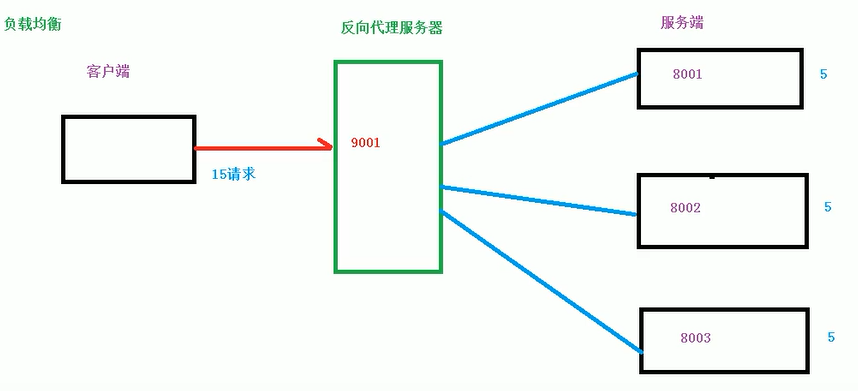
[什么是负载均衡？](https://www.cnblogs.com/fanBlog/p/10936190.html)

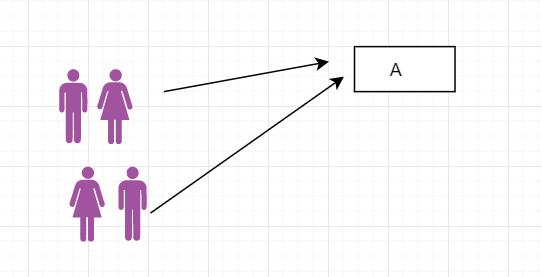
1. **什么是负载均衡**



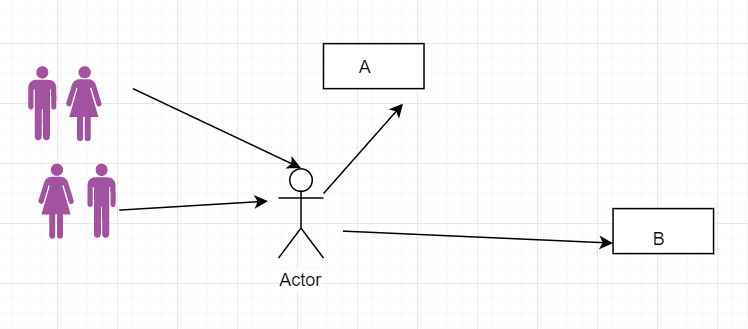
Load balancing，即负载均衡，是一种计算机技术，用来在多个计算机（计算机集群）、网络连接、CPU、磁盘驱动器或其他资源中分配负载，以达到最优化资源使用、最大化吞吐率、最小化响应时间、同时避免过载的目的。

**2.为什么需要负载均衡**

我们在日常生活中经常免不了要去一些比较拥挤的地方，比如地铁站、火车站、电影院、银行等。无论是买票，还是排队入场，这些场所一般都会设置多个服务点或者入口的。如果没有人引导的话，大多数情况下，最近的入口会挤满人。而哪些距离较远的服务点或者入口就宽松很多。



 这种情况下，就会大大浪费资源，因为如果可以把这些排队的人很好的分散到各个入口的话会大大缩短排队时间。其实，网站的建设也是一样的。为了提升网站的服务能力，很多网站采用集群部署，就像话剧院有多个入口一样。这时候，就需要一个协调者，来均衡的分配这些用户的请求，可以让用户的可以均匀的分派到不同的服务器上。

