随笔-161 文章-0 评论-13

LVS 介绍以及配置应用

1、负载均衡集群介绍

1.1、什么是负载均衡集群

负载均衡集群提供了一种廉价、有效、透明的方法，来扩展网络设备和服务器的负载、带宽、增加吞吐量、加强网络数据的处理能力、提高网络的灵活性和可用性

搭建负载均衡器的需求：

1）把单台计算机无法承受的大规模的并发访问或数据流量分担到多台节点设备上分别处理，减少用户等待时间，提升用户体验

2）单个重负载的运算分担到多台节点设备上做并行处理，每个节点设备处理结束后，将结果汇总，返回给用户，系统处理能力得到大幅度的提高。

3）7*24的服务保证，任意一个或多个有限后面节点设备宕机，要求不能影响业务。

在负载均衡器集群中，所有的计算节点都应该提供相同的服务。集群负载均衡获取所有对该服务的入站要求，然后将这些请求尽可能的平均分配在所有集群节点上。

1.2、常见的负载均衡器

a 根据工作在的协议层划分可划分为：

1.四层负载均衡（位于内核层）：根据请求报文中的目标地址和端口进行调度

2.七层负载均衡（位于应用层）：根据请求报文的内容进行调度，这种调度属于「代理」的方式

b 根据软硬件划分：

硬件负载均衡：

1.F5 的 BIG-IP

2.Citrix 的 NetScaler

3.这类硬件负载均衡器通常能同时提供四层和七层负载均衡，但同时也价格不菲

软件负载均衡：

1.TCP 层：LVS，HaProxy，Nginx

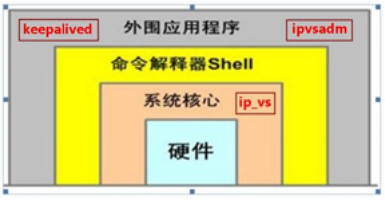
2.基于 HTTP 协议：Haproxy，Nginx，ATS（Apache Traffic Server），squid，varnish

3.基于 MySQL 协议：mysql-proxy

2、LVS（Linux Virtual Server）介绍

Internet的快速增长使多媒体网络服务器面对的访问数量快速增加，服务器需要具备提供大量并发访问服务的能力，因此对于大负载的服务器来讲， CPU、I/O处理能力很快会成为瓶颈。由于单台服务器的性能总是有限的，简单的提高硬件性能并不能真正解决这个问题。为此，必须采用多服务器和负载均衡技术才能满足大量并发访问的需要。Linux 虚拟服务器(Linux Virtual Servers,LVS) 使用负载均衡技术将多台服务器组成一个虚拟服务器。它为适应快速增长的网络访问需求提供了一个负载能力易于扩展，而价格低廉的解决方案。

LVS (Linux Virtual Server)其实是一种集群(Cluster)技术，采用IP负载均衡技术（LVS 的 IP 负载均衡技术是通过 IPVS 模块来实现的，linux内核2.6版本以上是默认安装IPVS的）和基于内容请求分发技术。调度器具有很好的吞吐率，将请求均衡地转移到不同的服务器上执行，且调度器自动屏蔽掉服务器的故障，从而将一组服务器构成一个高性能的、高可用的虚拟服务器。整个服务器集群的结构对客户是透明的，而且无需修改客户端和服务器端的程序。



LVS负载均衡调度技术是在LINUX内核中实现的，因此被称之为LINUX虚拟服务器。我们使用该软件配置LVS时候，不能直接配置内核中的IPVS，而需要使用IPVS的管理工具ipvsadm进行管理，当然我们也可以通过keepalived软件直接管理IPVS，并不是通过ipvsadm来管理ipvs。

LVS项目介绍

http://www.linuxvirtualserver.org/zh/lvs1.html

LVS集群的体系结构

http://www.linuxvirtualserver.org/zh/lvs2.html

LVS集群中的IP负载均衡技术 http://www.linuxvirtualserver.org/zh/lvs3.html
LVS集群的负载调度

http://www.linuxvirtualserver.org/zh/lvs4.html

LVS技术点小结：

1、真正实现调度的工具是IPVS，工作在LINUX内核层面。

2、LVS自带的IPVS管理工具是ipvsadm。

3、keepalived实现管理IPVS及负载均衡器的高可用。

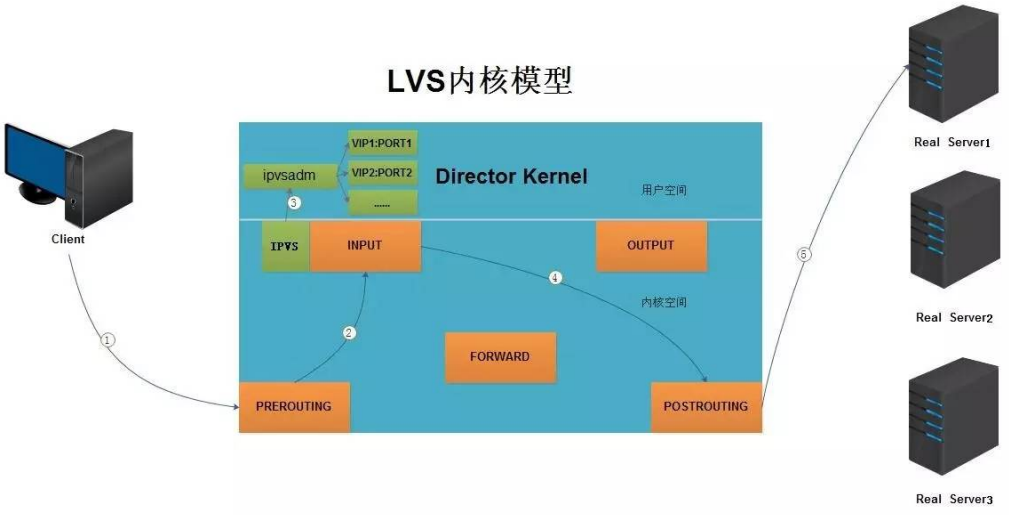
4、Red hat 工具Piranha WEB管理实现调度的工具IPVS（不常用）。

3、LVS集群的结构

LVS由前端的负载均衡器(Load Balancer，LB)和后端的真实服务器(Real Server，RS)群组成。RS间可通过局域网或广域网连接。LVS的这种结构对用户是透明的，用户只能看见一台作为LB的虚拟服务器(Virtual Server)，而看不到提供服务的RS群。当用户的请求发往虚拟服务器，LB根据设定的包转发策略和负载均衡调度算法将用户请求转发给RS。RS再将用户请求结果返回给用户。

- 负载调度器(load balancer/ Director)，它是整个集群对外面的前端机，负责将客户的请求发送到一组服务器上执行，而客户认为服务是来自一个IP地址(我们可称之为虚拟IP地址)上的。
- 服务器池(server pool/ Realserver)，是一组真正执行客户请求的服务器，执行的服务一般有WEB、MAIL、FTP和DNS等。
- 共享存储(shared storage)，它为服务器池提供一个共享的存储区，这样很容易使得服务器池拥有相同的内容，提供相同的服务。

4、LVS内核模型



1.当客户端的请求到达负载均衡器的内核空间时，首先会到达PREROUTING链。

2.当内核发现请求数据包的目的地址是本机时，将数据包送往INPUT链。

3.LVS由用户空间的ipvsadm和内核空间的IPVS组成，ipvsadm用来定义规则，IPVS利用ipvsadm定义的规则工作，IPVS工作在INPUT链上,当数据包到达INPUT链时，首先会被IPVS检查，如果数据包里面的目的地址及端口没有在规则里面，那么这条数据包将被放行至用户空间。

4.如果数据包里面的目的地址及端口在规则里面，那么这条数据报文将被修改目的地址为事先定义好的后端服务器，并送往POSTROUTING链。

5.最后经由POSTROUTING链发往后端服务器。

5、LVS的工作模式（包转发模型）

负载均衡技术有很多实现方案，有基于 DNS 域名轮流解析的方法、有基于客户端调度访问的方法、有基于应用层系统负载的调度方法，还有基于 IP 地址的调度方法，在这些负载调度算法中，执行效率最高的是IP 负载均衡技术。

