

5 负载均衡策略与算法剖析

2019年5月27日 10:02

0 负载均衡按功能分类

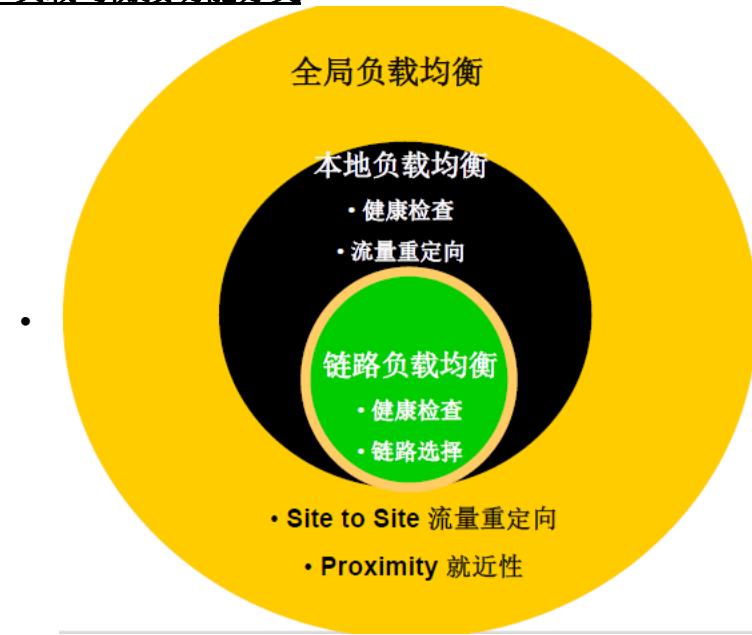
1 负载均衡策略

- 基于DNS的负载均衡
- 基于反向代理的负载均衡
- 基于特定服务器软件的负载均衡
- 基于NAT的负载均衡
- 基于CDN的负载均衡

2 负载均衡算法

- 轮询算法
- Hash散列算法
- 最少链接算法
- 最快链接算法

0 负载均衡按功能分类



本地/全局负载均衡

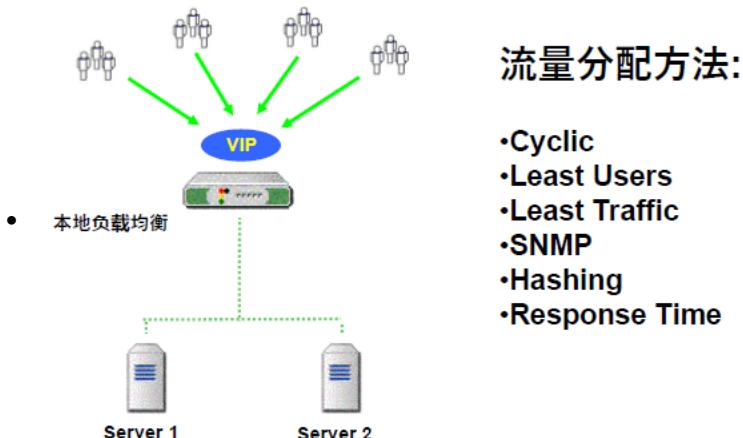
- 负载均衡从其应用的地理结构上分为本地负载均衡 (LocalLoadBalance) 和全局负载均衡 (GlobalLoadBalance, 也叫地域负载均衡)，本地负载均衡是指对本地的服务器群做负载均衡，全局负载均衡是指对分别放置在不同的地理位置、有不同网络结构的服务器群间作负载均衡。
- 本地负载均衡能有效地解决数据流量过大、网络负荷过重的问题，并且不需购置性能卓越的服务器，充分利用现有设备，避免服务器单点故障造成数据流量的损失。
- 其有灵活多样的均衡策略把数据流量合理地分配给服务器群内的服务器共同负担。扩充升级方便，增加一个新的服务器到服务群中，而不需改变现有网络结构、停止现有的服务。