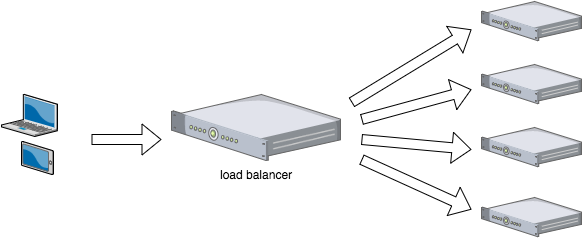


 （图挺丑的，但是不想在画图上浪费太多时间~~）

**什么是负载均衡**  
在早高峰乘地铁时候，紧挨小区的地铁口人特别多，一般会有限流，还会有个地铁工作人员D那个大喇叭在喊“着急的人员请走B口，B口人少车空”。。。

而这个地铁工作人员D就是负责负载均衡的**。**为了提升网站的各方面能力，我们一般会把多台机器组成一个集群对外提供服务。然而，我们的网站对外提供的访问入口都是一个的，比如www.taobao.com。那么当用户在浏览器输入www.taobao.com的时候如何将用户的请求分发到集群中不同的机器上呢，这就是负载均衡在做的事情。

￼**负载均衡（Load Balance），意思是将负载（工作任务，访问请求）进行平衡、分摊到多个操作单元（服务器，组件）上进行执行。是解决高性能，单点故障（高可用），扩展性（水平伸缩）的终极解决方案。**

****

**3.负载均衡分类**

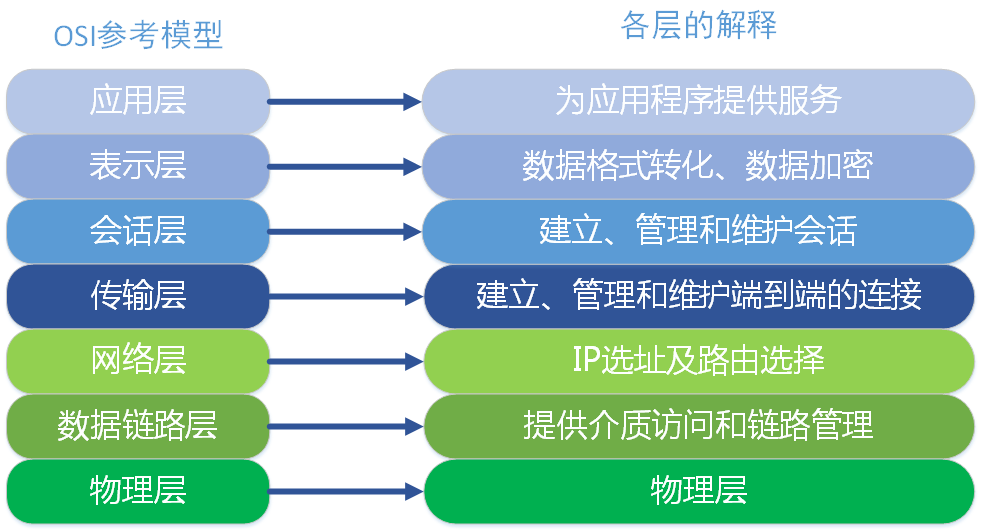
先看下OSI七层模型

**OSI是一个开放性的通信系统互连参考模型，他是一个定义得非常好的协议规范。**

OSI模型有7层结构，每层都可以有几个子层。 OSI的7层从上到下分别是 7、应用层；6、表示层；5、会话层；4、传输层；3、网络层；2、数据链路层；1、物理层；

其中高层（即7、6、5、4层）定义了应用程序的功能，下面3层（即3、2、1层）主要面向通过网络的端到端的数据流。

在这七层模型种，高层次都是依赖于低层次的。层次越高，使用起来越方便。



计算机网络有关的概念：

TELNET、HTTP、FTP、NFS、SMTP、DNS等属于第七层应用层的概念。

TCP、UDP、SPX等属于第四层传输层的概念。

IP、IPX等属于第三层网络层的概念。

ATM、FDDI等属于第二层数据链路层的概念。

了解了网络协议的七层模型以后，再来看看负载均衡。我们可以很明确的一点是，**负载均衡是要在网络传输中做文章的**。而要在网络传输过程搞事情，那么这七层模型就势必躲不开。