LVS 的 IP 负载均衡技术是通过 IPVS 模块来实现的，IPVS 是 LVS集群系统的核心软件，它的主要作用是：安装在 Director Server 上，同时在 Director Server上虚拟出一个 IP 地址，用户必须通过这个虚拟的 IP 地址访问服务器。这个虚拟 IP 一般称为 LVS 的 VIP，即 Virtual IP。访问的请求首先经过 VIP 到达负载调度器，然后由负载调度器从 Real Server 列表中选取一个服务节点响应用户的请求。 在用户的请求到达负载调度器后，调度器如何将请求发送到提供服务的 Real Server 节点，而 Real Server节点如何返回数据给用户，是 IPVS 实现的重点技术。

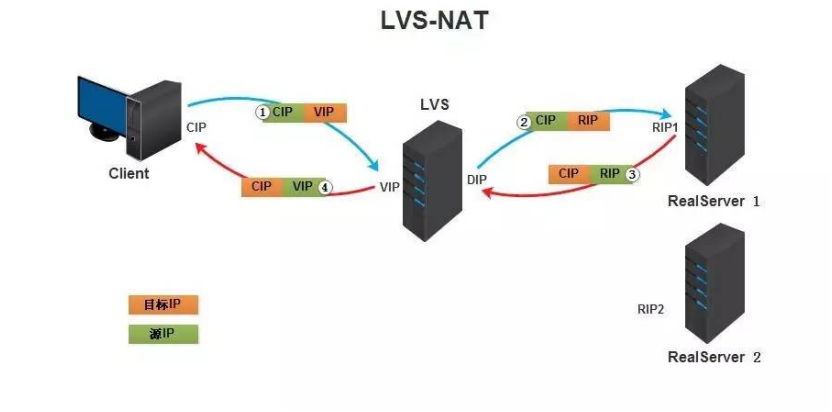
为了更好的理解LVS工作模式，我们可以约定一些名词

名称	缩写	说明
虚拟 IP 地址(Virtual Ip Address)	VIP	VIP 为 Director 用于向客户端计算机提供服务的 IP 地址。比如：www.etiantian.org 域名就要解析到 vip 上提供服务。
真实 IP 地址(Real Server Ip Address)	RIP	在集群下面节点上使用的 IP 地址，物理 IP 地址。
Director 的 IP 地址(Director Ip Address)	DIP	Director 用于连接内外网络的 IP 地址，物理网卡上的 IP 地址。是负载均衡器上的 IP。
客户端主机 IP 地址（Client Ip Address）	CIP	客户端用户计算机请求集群服务器的 IP 地址，该地址用作发送给集群的请求的源 IP 地址。

其中DR模式中重点。其他了解即可。

5.1、NAT模型

a 原理图



①.客户端将请求发往前端的负载均衡器，请求报文源地址是CIP(客户端IP),后面统称为CIP)，目标地址为VIP(负载均衡器前端地址，后面统称为VIP)。

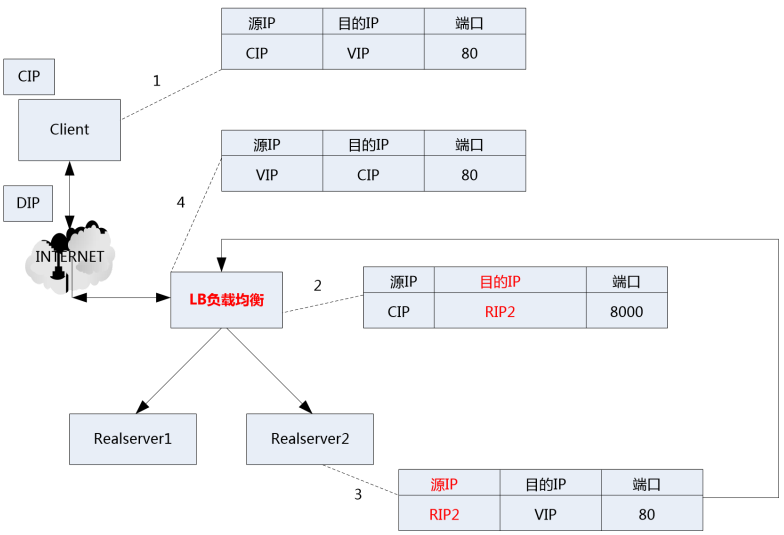
②.负载均衡器收到报文后，发现请求的是在规则里面存在的地址，那么它将客户端请求报文的目标地址改为了后端服务器的RIP地址并将报文根据算法发送出去。

③.报文送到Real Server后，由于报文的目标地址是自己，所以会响应该请求，并将响应报文返还给LVS。

④.然后lvs将此报文的源地址修改为本机并发送给客户端。

注意：在 NAT 模式中， Real Server 的网关必须指向 LVS ，否则报文无法送达客户端。

b IP包调度过程图



c 小结

1、NAT 技术将请求的报文和响应的报文都需要通过 LB 进行地址改写，因此网站访问量比较大的时候 LB 负载均衡调度器有比较大的瓶颈，一般要求最多之能 10-20 台节点

2、只需要在 LB 上配置一个公网 IP 地址就可以了。

3、每台内部的 realserver 服务器的网关地址必须是调度器 LB 的内网地址。

4、NAT 模式支持对 IP 地址和端口进行转换。即用户请求的端口和真实服务器的端口可以不一致。

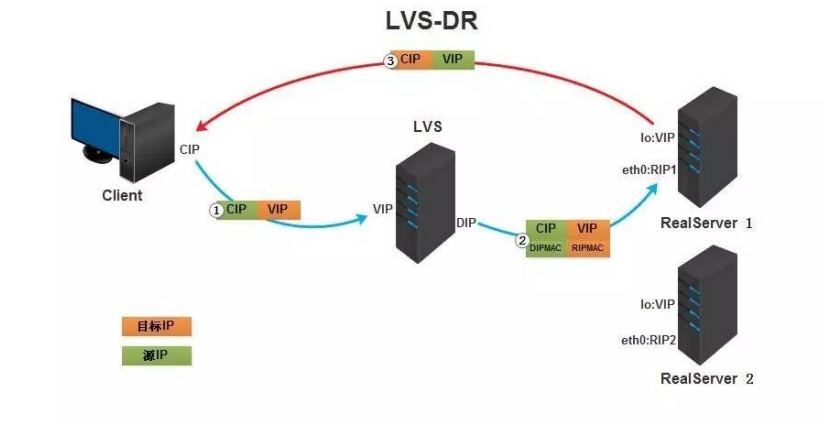
d 优缺点

优点：集群中的物理服务器可以使用任何支持TCP/IP操作系统，只有负载均衡器需要一个合法的IP地址。

缺点：扩展性有限。当服务器节点（普通PC服务器）增长过多时,负载均衡器将成为整个系统的瓶颈，因为所有的请求包和应答包的流向都经过负载均衡器。当服务器节点过多时，大量的数据包都交汇在负载均衡器那，速度就会变慢！

5.2、DR模型

a 原理图



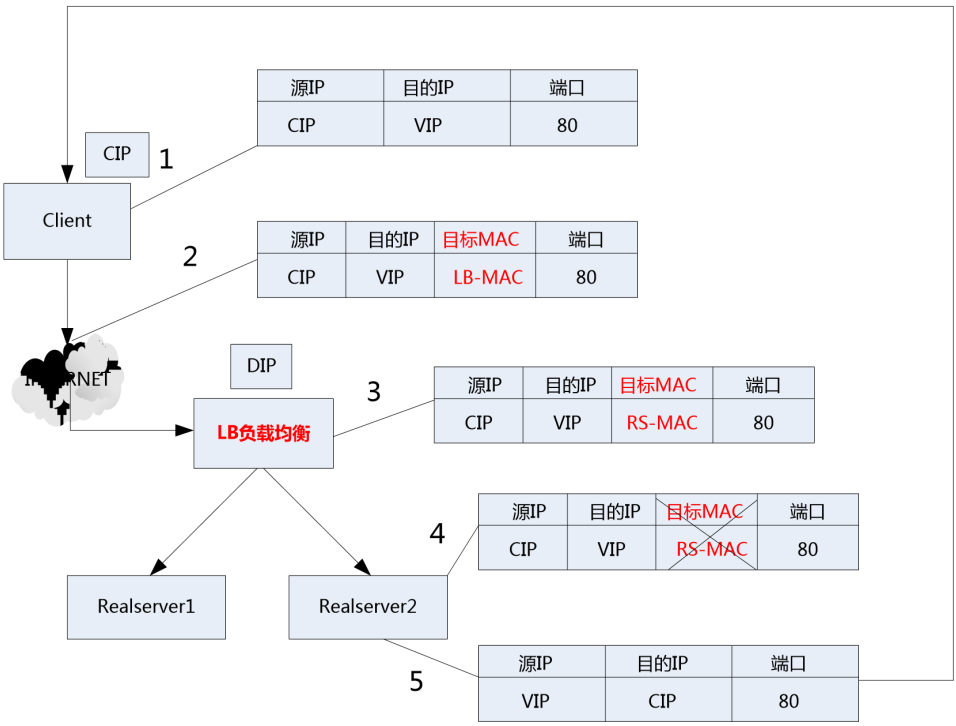
①.客户端将请求发往前端的负载均衡器，请求报文源地址是CIP，目标地址为VIP。

