# 一. 集群的概念

服务器集群简称集群是一种服务器系统，它通过一组松散集成的服务器软件和/或硬件连接起来高度紧密地协作完成计算工作。在某种意义上，他们可以被看作是一台服务器。
集群系统中的单个服务器通常称为节点，通常通过局域网连接，但也有其它的可能连接方式。集群服务器通常用来改进单个服务器的计算速度和/或可靠性。一般情况下集群
服务器比单个服务器，比如工作站或超级服务器性能价格比要高得多。集群就是一组独立的服务器，通过网络连接组合成一个组合来共同完一个任务。

说的直白点，集群就是一组相互独立的服务器，通过高速的网络组成一个服务器系统，每个集群节点都是运行其自己进程的一个独立服务器。对网络用户来讲，网站后
端就是一个单一的系统，协同起来向用户提供系统资源，系统服务。

# 二. 为什么要使用集群

### 1）集群的特点

- 高性能performance

一些需要很强的运算处理能力比如天气预报，核试验等。这就不是几台服务器能够搞定的。这需要上千台一起来完成这个工作的。

- 价格有效性

通常一套系统集群架构，只需要几台或数十台服务器主机即可，与动则上百万的专用超级服务器具有更高的性价比。

- 可伸缩性

当服务器负载压力增长的时候，系统能够扩展来满足需求，且不降低服务质量。

- 高可用性

尽管部分硬件和软件发生故障，整个系统的服务必须是7*24小时运行的。

### 2）集群的优势

- 透明性

如果一部分服务器宕机了业务不受影响，一般耦合度没有那么高，依赖关系没有那么高。比如NFS服务器宕机了其他就挂载不了了，这样依赖性太强。

- 高性能

访问量增加，能够轻松扩展。

- 可管理性

整个系统可能在物理上很大，但很容易管理。

- 可编程性

在集群系统上，容易开发应用程序，门户网站会要求这个。

### 3）集群分类及不同分类的特点

计算机集群架构按照功能和结构一般分成以下几类：

- 负载均衡集群（Loadbalancingclusters）简称LBC
- 高可用性集群（High-availabilityclusters）简称HAC
- 高性能计算集群（High-perfomanceclusters）简称HPC
- 网格计算（Gridcomputing）

就集群分类而言，网络上面一般认为是有三个，负载均衡和高可用集群式我们互联网行业常用的集群架构。

### 1）负载均衡集群

负载均衡集群为企业提供了更为实用，性价比更高的系统架构解决方案。负载均衡集群把很多客户集中访问的请求负载压力可能尽可能平均的分摊到计算机集群中处理。
客户请求负载通常包括应用程度处理负载和网络流量负载。这样的系统非常适合向使用同一组应用程序为大量用户提供服

务。每个节点都可以承担一定的访问请求负载压力，
并且可以实现访问请求在各节点之间动态分配，以实现负载均衡。

负载均衡运行时，一般通过一个或多个前端负载均衡器将客户访问请求分发到后端一组服务器上，从而达到整个系统的高性能和高可用性。这样集群有时也被称为服务器群。
一般高可用性集群和负载均衡集群会使用类似的技术，或同时具有高可用性与负载均衡的特点。

负载均衡集群的作用：
a）分担访问流量（负载均衡）
b）保持业务的连续性（高可用）

### 2) 高可用性集群

一般是指当集群中的任意一个节点失效的情况下，节点上的所有任务自动转移到其他正常的节点上，并且此过程不影响整个集群的运行，不影响业务的提供。类似是集群中运行着两个或两个以上的一样的节点，当某个主节点出现故障的时候，那么其他作为从 节点的节点就会接替主节点上面的任务。从节点可以接管主节点的资源（IP地址，架构身份等），此时用户不会发现提供服务的对象从主节点转移到从节点。
高可用性集群的作用：当一台机器宕机另一台进行接管。比较常用的高可用集群开源软件有：keepalive，heardbeat。

### 3) 高性能计算集群

高性能计算集群采用将计算任务分配到集群的不同计算节点儿提高计算能力，因而主要应用在科学计算领域。比较流行的HPC采用Linux操作系统和其它一些免费软件来完成并行运算。这一集群配置通常被称为Beowulf集群。这类集群通常运行特定的程序以发挥HPCcluster的并行能力。这类程序一般应用特定的运行库，比如专为科学计算设计的MPI库。HPC集群特别适合于在计算中各计算节点之间发生大量数据通讯的计算作业，比如一个节点的中间结果或影响到其它节点计算结果的情况。

## 三. 负载均衡集群介绍

负载均衡集群是 Load Balance 集群，是一种将网络上的访问流量分布于各个节点，以降低服务器压力，更好的向客户端提供服务的一种方式。
负载均衡集群的作用：提供一种廉价、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，加强网络数据处理能力、提高网络的灵活性和可用性。简单来说,也就是：
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，减少用户等待响应的时间，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提高。
3) 7*24小时的服务保证，任意一个或多个设备节点设备宕机，不能影响到业务。在负载均衡集群中，所有计算机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如站请求。

常用的负载均衡分为：
1) 开源软件负载均衡： Nginx, LVS, Haproxy （Nginx和Haproxy通常做七层负载均衡, LVS做四层负载均衡. 但是Nginx也可以通过stream模块做四层负载均衡, Haproxy也可以做四层负载均衡 ）；
2) 商业的硬件负载均衡：设备F5、Netscale ；

简单理解一下软件负载均衡：
1) 所谓分层的负载均衡，都是以网络的模型来说的。四层就是基于IP和端口的负载均衡，七层就是基于URL等应用信息的负载均衡。所以简单的说四层负载均衡就是通过IP和端口接收请求再分发至真实的服务器，七层是通过URL或主机名接收请求，然后分发至真实的服务器。
2) .而七层的实现也是在四层的基础上是实现的，没有四层就不可能有七层。在第七层上可以做许多事情，比如可以根据七层的浏览器类别区分是手机还是PC，将WEB服务器分为2组，手机登陆专门的移动端网站。
3) 对客户端来说，客户端好像是访问的同一台主机。其实为了有更好的用户体验，从智能DNS入手，根据客户端IP来源将域名解析到距离客户端最近的一台服务器或者访问最快速的一台服务器，但这些内容客户端都是感觉不到的，客户端感觉到的只能是访问网站很快。