所以，根据负载均衡技术实现在OSI七层模型的不同层次，是可以给负载均衡分类的。

常见的实现方式中，主要可以在应用层、传输层、网络层和数据传输层做文章。所以，工作在应用层的负载均衡，我们通常称之为七层负载均衡、工作在传输层的我们称之为四层负载均衡。

大致可以分为以下几种，其中最常用的是四层和七层负载均衡：

**二层负载均衡**

负载均衡服务器对外依然提供一个VIP（虚IP），集群中不同的机器采用相同IP地址，但是机器的MAC地址不一样。当负载均衡服务器接受到请求之后，通过改写报文的目标MAC地址的方式将请求转发到目标机器实现负载均衡。

**三层负载均衡**

和二层负载均衡类似，负载均衡服务器对外依然提供一个VIP（虚IP），但是集群中不同的机器采用不同的IP地址。当负载均衡服务器接受到请求之后，根据不同的负载均衡算法，通过IP将请求转发至不同的真实服务器。

**四层负载均衡**

四层负载均衡工作在OSI模型的传输层，由于在传输层，只有TCP/UDP协议，这两种协议中除了包含源IP、目标IP以外，还包含源端口号及目的端口号。四层负载均衡服务器在接受到客户端请求后，以后通过修改数据包的地址信息（IP+端口号）将流量转发到应用服务器。

**七层负载均衡**

七层负载均衡工作在OSI模型的应用层，应用层协议较多，常用http、radius、dns等。七层负载就可以基于这些协议来负载。这些应用层协议中会包含很多有意义的内容。比如同一个Web服务器的负载均衡，除了根据IP加端口进行负载外，还可根据七层的URL、浏览器类别、语言来决定是否要进行负载均衡。

**负载均衡工具**

市面上有很多开源的负载均衡的工具或软件，基本都是基于前面提到的方案实现的，大多数是工作在第七层和第四层的。Nginx/LVS/HAProxy是目前使用最广泛的三种负载均衡软件。

**LVS** ：**LVS主要用来做四层负载均衡**

LVS（Linux Virtual Server），也就是Linux虚拟服务器, 是一个由章文嵩博士发起的自由软件项目。使用LVS技术要达到的目标是：通过LVS提供的负载均衡技术和Linux操作系统实现一个高性能、高可用的服务器群集，它具有良好可靠性、可扩展性和可操作性。从而以低廉的成本实现最优的服务性能。

**Nginx** ：**Nginx主要用来做七层负载均衡**

Nginx（发音同engine x）是一个网页服务器，它能反向代理HTTP, HTTPS, SMTP, POP3, IMAP的协议链接，以及一个负载均衡器和一个HTTP缓存。**。**

**HAProxy** ：**HAProxy主要用来做七层负载均衡**

HAProxy是一个使用C语言编写的自由及开放源代码软件，其提供高可用性、负载均衡，以及基于TCP和HTTP的应用程序代理。

**4.负载均衡算法**

负载均衡服务器在决定将请求转发到具体哪台真实服务器的时候，是通过负载均衡算法来实现的。负载均衡算法，是一个负载均衡服务器的核心。

就像电影院门口的引导员一样，他根据什么把排队人员分配到具体的入口呢？是哪个入口人少吗？还是哪个入口速度最快？还是哪个入口最近呢？

负载均衡算法可以分为两类：静态负载均衡算法和动态负载均衡算法。

1）.静态负载均衡算法包括：轮询，比率，优先权

* **轮询（Round Robin）：**顺序循环将请求一次顺序循环地连接每个服务器。当其中某个服务器发生第二到第7 层的故障，BIG-IP 就把其从顺序循环队列中拿出，不参加下一次的轮询，直到其恢复正常。
* **比率（Ratio）：**给每个服务器分配一个加权值为比例，根椐这个比例，把用户的请求分配到每个服务器。当其中某个服务器发生第二到第7 层的故障，BIG-IP 就把其从服务器队列中拿出，不参加下一次的用户请求的分配, 直到其恢复正常。
* **优先权（Priority）：**给所有服务器分组,给每个组定义优先权，BIG-IP 用户的请求，分配给优先级最高的服务器组（在同一组内，采用轮询或比率算法，分配用户的请求）；当最高优先级中所有服务器出现故障，BIG-IP 才将请求送给次优先级的服务器组。这种方式，实际为用户提供一种热备份的方式。