②.负载均衡器收到报文后，发现请求的是在规则里面存在的地址，那么它将客户端请求报文的源MAC地址改为自己DIP的MAC地址，目标MAC改为了RIP的MAC地址，并将此包发送给RS。

③.RS发现请求报文中的目的MAC是自己，就会将次报文接收下来，处理完请求报文后，将响应报文通过lo接口送给eth0网卡直接发送给客户端。

注意：需要设置 `lo` 接口的 `VIP` 不能响应本地网络内的 `arp` 请求。

b IP 包调度过程图



c 小结

1、通过在调度器 LB 上修改数据包的目的 MAC 地址实现转发。注意源地址仍然是 CIP，目的地址仍然是 VIP 地址。

2、请求的报文经过调度器，而 RS 响应处理后的报文无需经过调度器 LB，因此并发访问量大时使用效率很高（和 NAT 模式比）

3、因为 DR 模式是通过 MAC 地址改写机制实现转发，因此所有 RS 节点和调度器 LB 只能在一个局域网里面

4、RS 主机需要绑定 VIP 地址在 LO 接口（掩码32 位），并且需要配置 ARP 抑制。

5、RS 节点的默认网关不需要配置成 LB，而是直接配置为上级路由的网关，能让 RS 直接出网就可以。

6、由于 DR 模式的调度器仅做 MAC 地址的改写，所以调度器 LB 就不能改写目标端口，那么 RS 服务器就得使用和 VIP 相同的端口提供服务。

7、直接对外的业务比如WEB等，RS 的IP最好是使用公网IP。对外的服务，比如数据库等最好使用内网IP。

d 优缺点

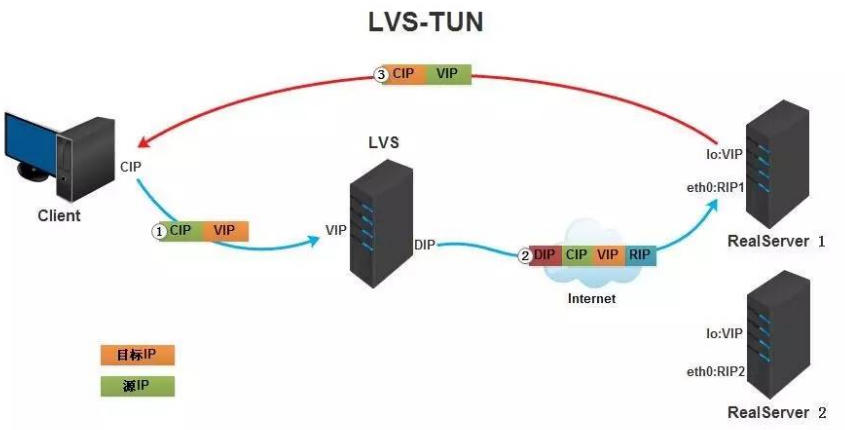
优点：和TUN（隧道模式）一样，负载均衡器也只是分发请求，应答包通过单独的路由方法返回给客户端。与VS-TUN相比，VS-DR这种实现方式不需要隧道结构，因此可以使用大多数操作系统做为物理服务器。

DR模式的效率很高，但是配置稍微复杂一点，因此对于访问量不是特别大的公司可以用haproxy/nginx取代。日1000-2000W PV或者并发请求1万一下都可以考虑用haproxy/nginx。

缺点：所有 RS 节点和调度器 LB 只能在一个局域网里面。

5.3、TUN模型

a 原理图



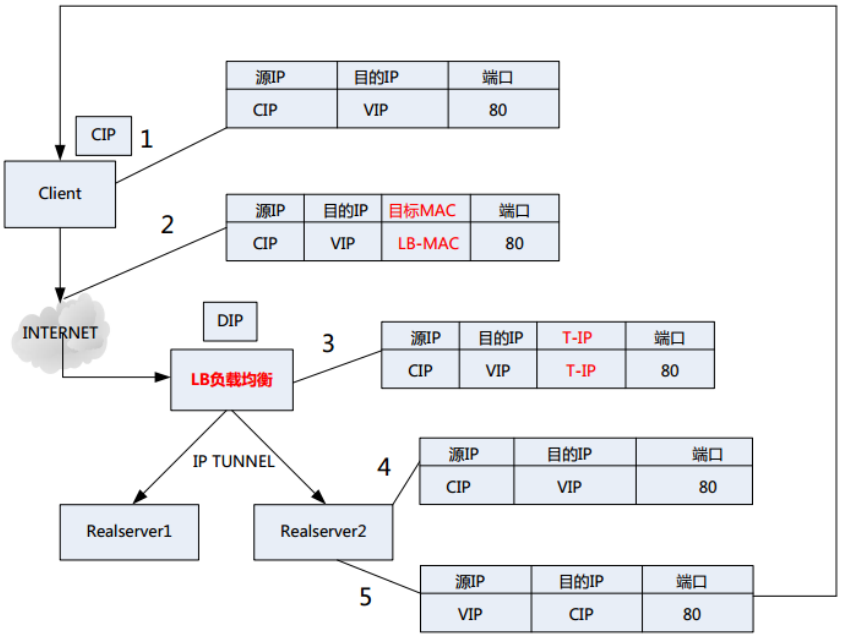
①.客户端将请求发往前端的负载均衡器，请求报文源地址是CIP，目标地址为VIP。

②.负载均衡器收到报文后，发现请求的是在规则里面存在的地址，那么它将在客户端请求报文的首部再封装一层IP报文,将源地址改为DIP，目标地址改为RIP,并将此包发送给RS。

③.RS收到请求报文后，会首先拆开第一层封装,然后发现里面还有一层IP首部的目标地址是自己lo接口上的VIP，所以会处理次请求报文，并将响应报文通过lo接口送给eth0网卡直接发送给客户端。

注意：需要设置`lo`接口的`VIP`不能在共网上出现。

b IP包调度过程图



c 小结

1.TUNNEL 模式必须在所有的 realserver 机器上面绑定 VIP 的 IP 地址

2.TUNNEL 模式的 vip ------>realserver 的包通信通过 TUNNEL 模式，不管是内网和外网都能通信，所以不需要 lvs vip 跟 realserver 在同一个网段内

3.TUNNEL 模式 realserver 会把 packet 直接发给 client 不会给 lvs 了

4.TUNNEL 模式走的隧道模式，所以运维起来比较难，所以一般不用。

d 优缺点

优点：负载均衡器只负责将请求包分发给后端节点服务器，而RS将应答包直接发给用户。所以，减少了负载均衡器的大量数据流动，负载均衡器不再是系统的瓶颈，就能处理很巨大的请求量，这种方式，一台负载均衡器能够为很多RS进行分发。而且跑在公网上就能进行不同地域的分发。

缺点：隧道模式的RS节点需要合法IP，这种方式需要所有的服务器支持”IP Tunneling”(IP Encapsulation)协议，服务器可能只局限在部分Linux系统上。