## 四. LVS负载均衡集群说明

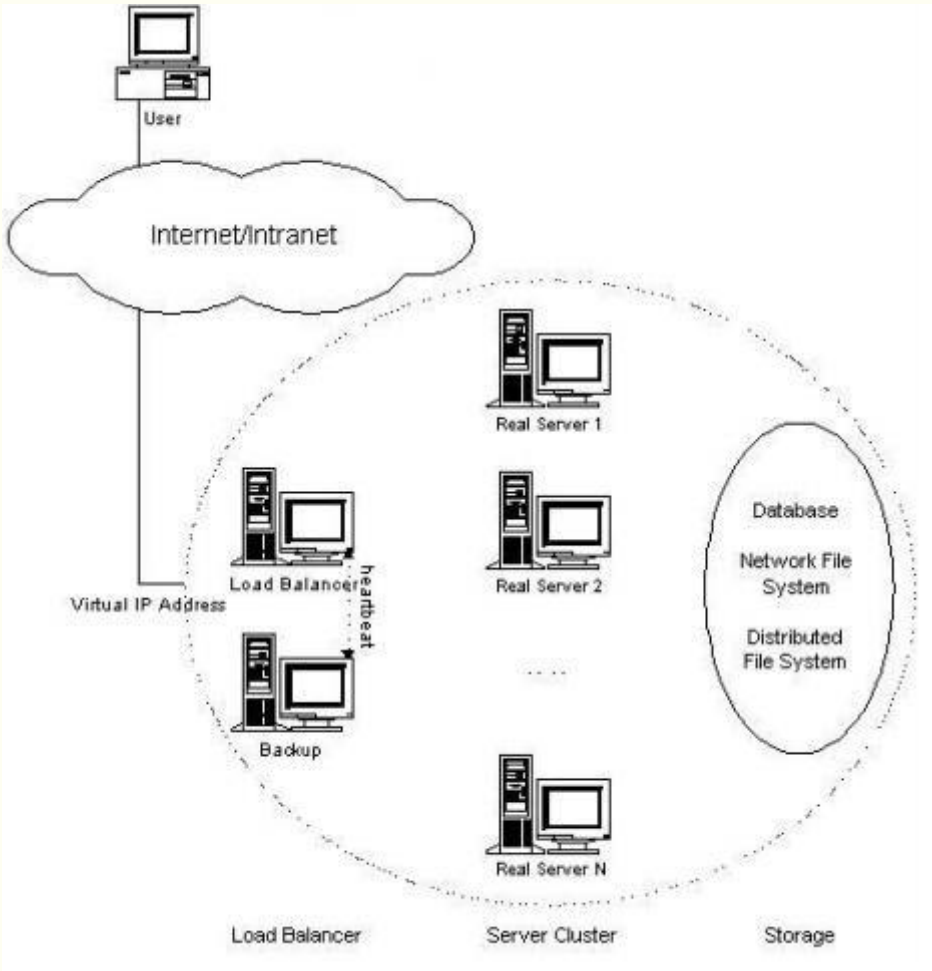
### 1) LVS是什么?

LVS是linux virtual server的简写linux虚拟服务器，是一个虚拟的服务器集群系统，可以在unix/linux平台下实现负载均衡集群功能。该项目在1998年5月由章文嵩博士组织成立。LVS是一种集群(Cluster)技术，采用IP负载均衡技术和基

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。

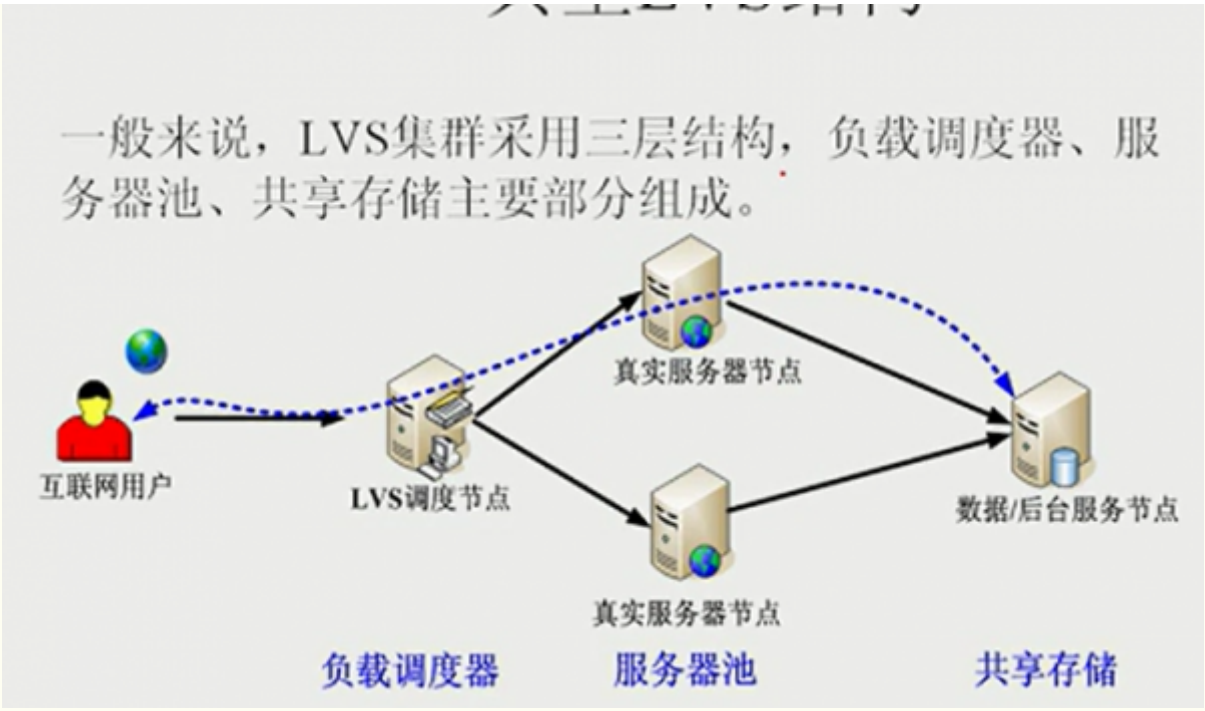
LVS集群采用IP负载均衡技术和基于内容请求分发技术。调度器具有很好的吞吐率，将请求均衡地转移到不同的服务器上执行，且调度器自动屏蔽掉服 务器的故障，从而将一组服务器构成一个高性能的、高可用的虚拟服务器。整个服务器集群的结构对客户是透明的，而且无需修改客户端和服务器端的程序。