本地负载均衡

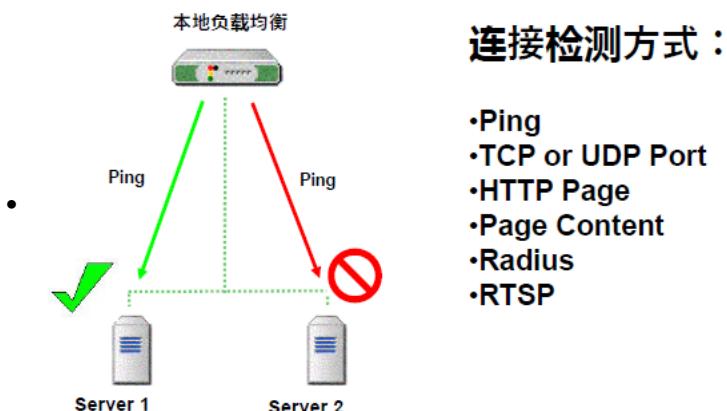




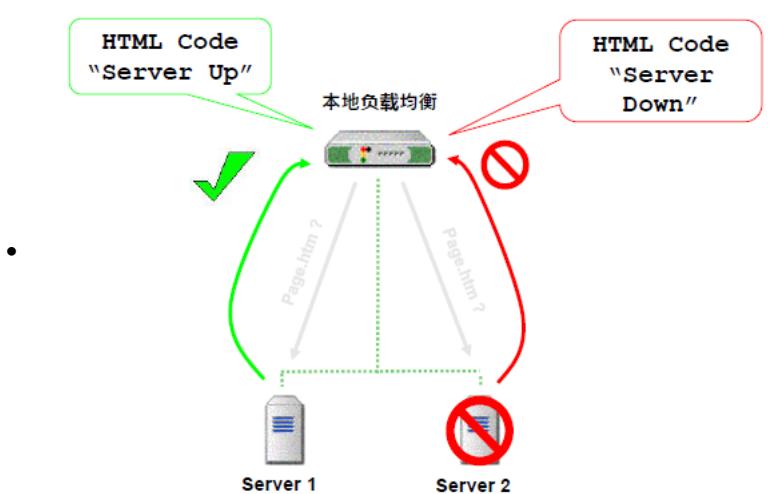
流量导向分配方法



连接检测方式

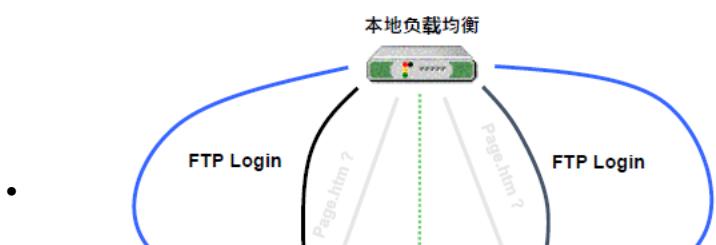


页面和内容检测



高级健康检查

检测同一个服务器上的多个应用：





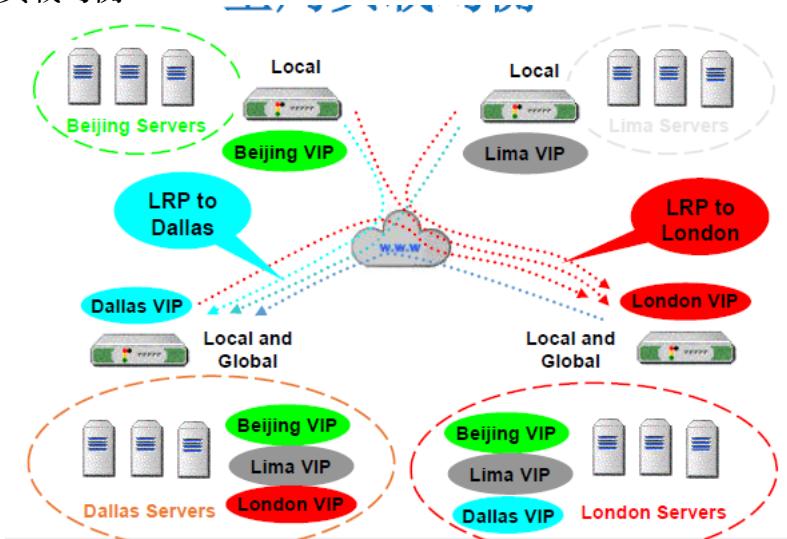
本地负载均衡能够基于以下七层信息进行流量导向

- URLs
- HTTP header information
- File type
- Browser Type
- Session ID
- Regular Expression within the HTTP Header
- Cookies

本地/全局负载均衡

- 全局负载均衡主要用于在一个多区域拥有自己服务器的站点，为了使全球用户只以一个IP地址或域名就能访问到离自己最近的服务器，从而获得最快的访问速度，也可用于子公司分散站点分布广的大公司通过 Intranet（企业内部互联网）来达到资源统一合理分配的目的。
- 全局负载均衡有以下的特点：
 - 实现地理位置无关性，能够远距离为用户提供完全的透明服务
 - 可避免服务器、数据中心等单点失效，可避免由于 ISP专线故障引起的单点失效。
 - 解决网络拥塞问题，提高服务器响应速度，服务就近提供，达到更好的访问质量。

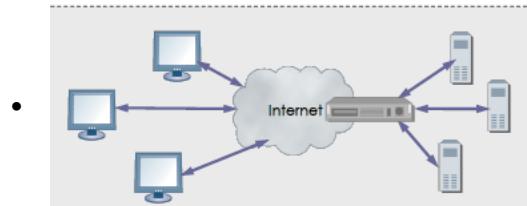
全局负载均衡



为什么需要负载均衡？

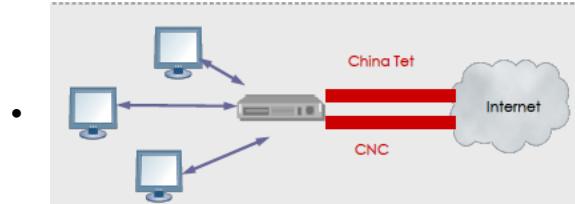


解决方案：服务器负载均衡



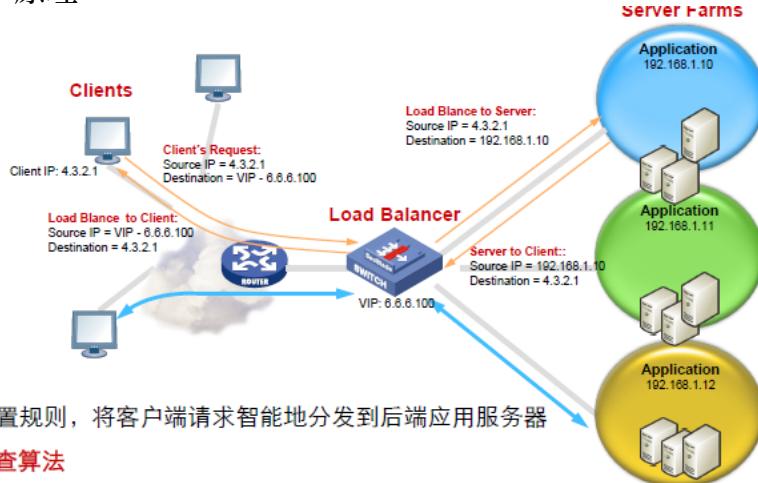
- 多台服务器组成一个群组，它们通过网络设备相连接。这些服务器提供相同或相似的网络服务。服务器群组前布局一个负载均衡设备，负责根据已配置均衡策略将用户请求在服务器群组中的分发，为用户提供服务，并对服务器可用性的维护。

