩技术。

　当然这只是七层应用的一个小案例，从技术原理上，**这种方式可以对客户端的请求和服务器的响应进行任意意义上的修改，极大的提升了应用系统在网络层的灵活性**。很多在后台，(例如Nginx或者Apache)上部署的功能可以前移到负载均衡设备上，例如客户请求中的Header重写，服务器响应中的关键字过滤或者内容插入等功能。

　另外一个常常被提到功能就是**安全性**。网络中最常见的SYN Flood攻击，即黑客控制众多源客户端，使用虚假IP地址对同一目标发送SYN攻击，通常这种攻击会大量发送SYN报文，耗尽服务器上的相关资源，以达到*Denial of Service*(*DoS*)的目的。

　从技术原理上也可以看出，**四层模式下这些SYN攻击都会被转发到后端的服务器上；而七层模式下这些SYN攻击自然在负载均衡设备上就截止，不会影响后台服务器的正常运营。**另外负载均衡设备可以在七层层面设定多种策略，过滤特定报文，例如*SQL* Injection等应用层面的特定攻击手段，从应用层面进一步提高系统整体安全。

　现在的**7层负载均衡，主要还是着重于应用广泛的HTTP协议，**所以其应用范围主要是众多的网站或者内部信息平台等基于B/S开发的系统。 4层负载均衡则对应其他TCP应用，例如基于C/S开发的ERP等系统。

**2.3 - 七层应用需要考虑的问题。**

* 是否真的必要，七层应用的确可以提高流量智能化，*同时必不可免的带来设备配置复杂，负载均衡压力增高以及故障排查上的复杂性等问题*。在设计系统时需要考虑四层七层同时应用的混杂情况。
* 是否真的可以提高安全性。例如**SYN Flood攻击**，七层模式的确将这些流量从服务器屏蔽，但*负载均衡设备***本身要有强大的抗DDoS能力**，否则即使服务器正常而作为中枢调度的负载均衡设备故障也会导致整个应用的崩溃。
* 是否有足够的灵活度。七层应用的优势是可以让整个应用的流量智能化，但是负载均衡设备需要提供完善的七层功能，满足客户根据不同情况的基于应用的调度。*最简单的一个考核就是能否取代后台Nginx或者Apache等服务器上的调度功能*。能够提供一个七层应用开发接口的负载均衡设备，可以让客户根据需求任意设定功能，才真正有可能提供强大的灵活性和智能性。

**三、负载均衡的算法？**

**1. 随机算法**

* Random随机，按权重设置随机概率。在一个截面上碰撞的概率高，但调用量越大分布越均匀，而且按概率使用权重后也比较均匀，有利于动态调整提供者权重。

**2. 轮询及加权轮询**

* 轮询(Round Robbin)当服务器群中各服务器的处理能力相同时，且每笔业务处理量差异不大时，最适合使用这种算法。 轮循，按公约后的权重设置轮循比率。存在慢的提供者累积请求问题，比如：第二台机器很慢，但没挂，当请求调到第二台时就卡在那，久而久之，所有请求都卡在调到第二台上。
* 加权轮询(Weighted Round Robbin)为轮询中的每台服务器附加一定权重的算法。比如服务器1权重1，服务器2权重2，服务器3权重3，则顺序为1-2-2-3-3-3-1-2-2-3-3-3- ......

**3. 最小连接及加权最小连接**

* 最少连接(Least Connections)在多个服务器中，与处理连接数(会话数)最少的服务器进行通信的算法。即使在每台服务器处理能力各不相同，每笔业务处理量也不相同的情况下，也能够在一定程度上降低服务器的负载。
* 加权最少连接(Weighted Least Connection)为最少连接算法中的每台服务器附加权重的算法，该算法事先为每台服务器分配处理连接的数量，并将客户端请求转至连接数最少的服务器上。

**4. 哈希算法**

* 普通哈希
* 一致性哈希一致性Hash，相同参数的请求总是发到同一提供者。当某一台提供者挂时，原本发往该提供者的请求，基于虚拟节点，平摊到其它提供者，不会引起剧烈变动。

**5. IP地址散列**

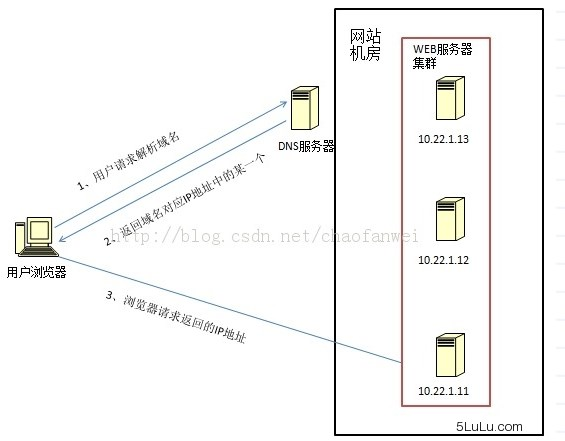
* 通过管理发送方IP和目的地IP地址的散列，将来自同一发送方的分组(或发送至同一目的地的分组)统一转发到相同服务器的算法。当客户端有一系列业务需要处理而必须和一个服务器反复通信时，该算法能够以流(会话)为单位，保证来自相同客户端的通信能够一直在同一服务器中进行处理。