LVS在设计时需要考虑系统的透明性、可伸缩性、高可用性和易管理性。一般来说，LVS集群采用三层结构，其体系结构如图所示：




负载均衡的原理很简单，就是当客户端发起请求时，请求直接发给Director Server（调度器），这时会根据设定的调度算法，将请求按照算法的规定智能的分发到真正的后台服务器。以达到将压力均摊。但是我们知道，http的连接时无状态的，假设这样一个场景，我登录某宝买东西，当我看上某款商品时，我将它加入购物车，但是我刷新了一下页面，这时由于负载均衡的原因，调度器又选了新的一台服务器为我提供服务，我刚才的购物车内容全都不见了，这样就会有十分差的用户体验。所以就还需要一个存储共享，这样就保证了用户请求的数据是一样的。所以LVS负载均衡分为三层架构(也就是LVS负载均衡主要组成部分)：

**第一层**：负载调度器（load balancer/ Director），它是整个集群的总代理，它在有两个网卡，一个网卡面对访问网站的客户端，一个网卡面对整个集群的内部。负责将客户端的请求发送到一组服务器上执行，而客户也认为服务是来自这台主的。举个生动的例子，集群是个公司，负载调度器就是在外接揽生意，将接揽到的生意分发给后台的真正干活的真正的主机们。当然需要将活按照一定的算法分发下去，让大家都公平的干活。
**第二层**：服务器池（server pool/ Realserver），是一组真正执行客户请求的服务器，可以当做WEB服务器。就是上面例子中的小员工。
**第三层**：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。一个公司得有一个后台账目吧，这才能协调。不然客户把钱付给了A，而换B接待客户，因为没有相同的账目。B说客户没付钱，那这样就不是客户体验度的问题了。




## 2) LVS负载均衡集群特点

### 2.1) IP负载均衡与负载调度算法

IP负载均衡技术

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。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。当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，**IPVS实现负载均衡机制有三种**，分别是NAT、TUN和DR(下面会详细介绍)；

负载调度算法

负载调度器是根据各个服务器的负载情况，动态地选择一台Real Server响应用户请求，那么动态选择是如何实现呢，其实也就是我们这里要说的负载调度算法，根据不同的网络服务需求和服务器配置，IPVS实现了如下**八种负载调度算法**：rr、wrr、Wlc、Dh、SH、Lc、Lblc(下面会详细介绍)；

### 2.2) 高可用性

LVS是一个基于内核级别的应用软件，因此具有很高的处理性能，后端服务器可运行任何支持TCP/IP的操作系统，包括Linux，各种Unix（如FreeBSD、Sun Solaris、HP Unix等），Mac/OS和Windows NT/2000等。负载调度器能够支持绝大多数的TCP和UDP协议.

### 2.3) 性能

LVS服务器集群系统具有良好的伸缩性，可支持几百万个并发连接。用LVS构架的负载均衡集群系统具有优秀的处理能力，每个服务节点的故障不会影响整个系统的正常使用，同时又实现负载的合理均衡，使应用具有超高负荷的服务能力，可支持上百万个并发连接请求。如配置百兆网卡，采用VS/TUN或VS/DR调度技术，整个集群系统的吞吐量可高达1Gbits/s；如配置千兆网卡，则系统的最大吞吐量可接近10Gbits/s。

### 2.4) 高可靠性

LVS负载均衡集群软件已经在企业、学校等行业得到了很好的普及应用，国内外很多大型的、关键性的web站点也都采用了LVS集群软件，所以它的可靠性在实践中得到了很好的证实。有很多以LVS做的负载均衡系统，运行很长时间，从未做过重新启动。这些都说明了LVS的高稳定性和高可靠性。

### 2.5) 适用环境

LVS对前端Director Server目前仅支持Linux和FreeBSD系统，但是支持大多数的TCP和UDP协议，支持TCP协议的应用有：HTTP，HTTPS ，FTP，SMTP，，POP3，IMAP4，PROXY，LDAP，SSMTP等等。支持UDP协议的应用有：DNS，NTP，ICP，视频、音频流播放协议等。LVS对Real Server的操作系统没有任何限制，Real Server可运行在任何

支持TCP/IP的操作系统上，包括Linux，各种Unix（如FreeBSD、Sun Solaris、HP Unix等），Mac/OS和Windows等。

### 2.6) 开源软件（软件许可证）

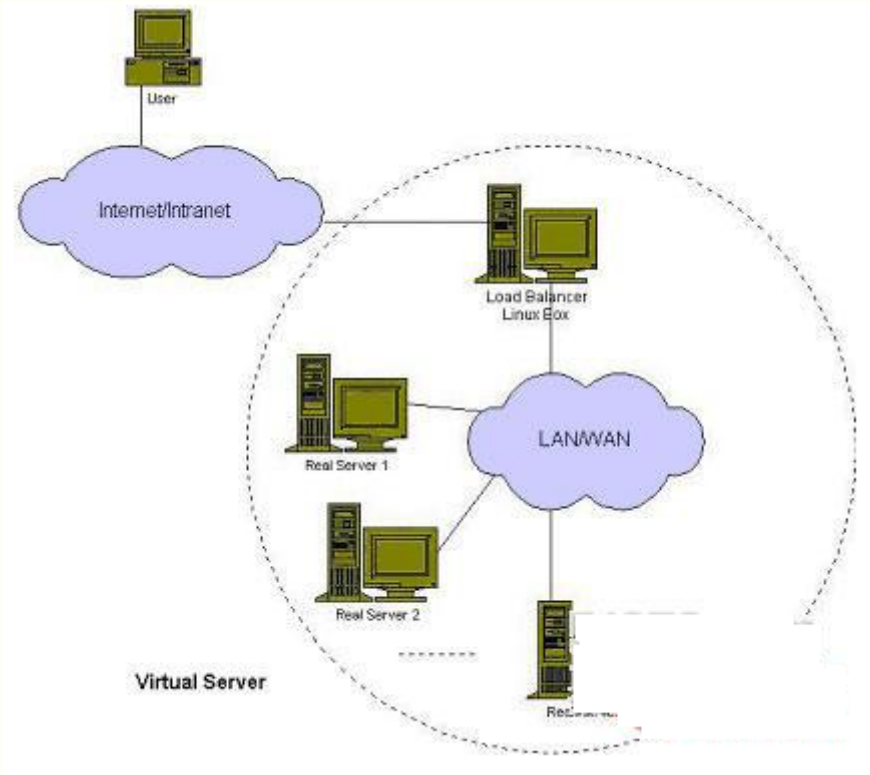
LVS集群软件是按GPL（GNU Public License）许可证发行的自由软件，因此，使用者可以得到软件的源代码，并且可以根据自己的需要进行各种修改，但是修改必须是以GPL方式发行。