解决方案：链路负载均衡



- 通过带宽或就近性等算法，在多条链路中进行负载均衡，选择最优的链路，提高访问速度。

负载均衡基本原理



调度算法

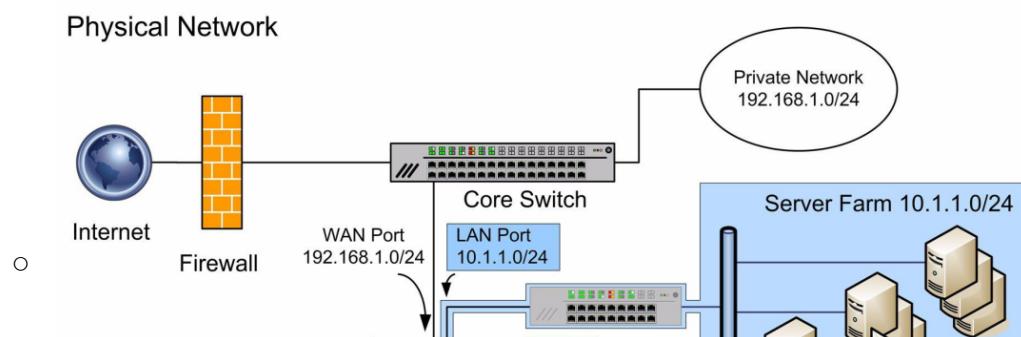
- 根据配置规则，将客户端请求智能地分发到后端应用服务器

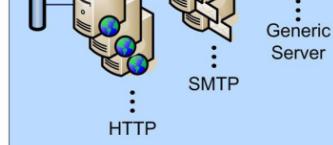
健康性检查算法

- 实时监控服务器运行状态

部署方式

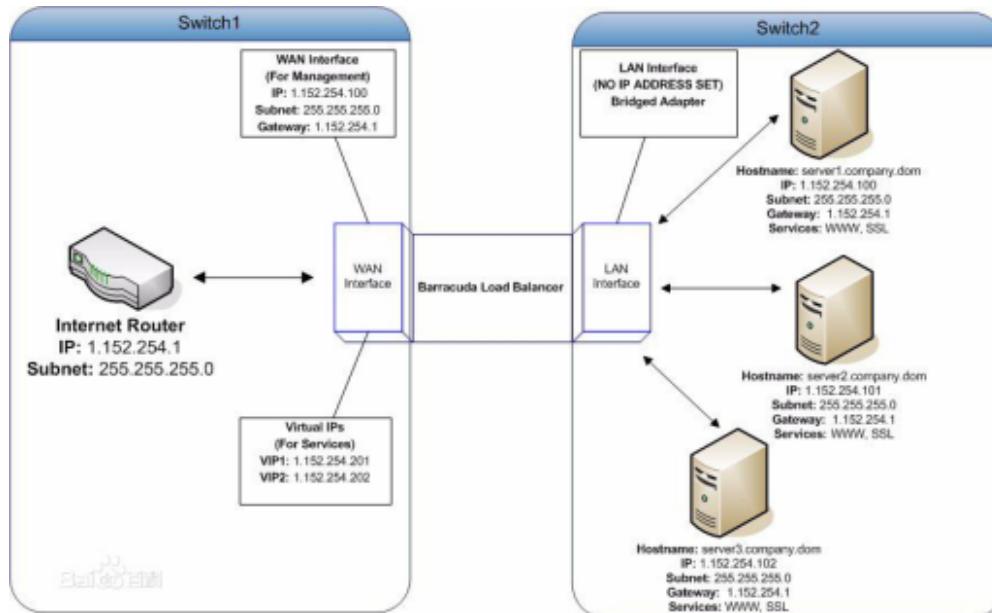
- 负载均衡有三种部署方式：路由模式、桥接模式、服务直接返回模式。
 - 路由模式部署灵活，约60%的用户采用这种方式部署；
 - 桥接模式不改变现有的网络架构；
 - 服务直接返回（DSR）比较适合吞吐量大特别是内容分发的网络应用。约30%的用户采用这种模式
- 路由模式：路由模式的部署方式如右图。服务器的网关必须设置成负载均衡机的LAN口地址，且与WAN口分署不同的逻辑网络。因此所有返回的流量也都经过负载均衡。这种方式对网络的改动小，能均衡任何下行流量



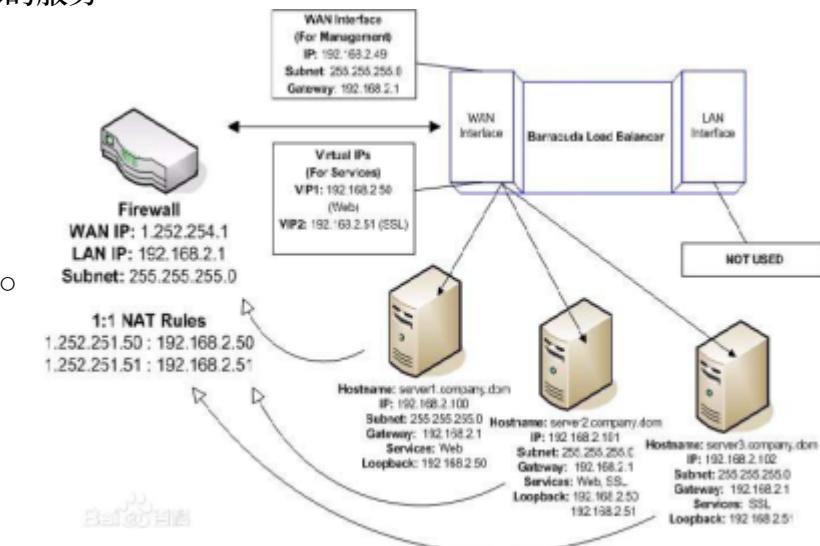


- 桥接模式：配置简单，不改变现有网络。负载均衡的WAN口和LAN口分别连接上行设备和下行服务器。LAN口不需要配置IP（WAN口与LAN口是桥连接），所有的服务器与负载均衡均在同一逻辑网络中。