5.4、FULLNAT模型

（Full Network Address Translation）

a 原理图

无论是 DR 还是 NAT 模式，不可避免的都有一个问题：LVS 和 RS 必须在同一个 VLAN 下，否则 LVS 无法作为 RS 的网关。

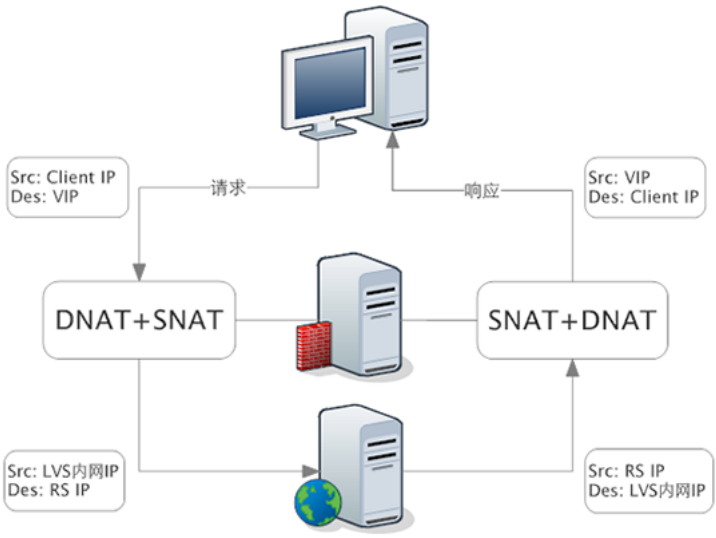
这引发的两个问题是：

1、同一个 VLAN 的限制导致运维不方便，跨 VLAN 的 RS 无法接入。

2、LVS 的水平扩展受到制约。当 RS 水平扩容时，总有一天其上的单点 LVS 会成为瓶颈。

Full-NAT 由此而生，解决的是 LVS 和 RS 跨 VLAN 的问题，而跨 VLAN 问题解决后，LVS 和 RS 不再存在 VLAN 上的从属关系，可以做到多个 LVS 对应多个 RS，解决水平扩容的问题。

Full-NAT 相比 NAT 的主要改进是，在 SNAT/DNAT 的基础上，加上另一种转换，转换过程如下：



在包从 LVS 转到 RS 的过程中，源地址从客户端 IP 被替换成了 LVS 的内网 IP。

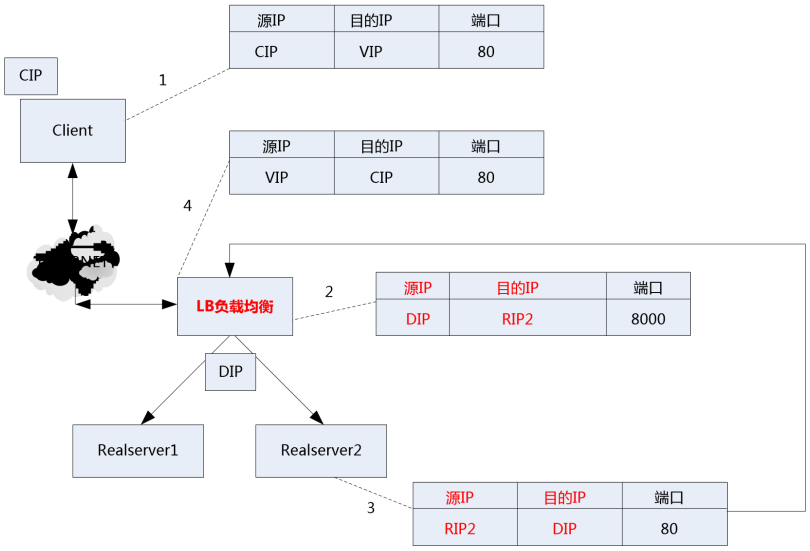
内网 IP 之间可以通过多个交换机跨 VLAN 通信。

当 RS 处理完接受到的包，返回时，会将这个包返回给 LVS 的内网 IP，这一步也不受限于 VLAN。

LVS 收到包后，在 NAT 模式修改源地址的基础上，再把 RS 发来的包中的目标地址从 LVS 内网 IP 改为客户端的 IP。

Full-NAT 主要的思想是把网关和其下机器的通信，改为了普通的网络通信，从而解决了跨 VLAN 的问题。采用这种方式，LVS 和 RS 的部署在 VLAN 上将不再有任何限制，大大提高了运维部署的便利性。

b IP包调度过程图



c 小结

1.FULL NAT 模式也不需要 LBIP 和 realserver ip 在同一个网段； full nat 跟 nat 相比的优点是：保证 RS 回包一定能够回到 LVS；因为源地址就是 LVS--> 不确定

2.full nat 因为要更新 sorce ip 所以性能正常比 nat 模式下降 10%

5.5、四种模式对比总结

性能：DR> TUN > NAT > FULL NAT

由于每个模式的功能不一样，所以具体的选择还是要根据公司业务的选择，实际环境来决定。

6、四层负载均衡LVS

LVS 是四层负载均衡，也就是说建立在 OSI 模型的第四层——传输层之上，传输层上有我们熟悉的 TCP/UDP，LVS 支持 TCP/UDP 的负载均衡。

LVS 的转发主要通过修改 IP 地址（NAT 模式，分为源地址修改 SNAT 和目标地址修改 DNAT）、修改目标 MAC（DR 模式）来实现。

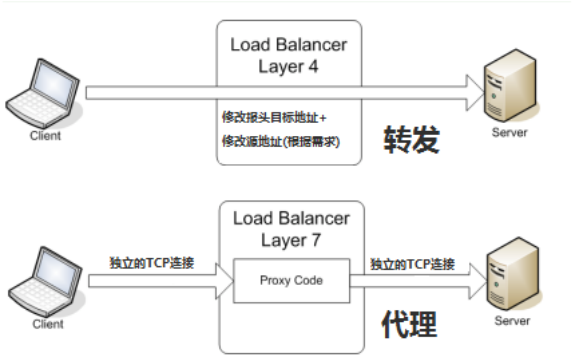
那么为什么 **LVS** 是在第四层做负载均衡？

首先 LVS 不像 HAProxy 等七层软负载面向的是 HTTP 包，所以七层负载可以做的 URL 解析等工作，LVS 无法完成。其次，某次用户访问是与服务端建立连接后交换数据包实现的，如果在第三层网络层做负载均衡，那么将失去「连接」的语义。软负载面向的对象应该是一个已经建立连接的用户，而不是一个孤零零的 IP 包。后面会看到，实际上 LVS 的机器代替真实的服务器与用户通过 TCP 三次握手建立了连接，所以 LVS 是需要关心「连接」级别的状态的

7、四层负载均衡和七层的区别

7.1. 四层负责均衡：

是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与请求客户端建立TCP连接，然后发送Client请求的数据。



由上图可知：在四层负载设备中，把client发送的报文目标地址(原来是负载均衡设备的IP地址)，根据均衡设备设置的选择web服务器的规则选择对应的web服务器IP地址，这样client就可以直接跟此服务器建立TCP连接并发送数据。

7.2. 七层负载均衡设备

也称内容交换，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，决定最终选择的服务器。

由上图可知，其实七层负载均衡服务器起了一个代理服务器的作用，我们知道建立一次TCP连接要三次握手；而client要访问webserver要先与七层负载设备进行三次握手后建立TCP连接，把要访问的报文信息发送给七层负载均衡；然后七层负载均衡再根据设置的均衡规则选择特定的webserver，然后通过三次握手与此台webserver建立TCP连接，然后webserver把需要的数据发送给七层负载均衡设备，负载均衡设备再把数据发送给client；所以，七层负载均衡设备起到了代理服务器的作用。

7.3. 七层的负责均衡设备的优点

(1) 使整个网络更“智能化”，能把对图片类的请求转发到图片服务器，对文字的请求转发到文字服务器

(2) 可以有效防止 SYN Flood攻击，是网站更安全

7.4. 七层负载均衡设备的缺点

因为七层负载均衡设备其实是一个代理服务器，则对此设备的要求也很高。

8、LVS的调度算法

Lvs的调度算法决定了如何在集群节点之间分布工作负荷。当director调度器收到来自客户端访问VIP的上的集群服务的入站请求时，director调度器必须决定哪个集群节点应该处理请求。