## 3) LVS体系结构

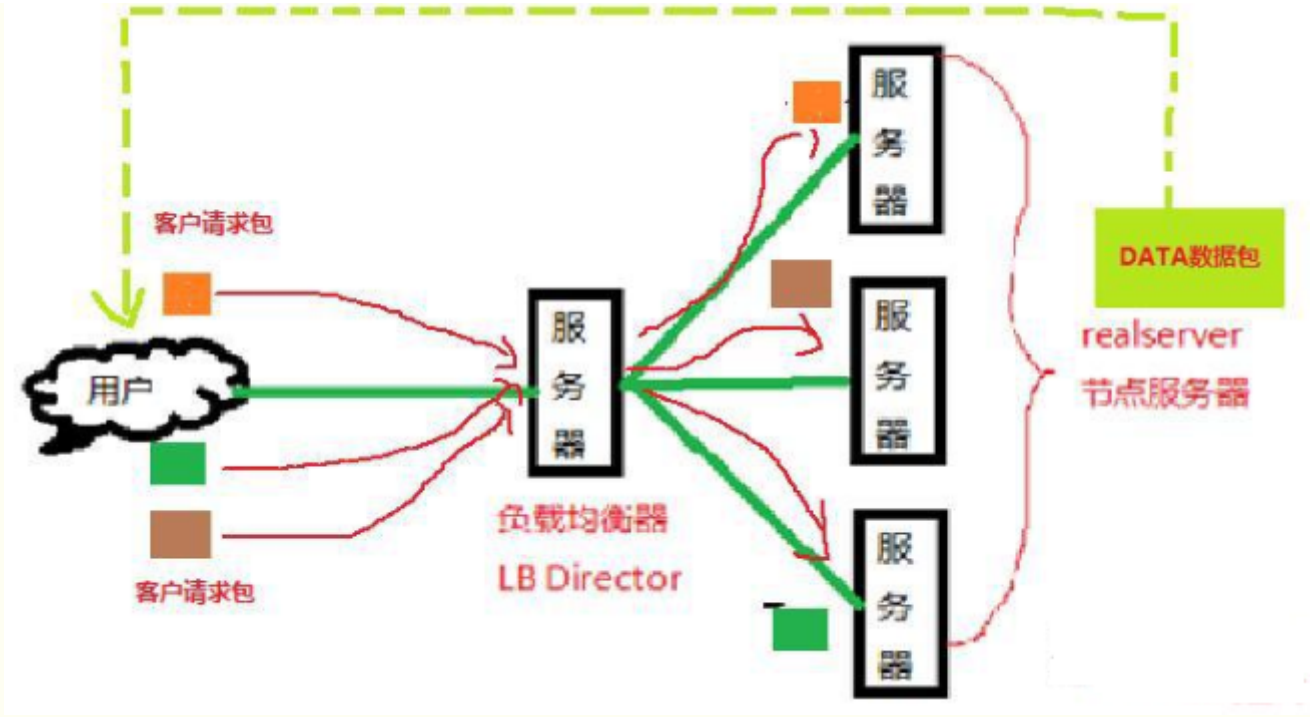
LVS集群负载均衡器接受服务的所有入展客户端的请求，然后根据调度算法决定哪个集群节点来处理回复客户端的请求。LVS虚拟服务器的体系如下图所示，一组服务器通过高速的局域网或者地理分布的广域网相互连接，在这组服务器之前有一个负载调度器（load balance）。负载调度器负责将客户的请求调度到真实服务器上。这样这组服务器集群的结构对用户来说就是透明的。客户访问集群系统就如只是访问一台高性能，高可用的服务器一样。客户程序不受服务器集群的影响，不做任何修改。

就比如说：我们去饭店吃饭点菜，客户只要跟服务员点菜就行。并不需要知道具体他们是怎么分配工作的，所以他们内部对于我们来说是透明的。此时这个服务员就会按照一定的规则把他手上的活，分配到其他人员上去。这个服务员就是负载均衡器（LB）而后面这些真正做事的就是服务器集群。

**LVS结构图如下:**




**LVS基本工作过程**




客户请发送向负载均衡服务器发送请求。负载均衡器接受客户的请求，然后先是根据LVS的调度算法（8种）来决定要将这个请求发送给哪个节点服务器。然后依据自己的工作模式（3种）来看应该如何把这些客户的请求如何发送给节点服务器，节点服务器又应该如何来把响应数据包发回给客户端。

**LVS组成**

lvs分为两个部分，分别是内核模块和lvs的管理工具。目前来说，centos6及其以上的内核版本已经包括了ipvs的相关模块了。

```
[root@localhost ~]# modprobe -l|grep -i ipvs
kernel/net/netfilter/ipvs/ip_vs.ko
kernel/net/netfilter/ipvs/ip_vs_rr.ko
kernel/net/netfilter/ipvs/ip_vs_wrr.ko
kernel/net/netfilter/ipvs/ip_vs_lc.ko
kernel/net/netfilter/ipvs/ip_vs_wlc.ko
kernel/net/netfilter/ipvs/ip_vs_lblc.ko
kernel/net/netfilter/ipvs/ip_vs_lblcr.ko
kernel/net/netfilter/ipvs/ip_vs_dh.ko
kernel/net/netfilter/ipvs/ip_vs_sh.ko
kernel/net/netfilter/ipvs/ip_vs_sed.ko
kernel/net/netfilter/ipvs/ip_vs_nq.ko
kernel/net/netfilter/ipvs/ip_vs_ftp.ko
kernel/net/netfilter/ipvs/ip_vs_pe_sip.ko
```