参见右图：由于这种安装方式容错性差，网络架构缺乏弹性，对广播风暴及其他生成树协议循环相关联的错误敏感，因此一般不推荐这种安装架构



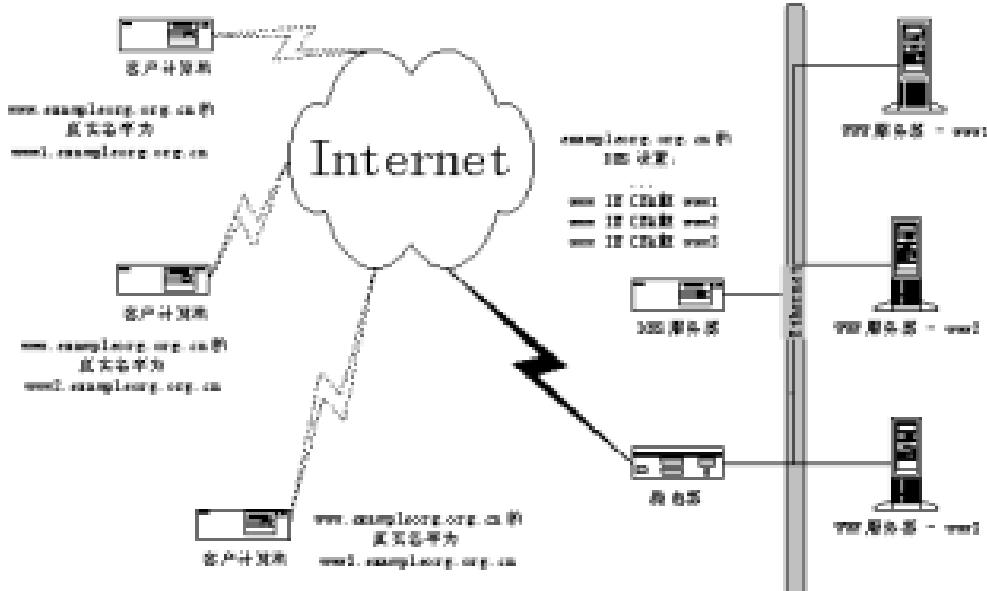
- 服务直接返回模式：这种安装方式负载均衡的LAN口不使用，WAN口与服务器在同一个网络中，互联网的客户端访问负载均衡的虚IP（VIP），虚IP对应负载均衡机的WAN口，负载均衡根据策略将流量分发到服务器上，服务器直接响应客户端的请求。因此对于客户端而言，响应他的IP不是负载均衡机的虚IP（VIP），而是服务器自身的IP地址。返回的流量不经过负载均衡。因此这种方式适用大流量高带宽要求的服务



1 负载均衡策略

- [基于DNS的负载均衡](#)
- [基于反向代理的负载均衡](#)
- [基于特定服务器软件的负载均衡](#)
- [基于NAT的负载均衡](#)
- [基于CDN的负载均衡](#)

- 实现原理：一个域名绑定多个IP，通过DNS服务中的随机域名解析来实现



- 优点：

- 实现简单、实施容易、成本低、适用于大多数TCP/IP应用。

- 问题：

- 一旦某个服务器出现故障，即使及时修改了DNS设置，还是要等待足够的时间（刷新时间）才能发挥作用，在此期间保存了故障服务器地址的客户计算机将不能正常访问服务器。

- 缺陷

- DNS负载均衡无法得知服务器之间的差异，它不能做到为性能较好的服务器多分配请求，也不能了解到服务器的当前状态，甚至会出现客户请求集中在某一台服务器上的偶然情况。

基于反向代理的负载均衡

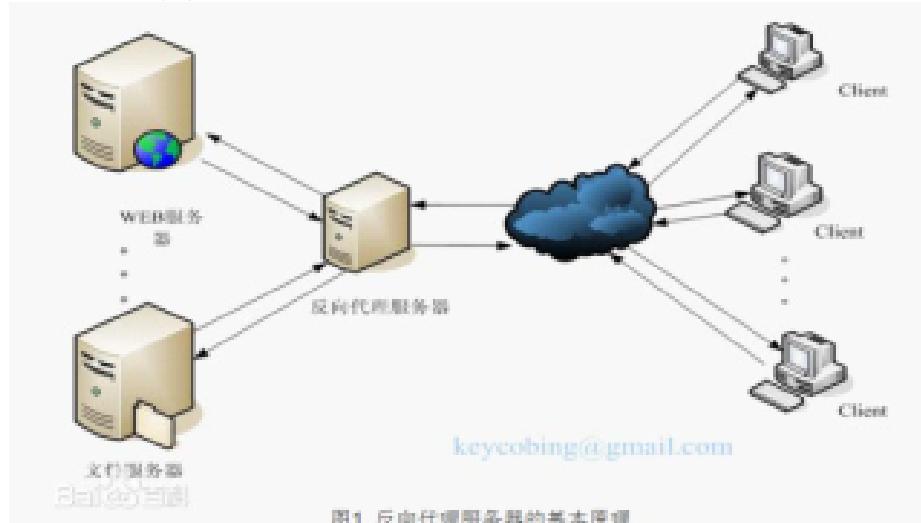


图1 反向代理服务器的基本原理

- 优点：

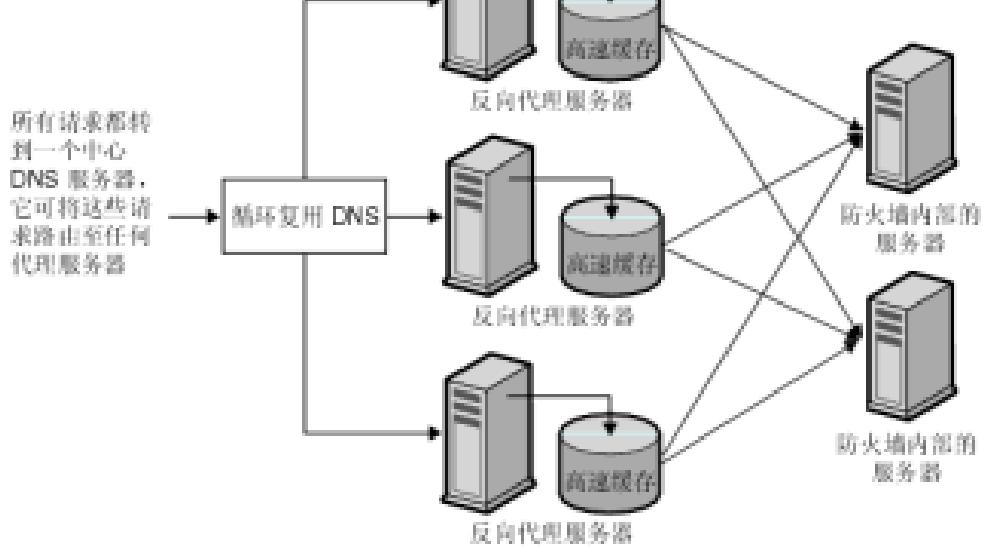
- 自带高速缓冲，可减轻内容服务器压力，提速网络访问效率。

- 问题：

- 针对每一次代理，代理服务器就必须打开两个连接，一个对外，一个对内，因此在并发连接请求数量非常大的时候，代理服务器的负载也就非常大，最后代理服务器本身可能会成为服务的瓶颈。