Director调度器用的调度方法基本分为两类：

固定调度算法：rr，wrr，dh，sh 动态调度算法：wlc，lc，lblc，lblcr , seq，nq。

对于这些算法我们只要知道常用的，其他了解即可，不需要深入其中的细节。

1、 静态算法（4种）：

只根据算法进行调度 而不考虑后端服务器的实际连接情况和负载情况

①.RR：轮叫调度（Round Robin）

调度器通过”轮叫”调度算法将外部请求按顺序轮流分配到集群中的真实服务器上，它均等地对待每一台服务器，而不管服务器上实际的连接数和系统负载。

②.WRR：加权轮叫（Weight RR）

调度器通过“加权轮叫”调度算法根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器处理更多的访问流量。调度器可以自动问询真实服务器的负载情况,并动态地调整其权值。

③.DH：目标地址散列调度（Destination Hash ）

根据请求的目标IP地址，作为散列键(HashKey)从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

④.SH：源地址 hash（Source Hash）

源地址散列”调度算法根据请求的源IP地址，作为散列键(HashKey)从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

2、 动态算法（6种）：

前端的调度器会根据后端真实服务器的实际连接情况来分配请求,Realserver 的负载状态通常由活动链接（active），非活动链接（inactive）和权重来计算。

①.LC：最少链接（Least Connections）

调度器通过”最少连接”调度算法动态地将网络请求调度到已建立的链接数最少的服务器上。如果集群系统的真实服务器具有相近的系统性能，采用”最小连接”调度算法可以较好地均衡负载。

②.WLC：加权最少连接(默认采用的就是这种)（Weighted Least Connections）

在集群系统中的服务器性能差异较大的情况下，调度器采用“加权最少链接”调度算法优化负载均衡性能，具有较高权值的服务器将承受较大比例的活动连接负载。调度器可以自动问询真实服务器的负载情况,并动态地调整其权值。

③.SED：最短延迟调度（Shortest Expected Delay ）

在WLC基础上改进，Overhead = （ACTIVE+1）*256/加权，不再考虑非活动状态，把当前处于活动状态的数目+1来实现，数目最小的，接受下次请求，+1的目的是为了考虑加权的时候，非活动连接过多缺陷：当权限过大的时候，会倒置空闲服务器一直处于无连接状态。

④.NQ永不排队/最少队列调度（Never Queue Scheduling NQ）

无需队列。如果有台 realserver的连接数＝0就直接分配过去，不需要再进行sed运算，保证不会有一个主机很空间。在SED基础上无论+几，第二次一定给下一个，保证不会有一个主机不会很空闲着，不考虑非活动连接，才用NQ，SED要考虑活动状态连接，对于DNS的UDP不需要考虑非活动连接，而httpd的处于保持状态的服务就需要考虑非活动连接给服务器的压力。

⑤.LBLC：基于局部性的最少链接（locality-Based Least Connections）

基于局部性的最少链接”调度算法是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。该算法根据请求的目标IP地址找出该目标IP地址最近使用的服务器，若该服务器是可用的且没有超载，将请求发送到该服务器;若服务器不存在，或者该服务器超载且有服务器处于一半的工作负载，则用“最少链接”的原则选出一个可用的服务器，将请求发送到该服务器。

⑥. LBLCR：带复制的基于局部性最少连接（Locality-Based Least Connections with Replication）

带复制的基于局部性最少链接”调度算法也是针对目标IP地址的负载均衡，目前主要用于Cache集群系统。它与LBLC算法的不同之处是它要维护从一个目标IP地址到一组服务器的映射，而LBLC算法维护从一个目标IP地址到一台服务器的映射。该算法根据请求的目标IP地址找出该目标IP地址对应的服务器组，按”最小连接”原则从服务器组中选出一台服务器，若服务器没有超载，将请求发送到该服务器；若服务器超载，则按“最小连接”原则从这个集群中选出一台服务器，将该服务器加入到服务器组中，将请求发送到该服务器。同时，当该服务器组有一段时间没有被修改，将最忙的服务器从服务器组中删除，以降低复制的程度。

3、算法总结

算法	说明
rr	轮询算法，它将请求依次分配给不同的rs节点，也就是RS节点中均摊分配。这种算法简单，但只适合于RS节点处理性能差不多的情况
wrr	加权轮训调度，它将依据不同RS的权值分配任务。权值较高的RS将优先获得任务，并且分配到的连接数将比权值低的RS更多。相同权值的RS得到相同数目的连接数。
Wlc	加权最小连接数调度，假设各台RS的全职依次为Wi，当前tcp连接数依次为Ti，依次去Ti/Wi为最小的RS作为下一个分配的RS 。
Dh	目的地址哈希调度（destination hashing）以目的地址为关键字查找一个静态hash表来获得需要的RS 。
SH	源地址哈希调度（source hashing）以源地址为关键字查找一个静态hash表来获得需要的RS。
Lc	最小连接数调度（least-connection）,IPVS表存储了所有活动的连接。LB会比较将连接请求发送到当前连接最少的RS。
Lblc	基于地址的最小连接数调度（locality-based least-connection）：将来自同一个目的地址的请求分配给同一台RS，此时这台服务器是尚未满负荷的。否则就将这个请求分配给连接数最小的RS，并以它作为下一次分配的首先考虑。
Lblcr	基于本地的带复制功能的最少连接，与 LBLC 不同的是 LVS 将请求 IP 映射至一个服务池中，使用 dh 算法调度请求至对应的服务池中，使用 lc 算法选择服务池中的节点，当服务池中的所有节点超载，使用 lc 算法从所有后端 Realserver 中选择一个添加至服务吃中。
Seq	最短期望延迟，它不对 inactive 状态的连接进行计算，根据 overhead = (active+1)*256/weight 计算服务器负载，选择 overhead 最小的服务器进行调度
Nq	当有空闲服务器时，直接调度至空闲服务器，当没有空闲服务器时，使用 SED 算法进行调度

4、生产环境算法的选择

1、一般的网络服务，如 www，mail，mysql 等常用的 LVS 调度算法为：

1、基本轮询调度 rr

2、加权最小连接调度 wlc

3、加权轮询调度 wrr

2、基于局部性的最小连接 lblc 和带复制的给予局部性最小连接 lblcr 主要适用于 web cache 和 DB cache ，一般不这么用。

3、源地址散列调度 SH 和目标地址散列调度 DH 可以结合使用在防火墙集群中，可以保证整个系统的出入口唯一。

实际适用中这些算法的适用范围很多，工作中最好参考内核中的连接调度算法的实现原理，然后根据具体的业务需求合理的选型。

rr，wrr，wlc是常用的。

9、Lvs安装与配置

1、主机准备

略

2、开始安装LVS

安装lvs

```
[root@lb01 application]# yum install ipvsadm -y
#YUM安装 LVS
[root@lb02 application]# ln -s /usr/src/kernels/`uname -r`/usr/src/linux
[root@lb01 application]# ipvsadm    #<=管理lvs的工具
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
[root@lb01 application]# lsmod |grep ip_vs   #查看IPVS加载到内核没有
ip_vs              126534  0
libcrc32c            1246  1 ip_vs
ipv6               335589  265 ip_vs
```

LVS安装提示：

1、CENTOS 5上安装LVS，使用1.24版本。不要用1.26。

2、CENTOS 6.4安装LVS，使用1.26版本 yum install libnl* popt* -y。

3、安装LVS后，要执行ipvsadm（modprobe ip_vs）把ip_vs模块加载到内核。

3、在LB上绑定VIP

命令行添加网卡：

```
ip addr add 10.0.0.3/24 dev eth0 label eth0:1
```

在LB上配置LVS

```
 [root@lb01 application]# ipvsadm -C     #<=清空整个表
[root@lb01 application]# ipvsadm --set 30 5 60
#<=修改LVS的超时参数
[root@lb01 application]# ipvsadm -A -t 10.0.0.4:80 -s wrr -p 1
#<=指定虚拟主机
[root@lb01 application]# ipvsadm -Ln
#<=列出LVS表
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.0.0.4:80 wrr persistent 1
[root@lb01 application]# ipvsadm -a -t 10.0.0.4:80 -r 10.0.0.7 -g
#<=添加后端真实服务节点
[root@lb01 application]# ipvsadm -a -t 10.0.0.4:80 -r 10.0.0.8 -g
[root@lb01 application]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
```

```
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.0.0.4:80 wrr persistent 1
  -> 10.0.0.7:80                  Route   10        0
  -> 10.0.0.8:80                  Route   10        0
```