从上面可知，**内核支持的ipvs模块**，上图中的rr，wrr，lc，wlc，lblc等等都是lvs中调度器的调度算法，根据不同的调度算法可以更好的分配服务，实现负载均衡。而ipvs(ip virtual server)：一段代码工作在内核空间，实现调度。

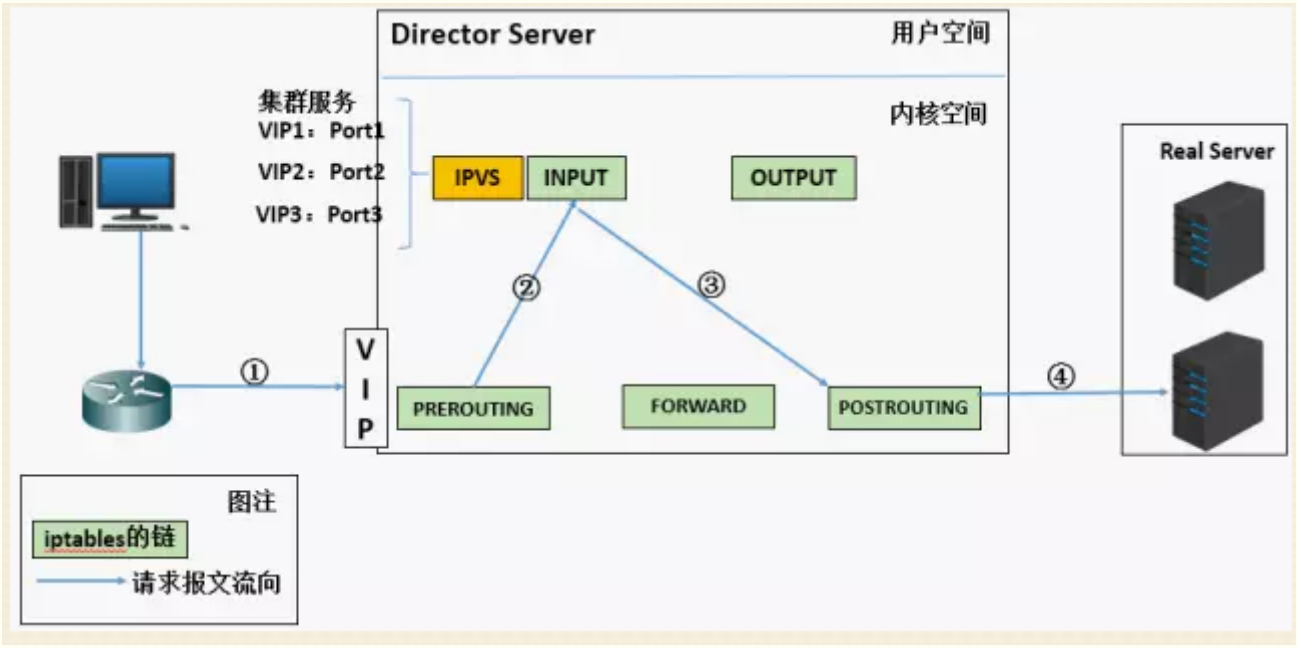
```
========================== N/S Matched: ipvsadm ==========================
ipvsadm.x86_64 : Utility to administer the Linux Virtual Server

  Name and summary matches only, use "search all" for everything.
```

上图是ipvsadm (即LVS客户端管理工具)，主要负责为ipvs内核框架编写规则，定义谁是集群服务，而谁是后端真实的服务器(Real Server)。

## 4) LVS的实现原理

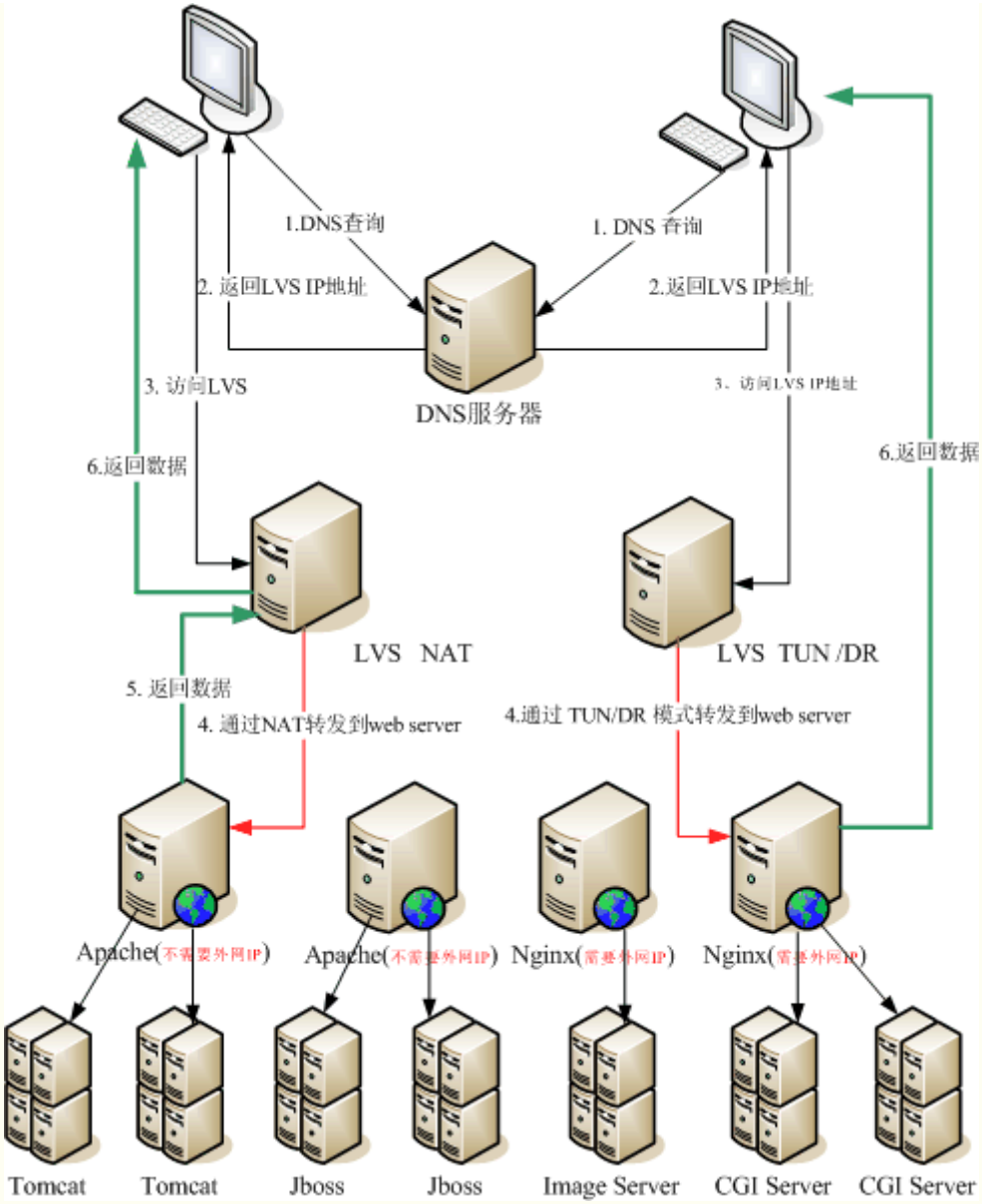
lvs的原理其实就是利用了Iptables的功能。了解防火墙的都知道四表五链。防火墙不仅仅有放火的功能还有转发，地址伪装，限流等等功能。