**6.URL散列**

* 通过管理客户端请求URL信息的散列，将发送至相同URL的请求转发至同一服务器的算法。

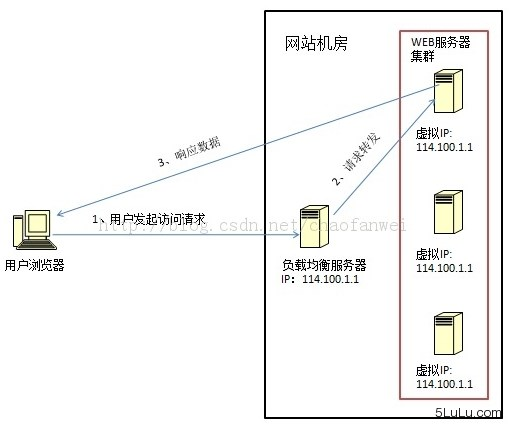
**四、负载均衡的实现（DNS > 数据链路层 > IP层 > Http层）？**

**1 - DNS域名解析负载均衡（延迟）**



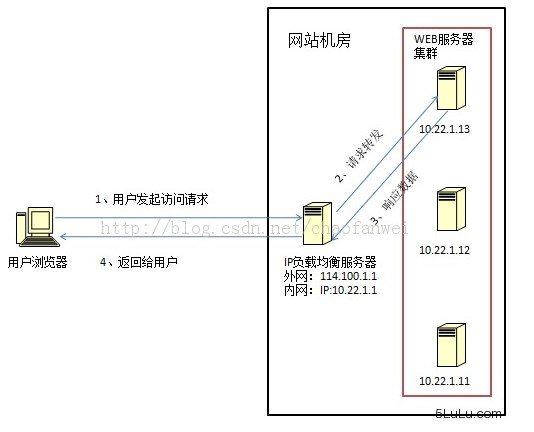
**利用DNS处理域名解析请求的同时进行负载均衡是另一种常用的方案**。在DNS服务器中配置多个A记录，如：www.mysite.com IN A 114.100.80.1、www.mysite.com IN A 114.100.80.2、www.mysite.com IN A 114.100.80.3.  
　**每次域名解析请求都会根据负载均衡算法计算一个不同的IP地址返回，这样A记录中配置的多个服务器就构成一个集群，并可以实现负载均衡。**　DNS域名解析负载均衡的优点是**将负载均衡工作交给DNS**，省略掉了网络管理的麻烦，缺点就是**DNS可能缓存A记录，不受网站控制**。事实上，大型网站总是部分使用DNS域名解析，作为第一级负载均衡手段，然后再在内部做第二级负载均衡。

**2 - 数据链路层负载均衡(LVS)**

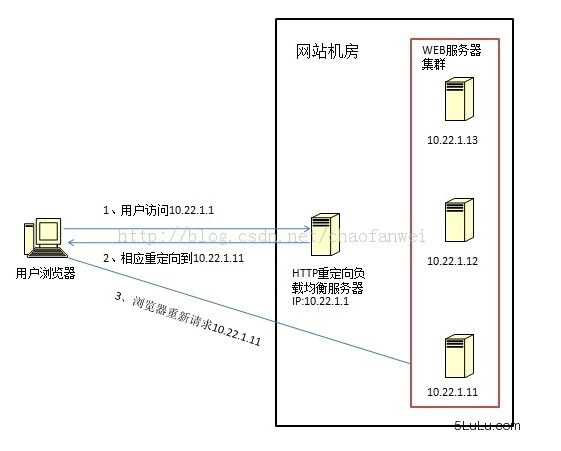


　数据链路层负载均衡是指在**通信协议的数据链路层修改mac地址进行负载均衡。**  
　这种数据传输方式又称作三角传输模式，负载均衡数据分发过程中不修改IP地址，**只修改目的的mac地址，**通过**配置真实物理服务器集群所有机器虚拟IP和负载均衡服务器IP地址一样，**从而达到负载均衡，这种负载均衡方式又称为直接路由方式（DR）.  
　在上图中，用户请求到达负载均衡服务器后，负载均衡服务器将请求数据的目的**mac地址修改为真实WEB服务器的mac地址**，并不修改数据包目标IP地址，因此数据可以正常到达目标WEB服务器，**该服务器在处理完数据后可以经过网管服务器而不是负载均衡服务器直接到达用户浏览器**。  
**使用三角传输模式的链路层负载均衡是目前大型网站所使用的最广的一种负载均衡手段。**在linux平台上最好的链路层负载均衡开源产品是LVS(linux virtual server)。