附：

ipvsadm

【功能说明】：

ipvs直接调用的命令

【语法格式】：

ipsadm [选型]

【选项参数】：

-C 清空整个表

-A 通过选型，添加虚拟主机

-t 添加tcp虚拟主机的地址 格式为：host[:port]

-d 添加dcp虚拟主机的地址 格式为：host[:port]

-s 指定轮询算法[rr|wrr|lc|wlc|lblc|lblcr]

-L 列出整个表

-n 以数字的形式输出地址和端口

-a通过选型，添加真实主机【RS】

-r 真实主机的IP地址 host (and port)

-g 通过直接路由模式，及DR模式

-d 删除真实主机

ipvsadm -d -t 10.0.0.4:80 -r 10.0.0.7:80

#<=删除指定的虚拟主机

-p [timeot]会话保持的功能 [会话保持的作用：例如ipvsadm -A -t 10.0.0.4:80 -s rr -p 300 这个时间段内，所有的请求都会找一台机器，但是会导致负载不均]

4、手动在RS上绑定VIP

在web服务器上执行：

```
[root@web01 ~]# ip addr add 10.0.0.4/32  dev lo label lo:1   #<=临时生效
[root@web01 ~]# route add -host 10.0.0.4 dev lo
 #添加主机路由，为了回复数据包的时候经过l0网卡，是数据包的源IP是VIP，不是必须的操作。
[root@web01 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.4        0.0.0.0         255.255.255.255 UH    00        0 lo
```

5、抑制arp

在LVS DR模式当中，LB和后端的RS共用一个VIP（对外提供服务），为了保证客户端在请求VIP能得到LB的MAC地址，我们需要在后端的RS上配置ARP抑制。

```
echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
```

arp_ignore = 1即不回应不是目标IP是物理网络IP的ARP请求。确保了客户端请求VIP的时候只会得到LB的MAC地址。

arp_announce = 2即RS在回复客户端的时候，发送的ARP请求不以自己的VIP的为源IP，确保了不会更新上端路由的ARP缓存，从而保证以后客户端请求VIP的时候读取路由器ARP缓存会得到LB的MAC地址。

说明：

arp_ignore-INTEGE：

定义目标地址为本地IP的ARP询问不同的应答模式

0-（默认值）：回应任何网络接口上对本地IP地址的arp查询请求

1-只回答目标IP地址是来访问网络接口本地地址的ARP查询请求

2- 只回答目标IP地址是来访网络接口本地地址的ARP查询请求，且来自网络接口的子网的内。

3- 不回应网络界面的ARP请求，而且只对设置的唯一和连接地址做出回应。

4-7- 保留未使用

8-不回应所有（本地地址）的ARP查询

arp_announce-INTEGER：

"0"代表是用ip包的源地址来设置ARP请求的源地址。

"1"代表不使用ip包的源地址来设置ARP请求的源地址，如果ip包的源地址是和该端口的IP地址相同的子网，那么用ip包的源地址，来设置ARP请求的源地址，否则使用"2"的设置。

"2"代表不使用ip包的源地址来设置ARP请求的源地址，而由系统来选择最好的接口来发送

当内网的机器要发送一个到外部的ip包，那么它就会请求路由器的Mac地址，发送一个arp请求，这个arp请求里面包括了自己的ip地址和Mac地址，而linux默认是使用ip的源ip地址作为arp里面的源ip地址，而不是使用发送设备上面的 ，这样在lvs这样的架构下，所有发送包都是同一个VIP地址，那么arp请求就会包括VIP地址和设备 Mac，而路由器收到这个arp请求就会更新自己的arp缓存，这样就会造成ip欺骗了，VIP被抢夺，所以就会有问题。

6、编写脚本实现

以上的手动操作我们可以通过脚本去实现。

ipvs server启动脚本

```
[root@LB01 scripts]# cat ipvs.server.sh
#!/bin/sh
. /etc/init.d/functions

VIP=10.0.0.29
PORT=80
RIP=(
10.0.0.8
10.0.0.7
)
start(){
   ifconfig eth2:0 $VIP/24 up
   route add -host $VIP dev eth2
   ipvsadm -C
   ipvsadm --set 30 5 60   30
   ipvsadm -A -t $VIP:$PORT -s rr -p 20
  for ((i=0;i<${#RIP[*]};i++))
  do
    ipvsadm -a -t $VIP:$PORT -r ${RIP[$i]} -g -w 1
  done
}

stop(){
    ipvsadm -C
```

```
    ifconfig eth2:0 down
    route del -host $VIP dev eth0
}
case "$1" in
        start)
                start
                echo "IPVS is started"
                ;;
        stop)
                stop
                echo "IPVS is stopped"
                ;;
        restart)
                stop
                sleep 2
                start
                echo "IPVS is restarted"
                ;;
        *)
                echo "USAGE:$0 {start|stop|restart}"
esac
```

7、关于在虚拟机上测试LVS负载均衡

由于我们在电脑上装的虚拟机上测试。

所以虚拟出来的路由以及LINUX客户端

可能不支持echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce这个参数，所以就会有ARP缓存的问题。

那么我们在浏览器访问的时候可以在每次访问后，清空arp缓存（arp –d ）。就会出现1比1的负载均衡。

如果是LINUX上我们可以

```
[root@lb01 ~]# arp -d 10.0.0.3 ; curl 10.0.0.3
yyyy
[root@lb01 ~]# arp -d 10.0.0.3 ; curl 10.0.0.3
yangliheng
```

10、LVS各个模型的实现

10.1、LVS NAT模型的实现

1. 集群环境：一台Director, 两台后端Real Server RS1，RS2

```
    Director:两网卡

eth1:192.168.0.33/24
eth2:172.16.12.1/24
eth1:1:192.168.0.200/24作为VIP地址

    RS1:
eth1:172.16.12.11

    RS2:
```

```
eth1:172.16.12.12


   Director的eth0为桥接，eth1和RS1,RS2的eth0使用仅主机模式。在RS1和RS2上使用仅主机模式前先分别安装好LAMP的环境，并设置好测试页面。
```

2，配置网络

```
配置RS1：


[root@localhost html]# ifconfig eth1 172.16.12.11/24
[root@localhost html]# route add default gw 172.16.12.1


配置RS2：


[root@localhost html]# ifconfig eth1 172.16.12.12/24
[root@localhost html]# route add default gw 172.16.12.1


在Director上的配置：
[root@localhost ~]# yum install ipvsadm -y    #安装客户端工具
[root@localhost ~]# ifconfig eth2 172.16.12.1/24
[root@localhost ~]# ifconfig eth1:1 192.168.0.200  #添加VIP
[root@localhost ~]# iptables -F
[root@localhost ~]# iptables -t filter -F
[root@localhost ~]# iptables -L -n
Chain INPUT (policy ACCEPT)
targetprot opt sourcedestination

Chain FORWARD (policy ACCEPT)
targetprot opt sourcedestination

Chain OUTPUT (policy ACCEPT)
targetprot opt sourcedestination
[root@localhost ~]# service iptables  save
iptables:Saving firewall rules to /etc/sysconfig/iptables:[  OK  ]
```

3. 修改内核参数，开启转发功能

```
[root@localhost ~]# vim /etc/sysctl.conf   #打开转发功能
net.ipv4.ip_forward =1


[root@localhost ~]# sysctl -p    #使其生效
```

4. 验证RS1和RS2上面创建的测试页

```
[root@localhost ~]# curl 172.16.12.11
web1 172.16.12.11
[root@localhost ~]# curl 172.16.12.12
web2 172.16.12.12
```