1) 首先，客户端向调度器（Director Server）发起一个请求，调度器将这个请求发送至内核
2) PREROUTING链首先会接收到用户请求，判断目标IP确定是本机IP，将数据包发往INPUT链。
3) 当请求达到INPUT链上，调度器判断报文中的目标端口来确定这个访问是不是要访问集群服务（因为还有可能只是ssh想单纯的远程登录主机这个主机），如果是访问的集群服务，那么就会强制修改这个包的目标IP
4) POSTROUTING链接收数据包后发现目标IP地址刚好是自己的后端服务器，那么此时通过选路，将数据包最终发送给后端的服务器

## 5) LVS的工作原理

LVS 的工作模式分为4中分别是 NAT，DR，TUN，FULL-NAT。其中做个比较，由于工作原理的关系的，NAT的配置最为简单，但是NAT对调度器的压力太大了，导致其效率最低，DR和TUN的工作原理差不多，但是DR中，所有主机必须处于同一个物理环境中，而在TUN中，所有主机可以分布在不同的位置，服务器一个在纽约，一个在深圳。最多应用的是FULL-NAT。




其中的专业术语

**DS**：Director Server。指的是前端负载均衡器。

**RS**：Real Server。后端真实的工作服务器。

**VIP**：向外部直接面向用户请求，作为用户请求的目标的IP地址。

**DIP**：Director Server IP，主要用于和内部主机通讯的IP地址。

**RIP**：Real Server IP，后端服务器的IP地址。

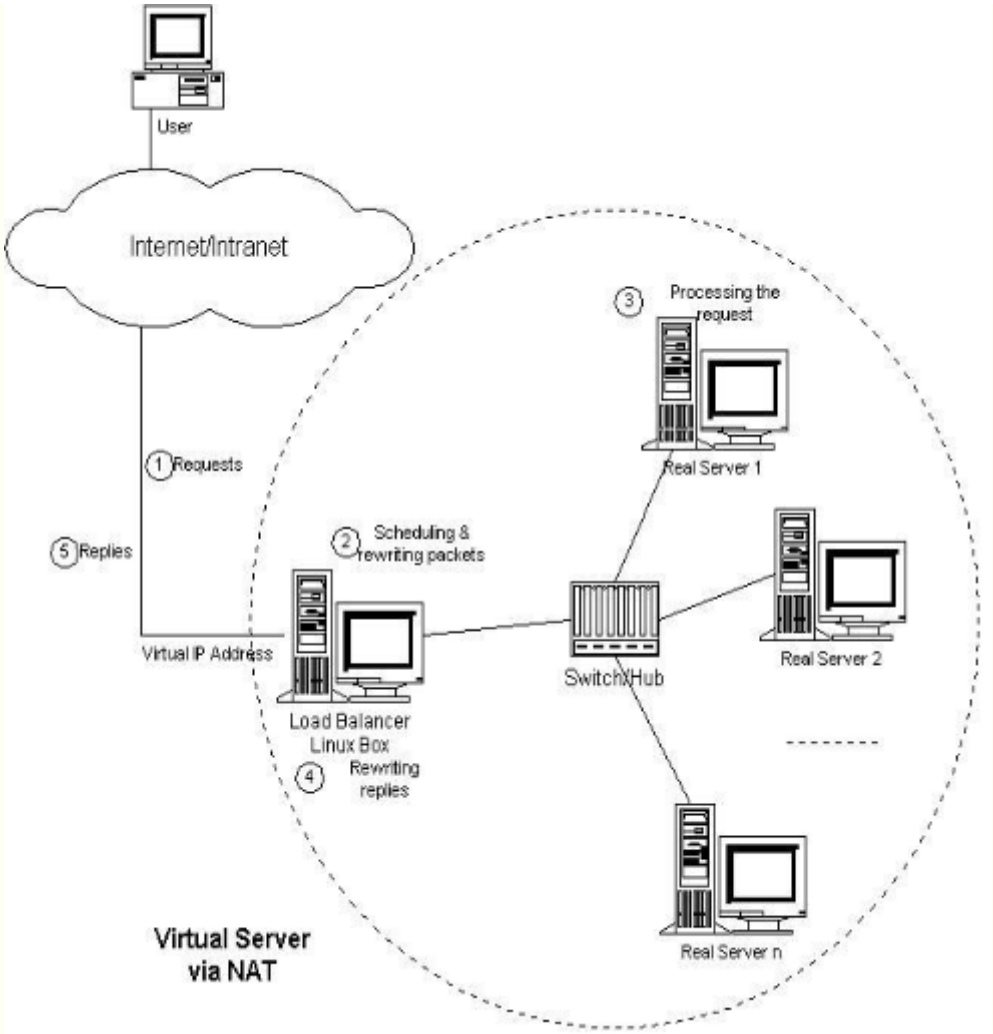
**CIP**：Client IP，访问客户端的IP地址。

下面介绍LVS常用的三种负载均衡模式

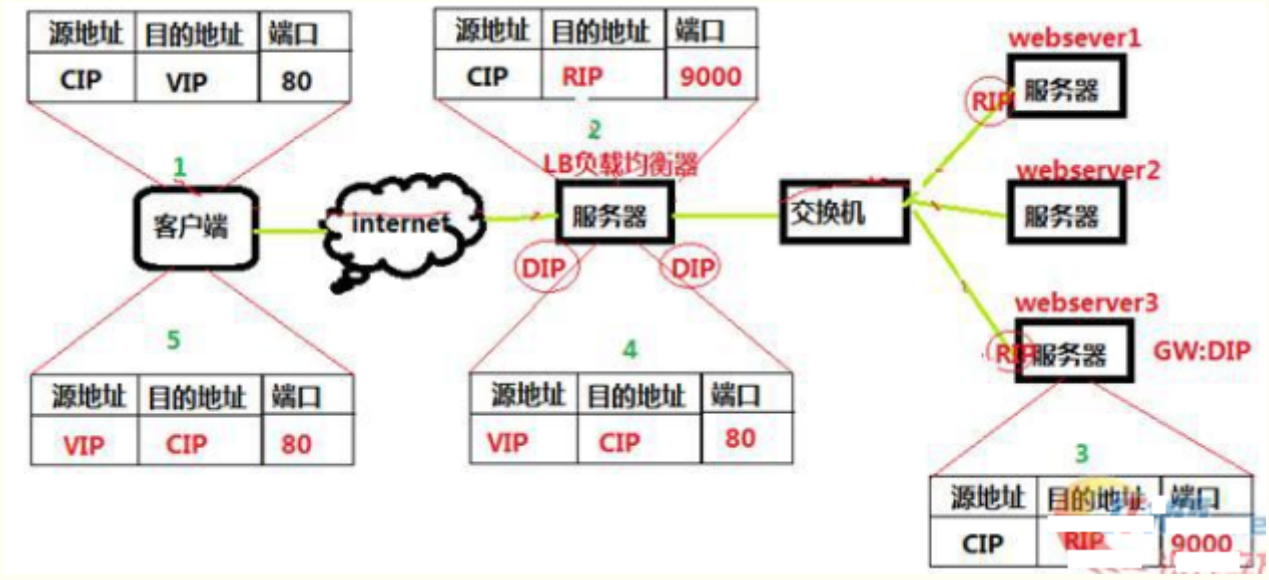