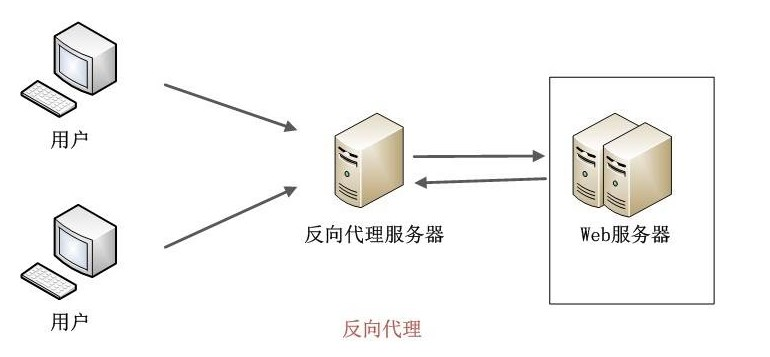
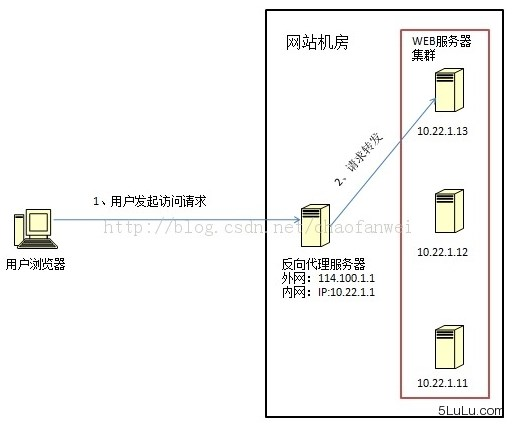
**3 - IP负载均衡(SNAT)**

  
　IP负载均衡：即**在网络层通过修改请求目标地址**进行负载均衡。  
　用户请求数据包到达负载均衡服务器后，负载均衡服务器在操作系统内核进行获取网络数据包，根据**负载均衡算法计算得到一台真实的WEB服务器地址**，然后将数据包的**IP地址修改为真实的WEB服务器地址**，不需要通过用户进程处理。真实的WEB服务器处理完毕后，相应数据包回到负载均衡服务器，负载均衡服务器再**将数据包源地址修改为自身的IP地址发送给用户浏览器。（代理）**　这里的关键在于真实WEB服务器相应数据包如何返回给负载均衡服务器，一种是负载均衡服务器在**修改目的IP地址的同时修改源地址**，将数据包源地址改为自身的IP，即源地址转换（SNAT），另一种方案是将负载均衡服务器同时作为真实物理服务器的网关服务器，这样所有的数据都会到达负载均衡服务器。  
　IP负载均衡在内核进程完成数据分发，较反向代理均衡有更好的处理性能。但由于所有请求响应的数据包都需要经过负载均衡服务器，**因此负载均衡的网卡带宽成为系统的瓶颈。**

**4 - HTTP重定向负载均衡(少见)**

  
　HTTP重定向服务器是一台普通的应用服务器，其唯一的功能就是根据用户的HTTP请求计算一台真实的服务器地址，并将真实的服务器地址写入HTTP重定向响应中（响应状态吗302）返回给浏览器，然后浏览器再自动请求真实的服务器。  
　这种负载均衡方案的优点是比较简单，缺点是浏览器需要每次请求两次服务器才能拿完成一次访问，性能较差；使用HTTP302响应码重定向，可能是搜索引擎判断为SEO作弊，降低搜索排名。重定向服务器自身的处理能力有可能成为瓶颈。因此这种方案在实际使用中并不见多。

**5 - 反向代理负载均衡(nginx)**

  
　传统代理服务器位于浏览器一端，代理浏览器将HTTP请求发送到互联网上。而**反向代理服务器则位于网站机房一侧，代理网站web服务器接收http请求**。  
　反向代理的作用是保护网站安全，所有互联网的请求都必须经过代理服务器，相当于**在web服务器和可能的网络攻击之间建立了一个屏障**。  
　除此之外，代理服务器也可以配置缓存加速web请求。当用户第一次访问静态内容的时候，**静态内存就被缓存在反向代理服务器上**，这样当其他用户访问该静态内容时，就可以直接从反向代理服务器返回**，加速web请求响应速度，减轻web服务器负载压力。**　另外，反向代理服务器也可以实现负载均衡的功能。  
  
　由于反向代理服务器转发请求在HTTP协议层面，因此也叫**应用层负载均衡**。优点是部署简单，缺点是可能成为系统的瓶颈。