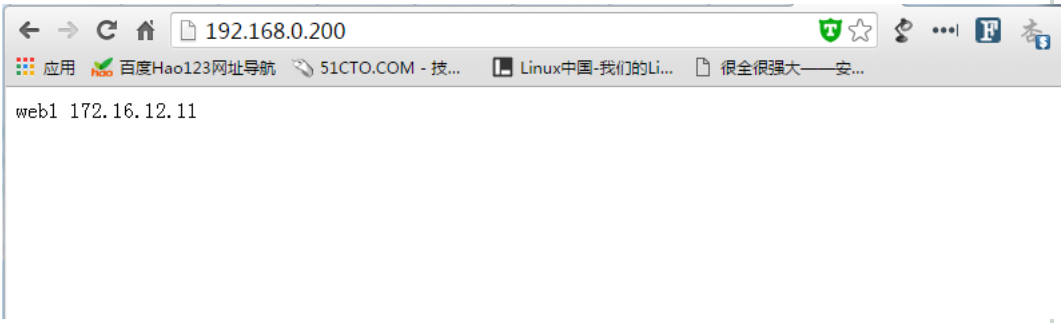
5. 在Director上添加集群服务

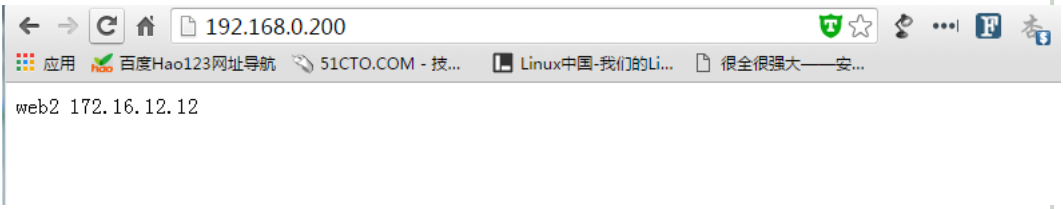
```
[root@localhost ~]# ipvsadm -A -t 192.168.0.200:80 -s rr    #创建一个集群服务
```

```
[root@localhost ~]# ipvsadm -L -n
IP VirtualServer version 1.2.1(size=4096)
ProtLocalAddress:PortSchedulerFlags
  ->RemoteAddress:Port           ForwardWeightActiveConnInActConn
TCP  192.168.0.35:80 rr


[root@localhost ~]# ipvsadm -a -t 192.168.0.200:80 -r 172.16.12.11 -m#-m类型，默认wlc
[root@localhost ~]# ipvsadm -a -t 192.168.0.200:80 -r 172.16.12.12 -m
[root@localhost ~]# ipvsadm -L -n
IP VirtualServer version 1.2.1(size=4096)
ProtLocalAddress:PortSchedulerFlags
  ->RemoteAddress:Port           ForwardWeightActiveConnInActConn
TCP  192.168.0.200:80 rr
  ->172.16.12.11:80Masq    1      08
  ->172.16.12.12:80Masq    1      08
```

好了，LVS的NAT模型就这样配置完成了，然后到浏览器中进行访问：





10.2、LVS DR模型的实现

1. 集群环境：一台Director, 两台Real Server RS1，RS2，此次使用较简单的配置方法，三个主机都使用桥接模式

```
   Director:两网卡


eth1:192.168.0.33/24
eth1:0:192.168.0.200/24   作为VIP地址，可以进行浮动


   RS1：
eth1:192.168.0.23


   RS2：
eth1:192.168.0.24
```

2，配置网络

```
配置Dircector
```

```
[root@localhost ~]# ifconfig eth1:0 192.168.0.200  up      #配置VIP
[root@localhost ~]# route add -host 192.168.0.200 dev eth1:0

配置RS1，修改内核参数，关闭lo的arp通告和lo的arp响应，并配置隐藏地址，并且保证其发出报文经过eth1之前，还要先经过lo:1 VIP,使得源地址成为VIP

[root@localhost ~]# echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
[root@localhost ~]# echo 1 > /proc/sys/net/ipv4/conf/eth1/arp_ignore
[root@localhost ~]# echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
[root@localhost ~]# echo 2 > /proc/sys/net/ipv4/conf/eth1/arp_announce

[root@localhost ~]# ifconfig lo:0 192.168.0.200 netmask 255.255.255.255 broadcast 192.168.0.200 up
[root@localhost ~]# route add -host 192.168.0.200 dev lo:0

配置RS2：

[root@localhost ~]# echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
[root@localhost ~]# echo 1 > /proc/sys/net/ipv4/conf/eth1/arp_ignore
[root@localhost ~]# echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
[root@localhost ~]# echo 2 > /proc/sys/net/ipv4/conf/eth1/arp_announce
[root@localhost ~]# ifconfig lo:0 192.168.0.200 netmask 255.255.255.255 broadcast 192.168.0.200 up   #只广播自己
[root@localhost ~]# route add -host 192.168.0.200 dev lo:0     #目标是192.168.0.200时经过设备lo:0
```

3，在Director上添加集群服务

```
[root@localhost ~]# ipvsadm -A -t 192.168.0.200:80 -s rr
[root@localhost ~]# ipvsadm -a -t 192.168.0.200:80 -r 192.168.0.23 -g -w 1#使用权重1
[root@localhost ~]# ipvsadm -a -t 192.168.0.200:80 -r 192.168.0.24 -g -w 3#使用权重3
```

4，在浏览器中进行测试，但是此配置中为RS1和RS2设置了权重，所以转发到后的RS时，响应的次数为3:1。

11、LVS Session持久机制

1、session绑定：始终将同一个请求者的连接定向至同一个RS（第一次请求时仍由调度方法选择）；没有容错能力，有损均衡效果；

2、session复制：在RS之间同步session，因此，每个RS持集群中所有的session；对于大规模集群环境不适用；

3、session服务器：利用单独部署的服务器来统一管理session;

12、LVS持久连接

12.1、持久连接的实现机制

无论使用什么算法，LVS持久连接都能实现在一点时间内，将来自于同一个客户端请求派发至此前选定的realserver,DH调度算法本身依赖于TCP会话超时时间等其他计时器，而持久连接依赖于持久连接模板， 每个新的连接请求，无论客户端的连接状态无论是否断开，只要客户端曾经访问过，LVS就会在持久连接模板中记录信息， 持久连接模板(内存缓冲区):记录每一个客户端及分配的realserver的映射关系。

经常用于SSL,建立一个SSL连接，需要交换SSL密钥，当启用持久性连接时，只需要做一次验证即可。

12.2、PPC 持久端口连接 基于端口

来自于同一个客户端对同一个服务的请求，始终定向至此前选定的realserver。

ipvsadm -A -t 192.**168.1.200**:23 -s rr-p 3600

ipvsadm -a -t 192.**168.1.200**:23 -r 192.**168.1.10** -g -w 2

ipvsadm -a -t 192.**168.1.200**:23 -r 192.**168.1.20** -g -w 1

12.3、PCC 持久客户端连接 基于所有端口

来自于同一个客户端对所有服务的请求，始终定向至此前选定的realserver

ipvsadm -A -t 192.**168.1.200**:0 -s rr -p 600

ipvsadm -a -t 192.**168.1.200**:0 -r 192.**168.1.10** -g -w 2

ipvsadm -a -t 192.**168.1.200**:0 -r 192.**168.1.20** -g -w 1

12.4、PNMPP 持久防火墙标记连接

来自于同一客户端对指定服务的请求，始终定向至此算定的realserver基于指定的端口，它可以 将两个毫不相干的端口定义为一个集群服务， 例如：合并http telnet为同一个集群服务。