### 1）NAT模式-网络地址转换　　Virtualserver via Network address translation(VS/NAT)

这个是通过网络地址转换的方法来实现调度的。首先调度器(LB)接收到客户的请求数据包时（请求的目的IP为VIP），根据调度算法决定将请求发送给哪个后端的真实服务器（RS）。然后调度就把客户端发送的请求数据包的目标IP地址及端口改成后端真实服务器的IP地址（RIP）,这样真实服务器（RS）就能够接收到客户的请求数据包了。真实服务器响应完请求后，查看默认路由（NAT模式下我们需要把RS的默认路由设置为LB服务器。）把响应后的数据包发送给LB,LB再接收到响应包后，把包的源地址改成虚拟地址（VIP）然后发送回给客户端。

VS/NAT是一种最简单的方式，所有的RealServer只需要将自己的网关指向Director即可。客户端可以是任意操作系统，但此方式下，一个Director能够带动的RealServer比较有限。在VS/NAT的方式下，Director也可以兼为一台RealServer。VS/NAT的体系结构如图所示。




NAT工作模式下，调度过程IP包详细图：




NAT模式的以上原理图简述：

1) 客户端请求数据，目标IP为VIP

2) 请求数据到达LB服务器，LB根据调度算法将目的地址修改为RIP地址及对应端口（此RIP地址是根据调度算法得出的。）并在连接HASH表中记录下这个连接。

3) 数据包从LB服务器到达RS服务器webserver，然后webserver进行响应。Webserver的网关必须是LB，然后将数据返回给LB服务器。

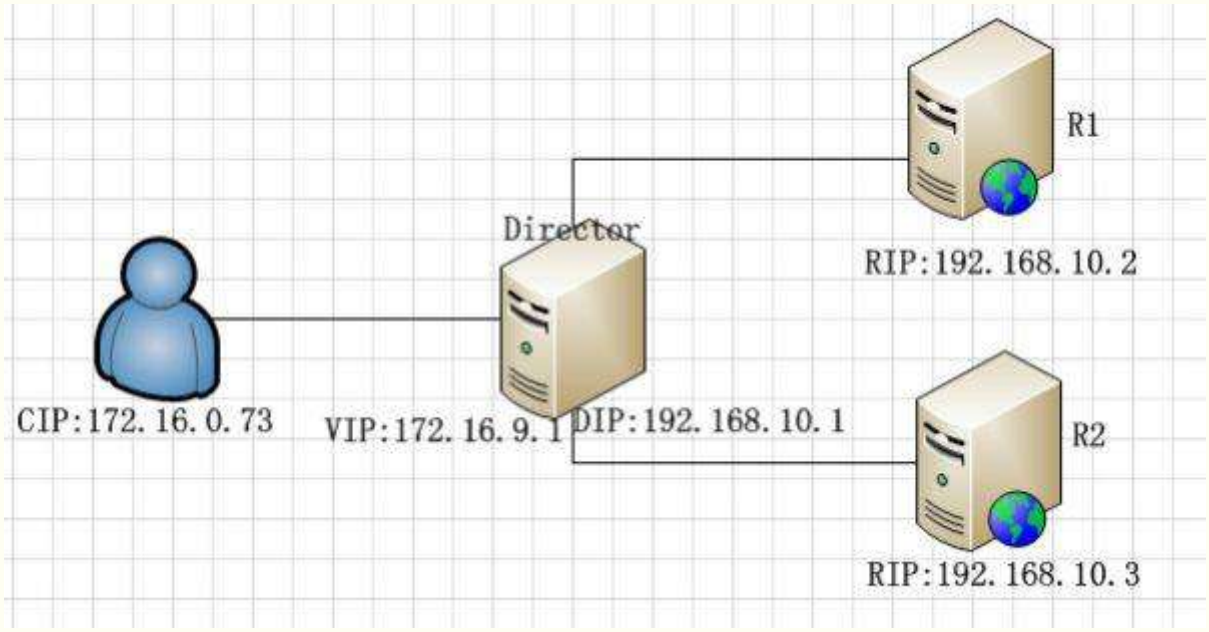
4) 收到RS的返回后的数据，根据连接HASH表修改源地址VIP&目标地址CIP，及对应端口80.然后数据就从LB出发到达客户端。