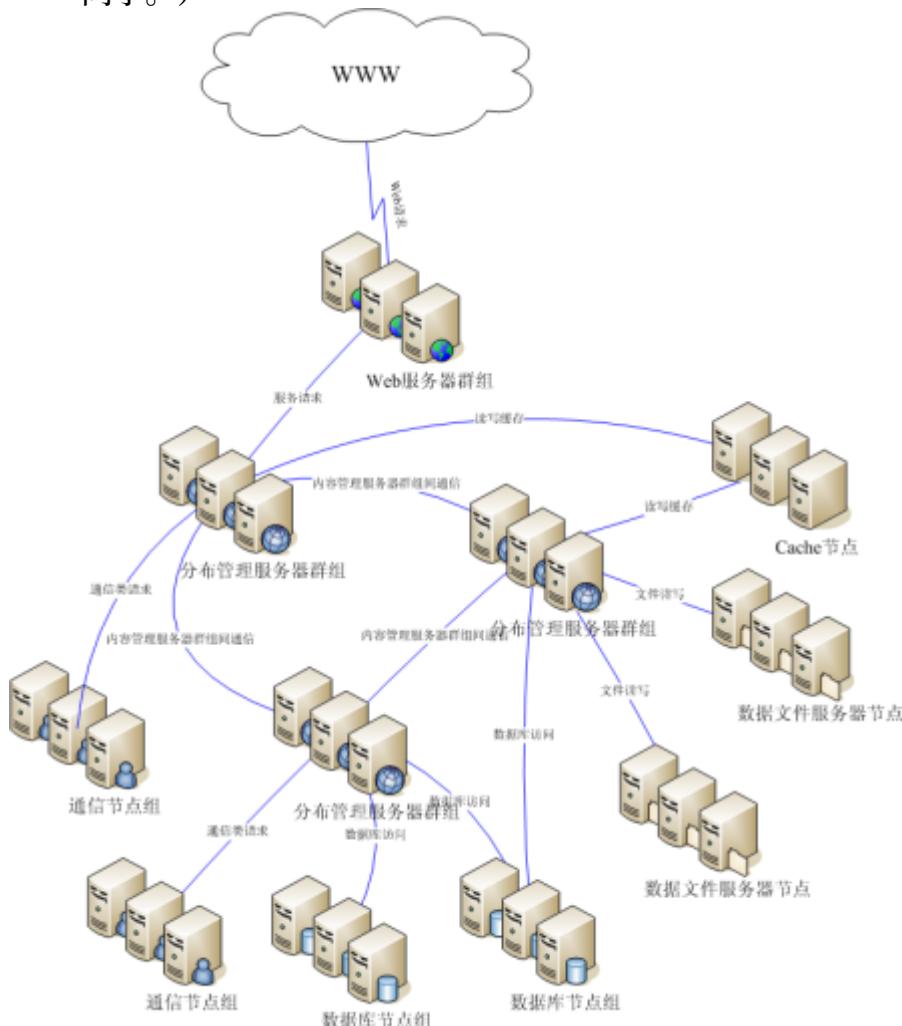
- 缺陷

- 反向代理是处于OSI参考模型第七层应用的，所以就必须为每一种应用服务专门开发一个反向代理服务器，这样就限制了反向代理负载均衡技术的应用范围，现在一般都用于对web服务器的负载均衡。



基于特定服务器软件的负载均衡

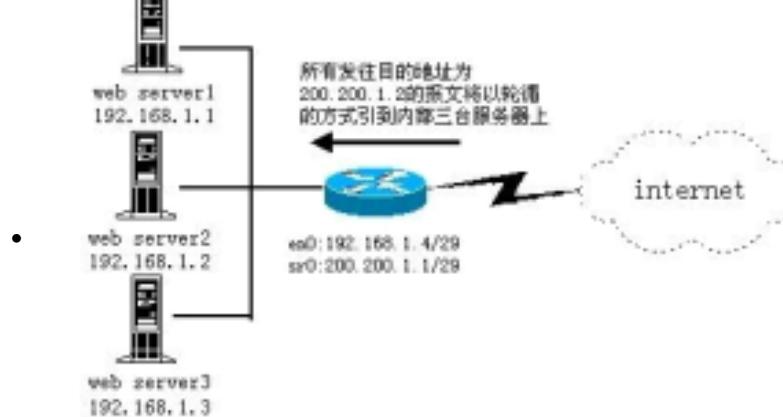
- 实现原理：
 - 利用网络协议的重定向功能来实现。
 - HTTP重定向：服务器无法处理浏览器发送过来的请求（request），服务器告诉浏览器跳转到可以处理请求的url上。（浏览器会自动访问该URL地址，以至于用户无法分辨是否重定向了。）