##PNMPP是通过路由前给数据包打标记来实现的

iptables -t mangle -A PREROUTING -d 192.168.1.200 -eth0 -p tcp --dport 80 -j MARK --**set**-mark 3

iptables -t mangle -A PREROUTING -d 192.168.1.200 -eth0 -p tcp --dport 23 -j MARK --**set**-mark 3

ipvsadm -A -f 3 -s rr -p 600

ipvsadm -a -f 3 -r 192.168.1.10 -g -w 2

ipvsadm -a -f 3 -r 192.168.1.20 -g -w 2

注意:以上三种持久连接均使用了"-p" 选项 ，它保证了持久性，默认为300秒

警告:Director没有定义用户请求的集群服务，将试图自己响应客户端请求。

13、常见的lvs高可用方案

lvs两大弱点：1、手动配置 2、健康检查

不能检测后端服务器的健康状况，总是发送连接到后端

13.1、通过redhat提供的工具piranha来配置LVS

The Piranha Solution

http://www.linuxvirtualserver.org/docs/ha/piranha.html

不推荐。

13.2、keepalived+LVS方案

这个方案符合简单、易用、高效的运维原则，也是接下来重点讲解的负载均衡及高可用解决方案。

The Keepalived Solution

http://www.linuxvirtualserver.org/docs/ha/keepalived.html

keepalived的作用

1、管理LVS负载均衡软件

2、使配置LVS更加自动化

3) 实现LVS主被服务器高可用

14、LVS负载不均衡的解决思路

生产案例：

生产环境中ipvsadm –L –n 发现两台RS的负载不均衡，一台有很多请求，一台米有。并且没有请求的那台RS经测试服务正常，lo:VIP也有。但是就是没有请求。

```
TCP  172.168.1.50:3307 wrr persistent 10
 -> 172.168.1.51:3307           Route   10      0
 -> 172.168.1.52:3307           Route   18      12758
```

问题原因：

persistent 10的原因，persistent会话保持，当clientA访问网站的时候，LVS把请求分发给了52，那么以后client A再点击的其他操作其他请求，也会发送给52台机器。

解决办法：

到keepalived中注释掉persistent 10 然后/etc/init.d/keepalived reload然后可以看到以后负载均衡两边都请求均衡了。

那么导致负载不均衡的原因有：

1、LVS滋生的会话保持参数设置（-p 300 ,persistent 300）。优化：大公司尽量用cookies替代session。

2、LVS调度算法设置，例如:rr，wrr，wlc，lc算法。

3、后端RS节点的会话保持参数。

4、访问量较少的时候，不均衡比较明显。

5、用户发送的请求时间长短，和请求资源多少大小。

实现会话保持的方案：
http://oldboy.blog.51cto.com/2561410/1331316
http://oldboy.blog.51cto.com/2561410/1323468

15、LVS排错

排错大体的思路就是，要熟悉LVS的工作原理过程，然后根据原理过程来排重，例如：

1、调度器上LVS调度规则及IP的正确性。

2、RS节点上VIP绑定和arp抑制的检查。

生产处理思路：

对RS绑定的VIP做实时监控，出问题报警或者自动处理后报警。

把RS绑定的VIP做成配置文件，例如：vi /etc/sysconfig/network-scripts/lo:0

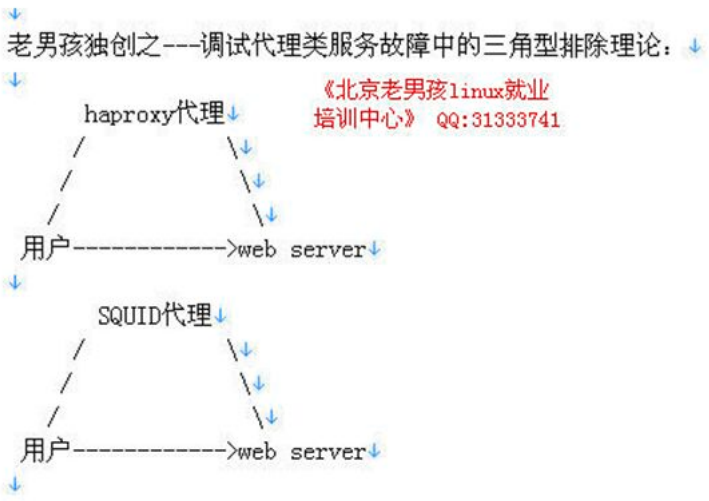
ARP抑制的配置思路：

当RS端有多个VIP绑定，即使停止了一个VIP也不要修改ARP抑制。

3、RS节点上自身提供服务的检查。

4、辅助排查工具有tepdump，ping等。

5、负载均衡以及反向代理集群的三角形排查理论：



16、突破LVS瓶颈

LVS的并发是十万级别的，那么如何突破这个瓶颈呢

1、突破**LVS**瓶颈，**LVS Cluster**部署（**OSPF + LVS**）

http://my.oschina.net/lxcong/blog/143904?fromerr=wW5KTv1c

2、智能**DNS**

在机房的前端增加DNS轮询。

实现的软件有dnspod,

17、LVS代码平滑上线发布思路

在生产环境中我们发布代码一般要平滑上线以及灰度发布。发布代码的时候要遵循一定的过程：

开发人员本地测试—办公室内部测试—SVN（配置管理员）--IDC机房测试环境—正式服务器

当我们要在正式服务器上线代码的时候一定要遵循平滑发布的思路即不能让客户感觉到网站在更新：

关于代码的上线更新一般分为2个种类：

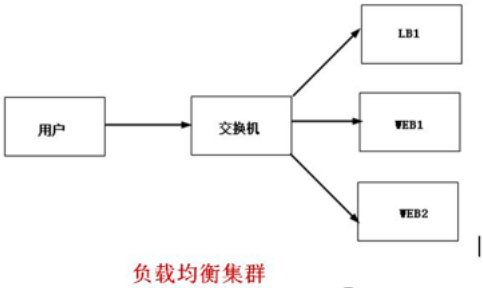
1、PHP这类不需要重启服务的代码。我们上线的时候不要直接从本地上传服务器去覆盖。而是应该先拷贝到服务器上的临时目录后，再去覆盖代码。当然我们也可以给代码做软链接，这样我们上线的时候，只需要删除、再建立软链接就可以了。

2、java这类需要重启服务的代码，我们上线的时候可以利用LVS负载来进行平滑上线。假设我们的real server节点有N个，那么我们需要在LVS上踢掉一半的节点，在这些被踢掉的节点上更新代码，同时在测试LVS集群上测试代码的可靠性。如果没问题，我们再将这些节点加入LVS，同时踢掉另一半的real server，在另一半的real server上也上线代码。测试没问题后，再加入到正式的LVS集群当中。这就是利用LVS的负载均衡做的平滑上线代码的思路。

18、LVS总结

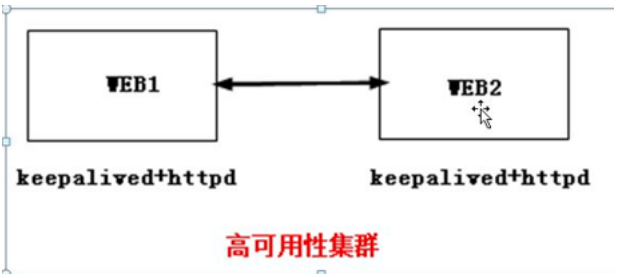
1、arp协议的介绍及实现原理。（这部分是为了理解LVS DR模式的原理。）

2、LVS集群的原理，安装以及手工开发脚本配置实现多种模式的负载均衡。



3、keepalived软件的介绍及高可用应用

1) httpd（nginx或者apache）服务的高可用



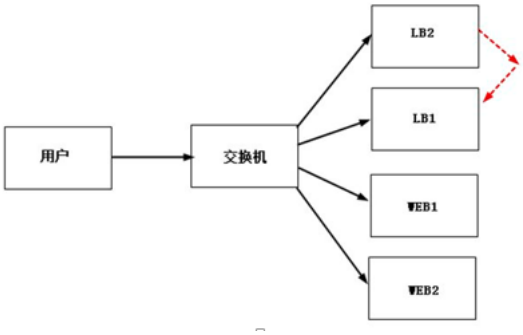
2) MYSQL服务的高可用

3) 反向代理的高可用

keepalived+nginx代理，也可以实现nginx的高可用性集群，而nginx本身还有负载均衡集群的功能。

keepalived+haproxy代理，也可以实现haproxy的高可用性集群，而haproxy本身还有负载均衡集群的功能。

4、lvs+keepalived实现负载均衡及高可用性。



来自为知笔记(Wiz)

分类: 大型集群 服务

杨小愚
关注 - 4
粉丝 - 42
+加关注

« 上一篇 : linux 运维知识体系
» 下一篇 : LVS高可用集群

posted @ 2016-07-21 15:48 杨小愚 阅读(5390) 评论(0) 编辑 收藏