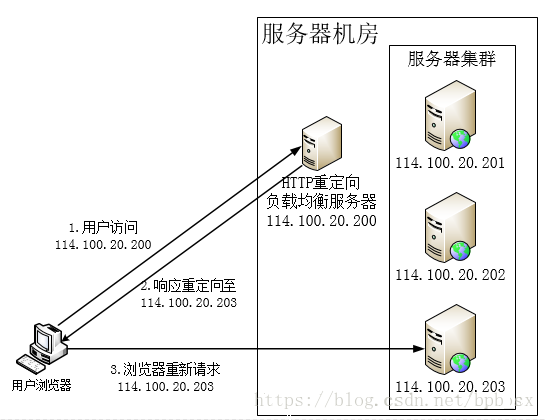
2).动态负载均衡算法包括: 最少连接数,最快响应速度，观察方法，预测法，动态性能分配，动态服务器补充，服务质量，服务类型，规则模式。

* **最少的连接方式（Least Connection）：**传递新的连接给那些进行最少连接处理的服务器。当其中某个服务器发生第二到第7 层的故障，BIG-IP 就把其从服务器队列中拿出，不参加下一次的用户请求的分配, 直到其恢复正常。
* **最快模式（Fastest）：**传递连接给那些响应最快的服务器。当其中某个服务器发生第二到第7 层的故障，BIG-IP 就把其从服务器队列中拿出，不参加下一次的用户请求的分配，直到其恢复正常。
* **观察模式（Observed）：**连接数目和响应时间以这两项的最佳平衡为依据为新的请求选择服务器。当其中某个服务器发生第二到第7 层的故障，BIG-IP就把其从服务器队列中拿出，不参加下一次的用户请求的分配，直到其恢复正常。
* **预测模式（Predictive）：**BIG-IP利用收集到的服务器当前的性能指标，进行预测分析，选择一台服务器在下一个时间片内，其性能将达到最佳的服务器相应用户的请求。(被BIG-IP 进行检测)
* **动态性能分配(Dynamic Ratio-APM)：**BIG-IP 收集到的应用程序和应用服务器的各项性能参数，动态调整流量分配。
* **动态服务器补充(Dynamic Server Act.)：**当主服务器群中因故障导致数量减少时，动态地将备份服务器补充至主服务器群。
* **服务质量(QoS）：**按不同的优先级对数据流进行分配。
* **服务类型(ToS)：**按不同的服务类型（在Type of Field中标识）负载均衡对数据流进行分配。
* **规则模式：**针对不同的数据流设置导向规则，用户可自行。

以上，就是目前实现负载均衡的主流算法。不同的负载均衡服务器会选择不同的算法。就像电影院和火车站可能会选用不同的引导策略一样。火车站可能会把行李少的旅客分配到一个专门的入口，可能给即将发车的旅客分派到特快入口，手持可扫描车票的用户单独分配到特殊入口等。

**几种常见的负载均衡**

1）http重定向



HTTP重定向服务器就是一个普通的服务器，当用户访问时，其会根据一定的算法得到服务器集群的一个真实服务器的IP地址，将其放在HTTP响应头中，响应状态码为（302），当用户浏览器接收到这个响应时，会将得到的真实服务器的IP地址提出并重新访问。如上图所示，当用户访问域名时通过DNS解析得到114.100.20.200，然后访问114.100.20.200，也就是HTTP重定向服务器，响应重定向至114.100.20.203，用户浏览器再重新访问。