- 优点：
 - 服务可定制，可依据底层服务器的性能及实况进行负载调控。
- 问题：
 - 需要改动软件，成本较高。

基于NAT的负载均衡

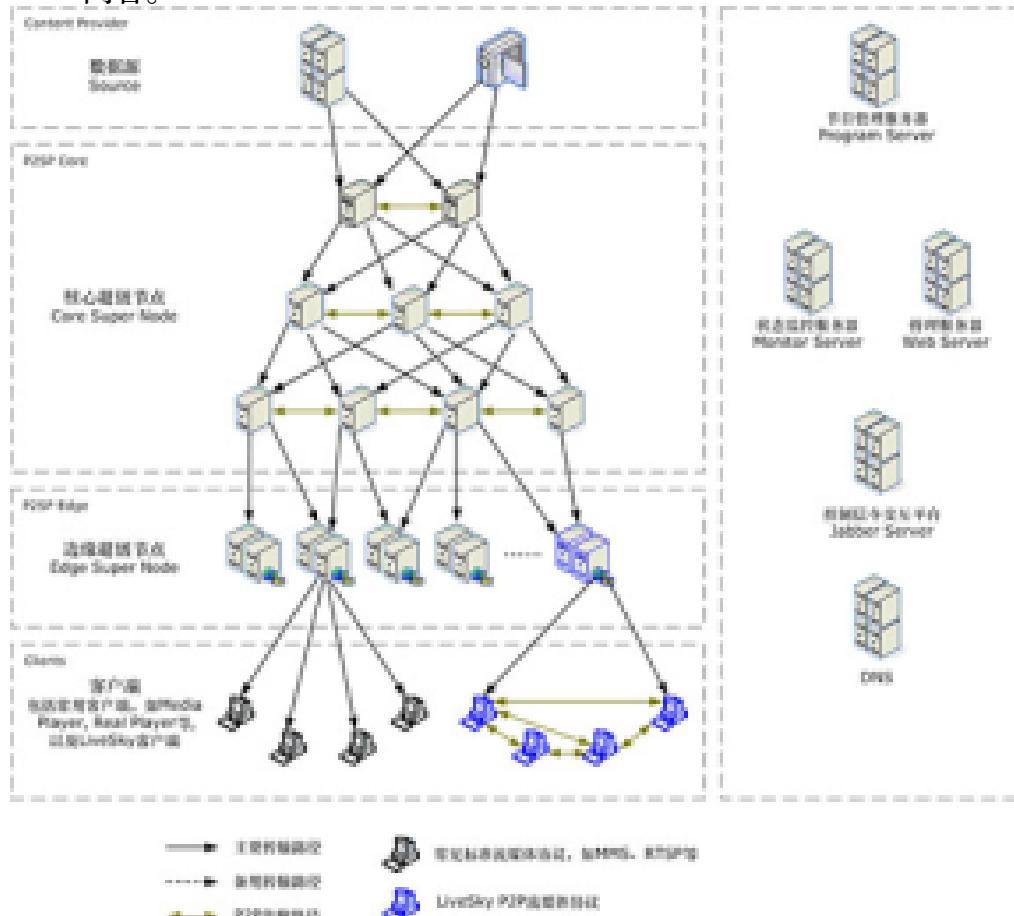
- 实现原理：
 - 将一个外部IP地址映射为多个内部IP地址。



- 优点:
 - 比较完善的负载均衡技术，均衡算法也较灵活，如随机选择、最少连接数及响应时间等来分配负载。
- 问题:
 - 伸缩能力有限，当服务器结点数目过多时，调度器本身有可能成为系统的新瓶颈。

基于CDN的负载均衡

- 实现原理:
 - 通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到最接近用户的网络“边缘”，使用户可以就近取得所需的内容。



- 优点:
 - 用户访问就近服务器，提高访问速度

2 负载均衡算法

- [轮询算法](#)
- [Hash散列算法](#)
- [最少链接算法](#)
- [最快链接算法](#)
- [加权算法](#)

- 动态反馈算法

静态负载均衡算法

- 轮询 (RoundRobin) : 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2到第7层的故障，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，根据这个比例，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2到第7层的故障，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，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，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。

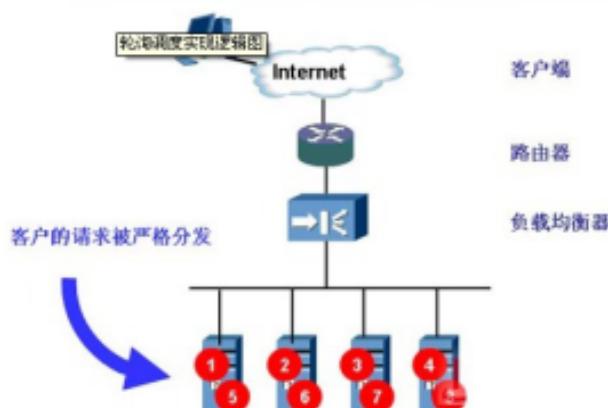
动态负载均衡算法

- 最少的连接方式 (LeastConnection) : 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2到第7层的故障，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2到第7层的故障，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2到第7层的故障，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) : 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。
- 动态性能分配(DynamicRatio-APM): 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(DynamicServerAct): 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TypeofField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，用户可自行设置

轮询算法

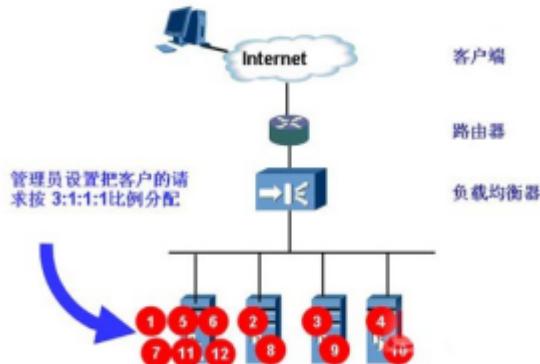
- 实现原理：
 - 每一次把来自用户的请求轮流分配给内部中的服务器，从1开始，直到N(内部服务器个数)，然后重新开始循环

负载均衡算法—轮询



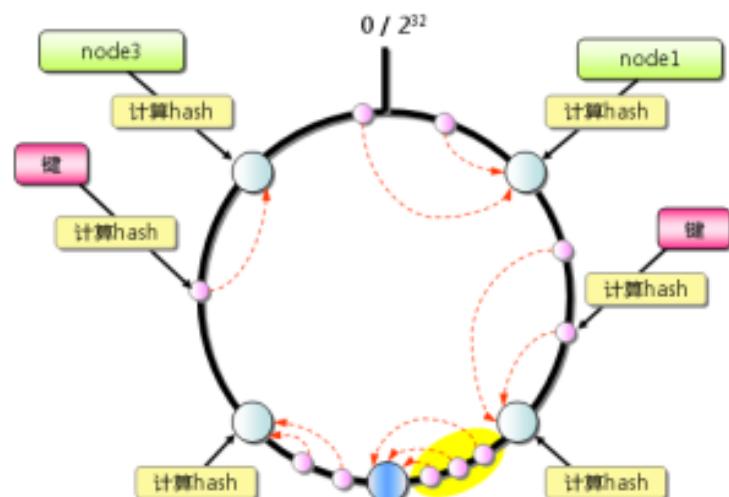
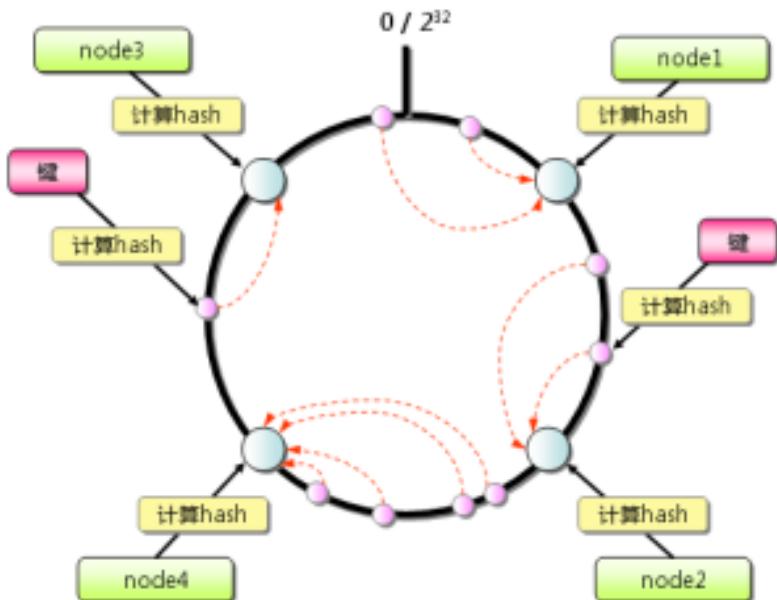
- 优点：
 - 简洁，无状态调度。
- 缺点：
 - 轮询调度算法假设所有服务器的处理性能都相同，不关心每台服务器的当前连接数和响应速度。当请求服务间隔时间变化比较大时，轮询调度算法容易导致服务器间的负载不平衡。
- 适用：
 - 服务器组中的所有服务器都有相同的软硬件配置并且平均服务请求相对均衡的情况