缺点:

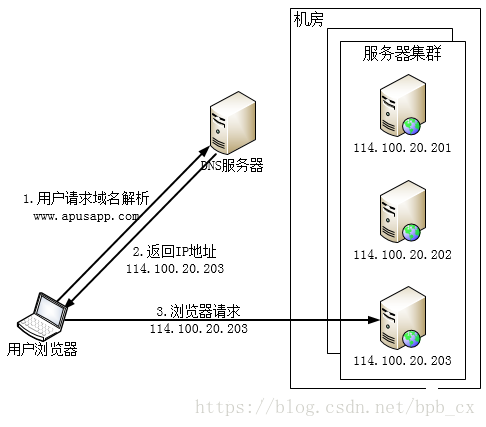
1. 这种方式需要用户浏览器访问两次，性能较差

2. HTTP重定向服务器会的处理能力会成为负载均衡的瓶颈

3. HTTP重定向返回302，可能会使搜索引擎判定为SEO作弊，降低搜索排名

2）dns域名解析（DNS:通过域名获取ip地址的手段）

当我们通过域名访问网站时，需要通过DNS服务器得到服务器的IP地址，我们可以在DNS服务器上设置一定的算法，每次得到不同的IP地址来进行访问从而实现负载均衡



如图，当用户访问www.apusapp.com时，这个域名对应了多个IP地址，通过DNS服务器解析会得到一个IP地址（可以看到，得到的这个IP地址，是服务器集群中一个服务器的IP地址），用户访问这个IP地址来达到真实的服务。

优点：

将负载均衡的工作丢给了DNS服务器去做，省去了网站管理人员的维护工作

对于真实地址的服务器，不需要做任何的配置

简单易用，成本低，而且方便灵活

服务器可以放在任何的地方

同时，DNS服务还可以做基于地理位置的解析，可以让一个距离最近的服务器的IP地址放回，提高性能

缺点:

1.DNS服务是有多级的（之后有时间写一个详细的DNS服务介绍）

大致上来说，首先是在浏览器中有一个DNS缓存，如果找不到就在本机地址的hosts文件中查找，再找不到就去路由器缓存中查找。然后是本地DNS服务器，如果没有，就是根服务器，顶级服务器，权限域名服务器等等等

总之，在每一级都有可能缓存这DNS的对应关系，所以有可能当某一台真实服务器下线之后，修改了DNS服务器的记录，但在生效之前还有一段时间，在这段期间，其IP地址已经不可用了，通过域名进行访问时还是会访问到这个IP地址。会访问失败

2.DNS服务器和真实服务器是完全分开的，所以DNS的负载均衡不能监测到真是服务器当前的运行状态，其负载均衡的效果不是很好

3.可能会造成额外的网络问题。为了使本DNS服务器和其他DNS服务器及时交互，保证DNS数据及时更新，使地址能随机分配，一般都要将DNS的刷新时间设置的较小，但太小将会使DNS流量大增造成额外的网络问题。

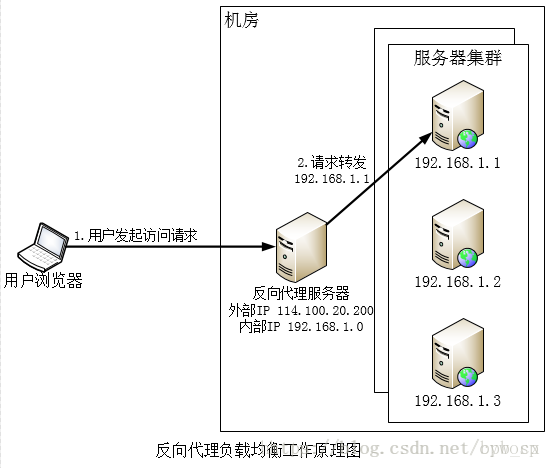
事实上，大型网站都将DNS负载均衡作为第一级的负载均衡手段，在服务器内部再进行第二级的负载均衡，也就是说，我们通过DNS得到的IP地址并不是真实服务器的IP地址，而是内部负载均衡服务器的IP地址。