负载均衡算法—权重轮询调度



Hash散列算法

- MD5
- 一致性Hash算法
- 各种经典Hash算法
- 自定义Hash算法

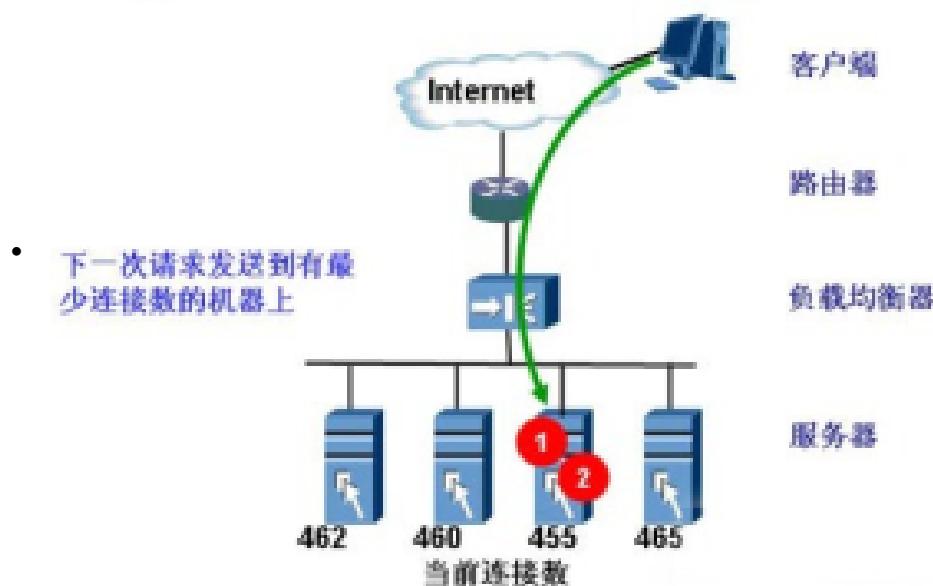




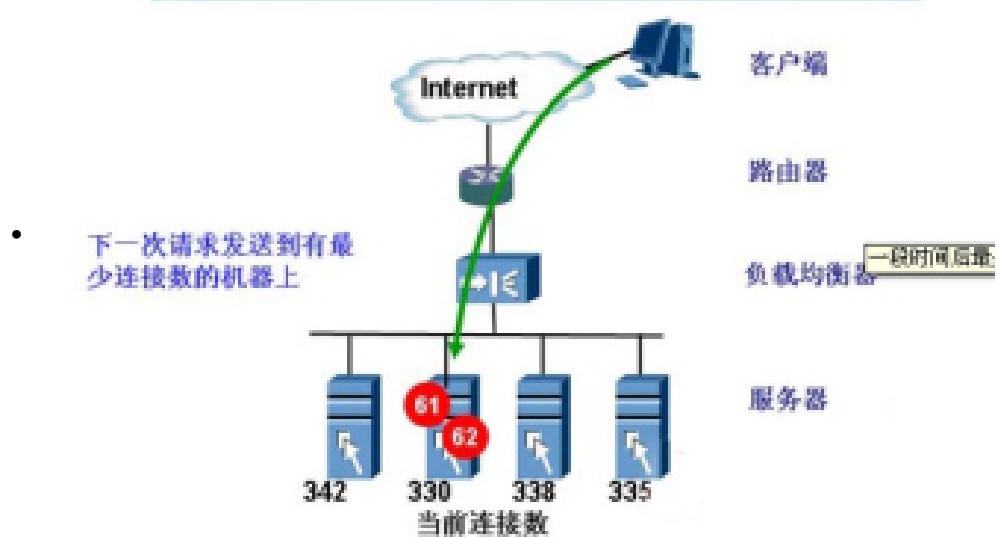
最少链接算法

- 实现原理:
 - 将请求分配至当前链接数最少的服务器

负载均衡算法—最少连接数



负载均衡算法—最少连接数—一定时间后



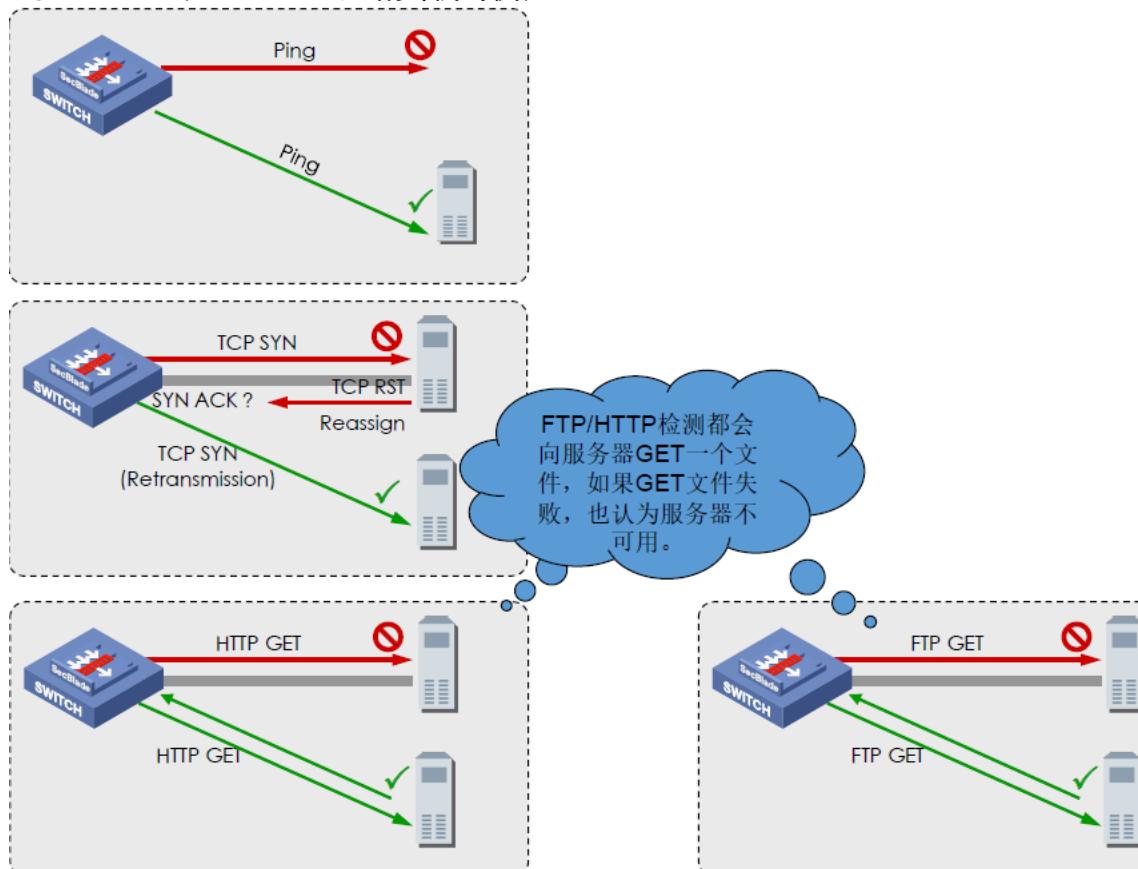
- 优点:
 - 实现起来比较简洁，在大多数情况下非常有效。
- 缺点:
 - 当各个服务器的处理能力不同时，该算法并不理想。
- 适用:
 - 需要长时处理的请求服务，如FTP等应用

最快链接算法

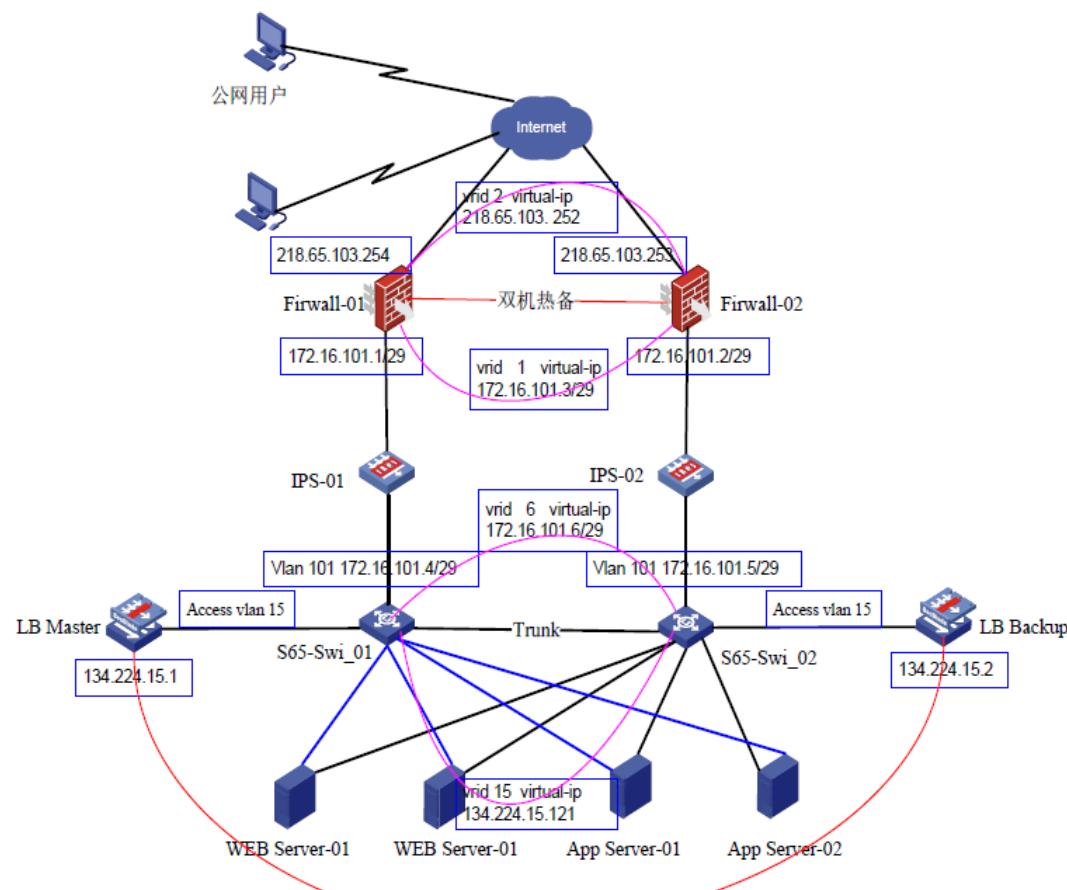
- 实现原理
 - 均衡器记录自身到每一个集群节点的网络响应时间，并将下一个到达的连接请求分配给响应时间最短的节点。
- 适用:
 - 基于拓扑结构重定向的高级均衡策略。

负载均衡产品中的关键指标：健康性检查算法

- 健康性检查算法的目的:
 - 通过某种探针机制，检查服务器群中真实服务器的健康情况，避免把客户端的请求分发给出现故障的服务器，以提高业务的HA能力。
- 目前常用的健康性检查算法:
 - Ping (ICMP)
 - TCP
 - HTTP
 - FTP
 - DNS (inbound 链路负载均衡)



负载均衡技术典型组网应用（双机热备）



双机热备
vrid 3 virtual-ip 134.224.15.3