3）反向代理

代理与反向代理：VPN服务就是我们常用的一种代理（正向代理），用户将请教交给代理服务器，代理服务器访问网站获取数据，之后代理服务器再将数据返还给用户。在这个过程中，应用服务器并不知道用户的存在。只知道代理浏览器的访问。

反向代理是指在服务器端的代理，代理服务器接收用户的请求，再转发给真实服务器，之后再返回给代理服务器再给用户，在这个过程中，用户并不知道真实服务器的存在。

反向代理服务器管理了一组服务器，当用户访问时，代理服务器根据负载均衡算法将请求转发到真实服务器，真实服务器也通过反向代理服务器返还数据。内部服务器不对外部提供服务，所以不需要外部IP，而反向代理服务器需要两个网卡，一个IP用于外部用户访问使用，另外一个用于内部使用。



如上图所示，当用户发起访问，请求访问的ip地址是114.100.20.200，到达反向代理服务器时，根据负载均衡算法得到一个真实服务器的IP地址，并将用户请求转发到该服务器上，当真实服务器处理完之后将数据返回到反向代理服务器。反相代理服务器再将该响应的内容返回给用户。

优点：

反向代理服务器位于应用层，负载均衡方案和反向代理服务器集成在了一起，部署简单

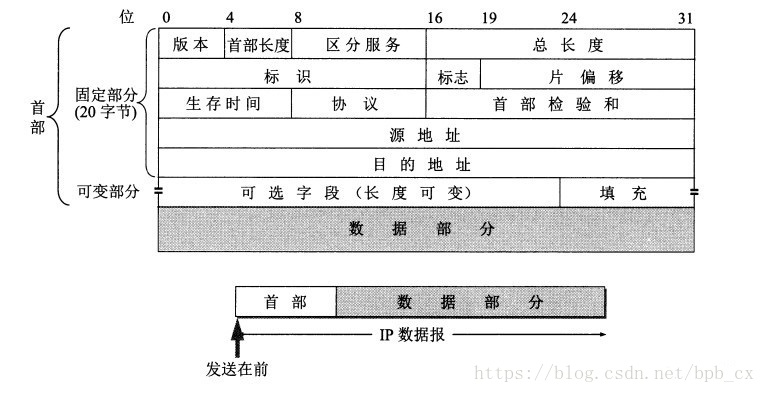
缺点：

反向代理服务器用户处理所有的请求和响应，其性能可能成为服务器集群的瓶颈

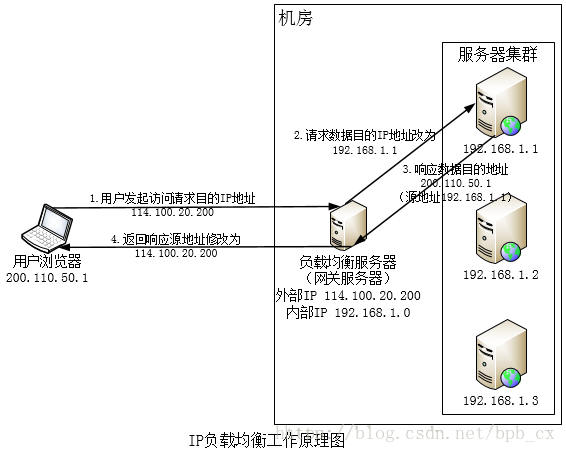
有名Nginx就是反向代理服务

4）IP层负载均衡

IP包结构



其中，可能看到有源地址和目的地址两项，这两项就是用来做IP层的负载均衡的关键。我们就是通过修改这两个地址来达到“转发”目的



如上图所示，当用户发起请求时（源地址为200.110.50.1），访问负载均衡服务器（目的地址为114.100.20.200），负载均衡服务器在内核进程获取网路数据包，根据一定的算法得到一个真实服务器的IP地址，其后将IP数据包的目的地址修改为该IP地址（192.168.1.1），之后就会将数据包发送到该真实服务器上去，之后再向负载均衡服务器返回数据，负载均衡服务器将源地址修改为114.100.20.200后返回给用户浏览器。

这个方法的关键就是因为只能在负载均衡服务器出修改源地址和目的地址，所以在真实服务器处理完之后要想办法将数据返回给负载均衡服务器而不是用户浏览器。

比如，当用户发出请求时，目的地址是114.100.20.200，源地址是200.110.50.1，到达负载均衡服务器后，将目的地址该位192.168.1.1，源地址还是200.110.50.1，所以当真实服务器处理完之后的数据无法回到负载均衡服务器

解决方法：

提供两个网卡，负载均衡服务器就有内部IP和外部IP两个IP，当请求到达负载均衡服务器时，修改目的地址，也修改源地址，将源地址修改为负载均衡服务器的内部IP，这样的话，真实服务器处理后的响应就会再次回到负载均衡服务器

将负载均衡服务器作为真实服务器集群的网关服务器，这样的话所有的请求响应都要经过网关服务器

优点:

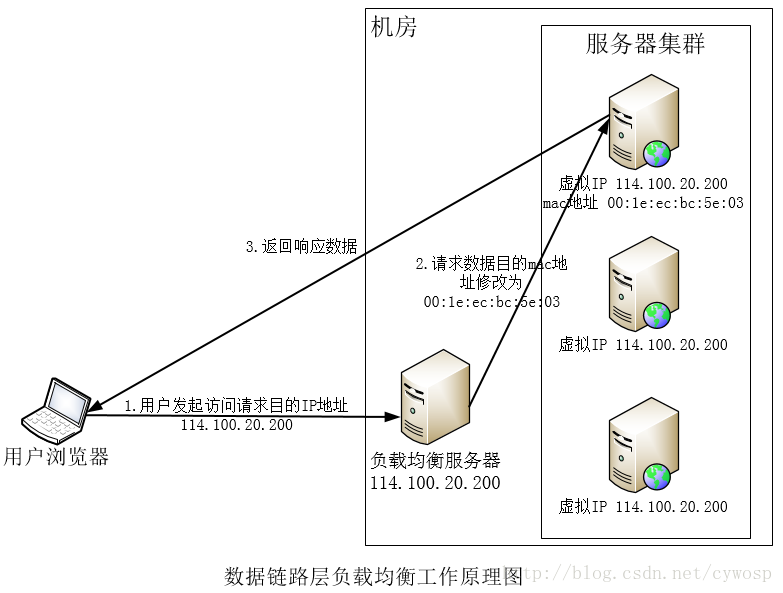
IP负载均衡在内核进程完成数据分发，处理性能得到了很好的提高。

缺点:

由于所有请求和响应都要经过负载均衡服务器，集群的最大响应数据吞吐量将受到负载均衡服务器网卡带宽的限制，对于提供下载服务或者视频服务等需要大量传输数据的站点而言，这是难以满足需求的

5）链路层的负载均衡

MAC地址：mac地址是与网卡相关，其编号只与网卡生产厂商和流水号有关，基本上可以作为每台电脑的“身份证”。以太网中数据帧之间是通过MAC寻址来到达对应的计算机网卡或者路由的



链路层的负载均衡通过修改帧数据包中的MAC地址来达到转发的目的。这种方法，所有的真实服务器和负载均衡服务器都有相同的IP地址，不用修改IP数据包的目的地址和源地址，只通过修改MAC地址就可以达到效果，因为请求的IP地址和实际处理的真实服务器的IP地址一致，所以不需要回到负载均衡服务器进行地址交换，可以将响应直接发会给用户浏览器，避免了负载均衡服务器成为传输瓶颈的可能