5) 客户端收到的就只能看到VIP\DIP信息。

NAT模式优缺点：

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) 每台内部的节点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。

再看下面的NAT模式图




客户发出请求，发送请求给链接调度器的VIP，调度器将请求报文中的目标Ip地址改为RIP。这样服务器RealServer将请求的内容发给调度器，调度器再将报文中的源IP地址改为VIP；

1) 当用户请求到达Director Server，此时请求的数据报文会先到内核空间的PREROUTING链。 此时报文的源IP为CIP，目标IP为VIP；
2) PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链；
3) IPVS比对数据包请求的服务是否为集群服务，若是，修改数据包的目标IP地址为后端服务器IP，然后将数据包发至POSTROUTING链。 此时报文的源IP为CIP，目标IP为RIP
4) POSTROUTING链通过选路，将数据包发送给Real Server；
5) Real Server比对发现目标为自己的IP，开始构建响应报文发回给Director Server。 此时报文的源IP为RIP，目标IP为CIP；
6) Director Server在响应客户端前，此时会将源IP地址修改为自己的VIP地址，然后响应给客户端。 此时报文的源IP为VIP，目标IP为CIP；

NAT模式特点和注意事项：

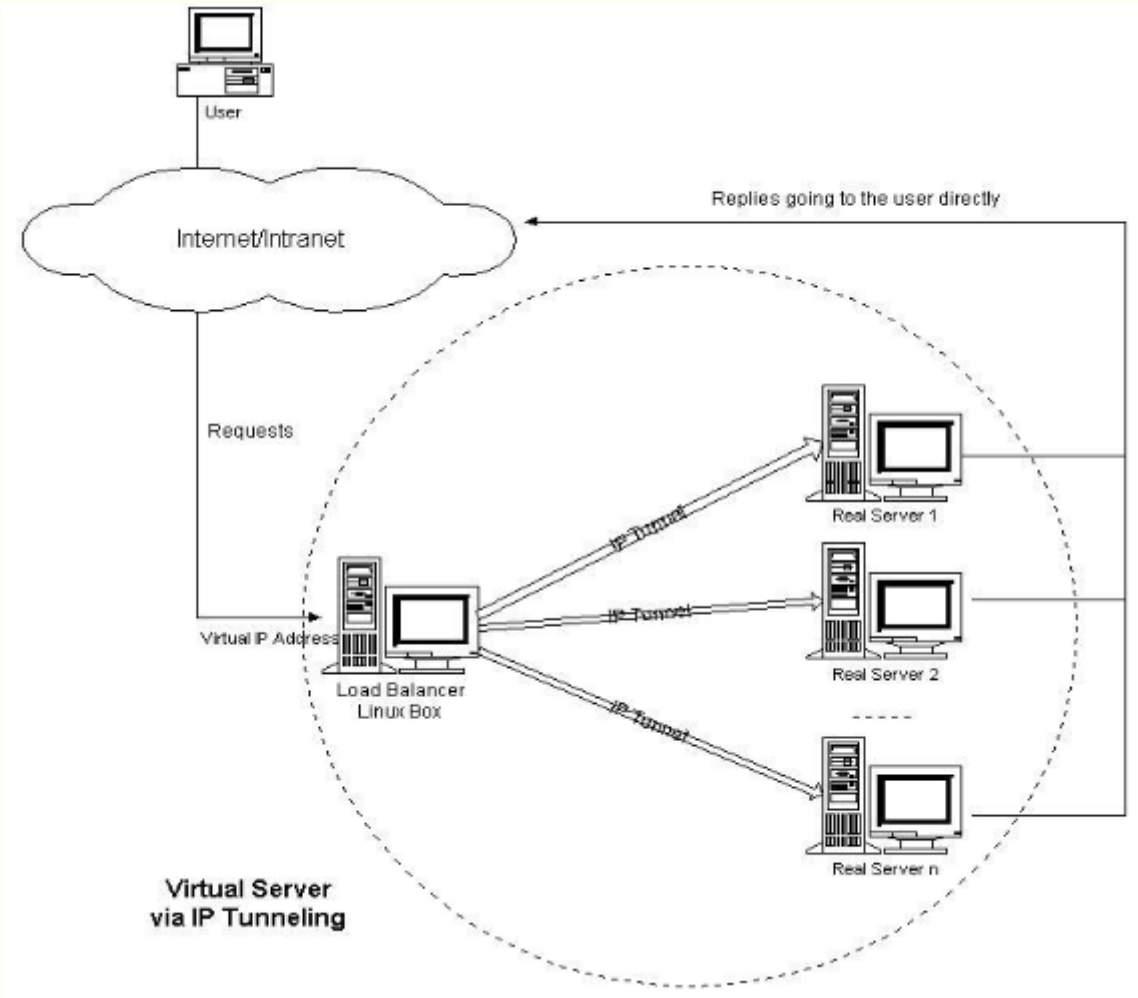
1) 很好配置，原理简单易懂；
2) 由于调度器的工作量太大，很容易成为整个集群系统的瓶颈；
3) RS应该使用私有地址；
4) RS的网关的必须指向DIP；
5) RIP和DIP必须在同一网段内；
6) 请求和响应的报文都得经过Director；在高负载场景中，Director很可能成为系统性能瓶颈；
7) 支持端口映射；
8) RS可以使用任意支持集群服务的OS；

### 2) TUN模式-IP隧道模式　　Virtual Server via IP Tunneling(VS/TUN)

IP隧道(IP tunneling)是将一个IP报文封装在另一个IP报文的技术，这可以使得目标为一个IP地址的数据报文能被封装和转发到另一个IP地址。IP隧道技术亦称为IP封装技术(IP encapsulation)。

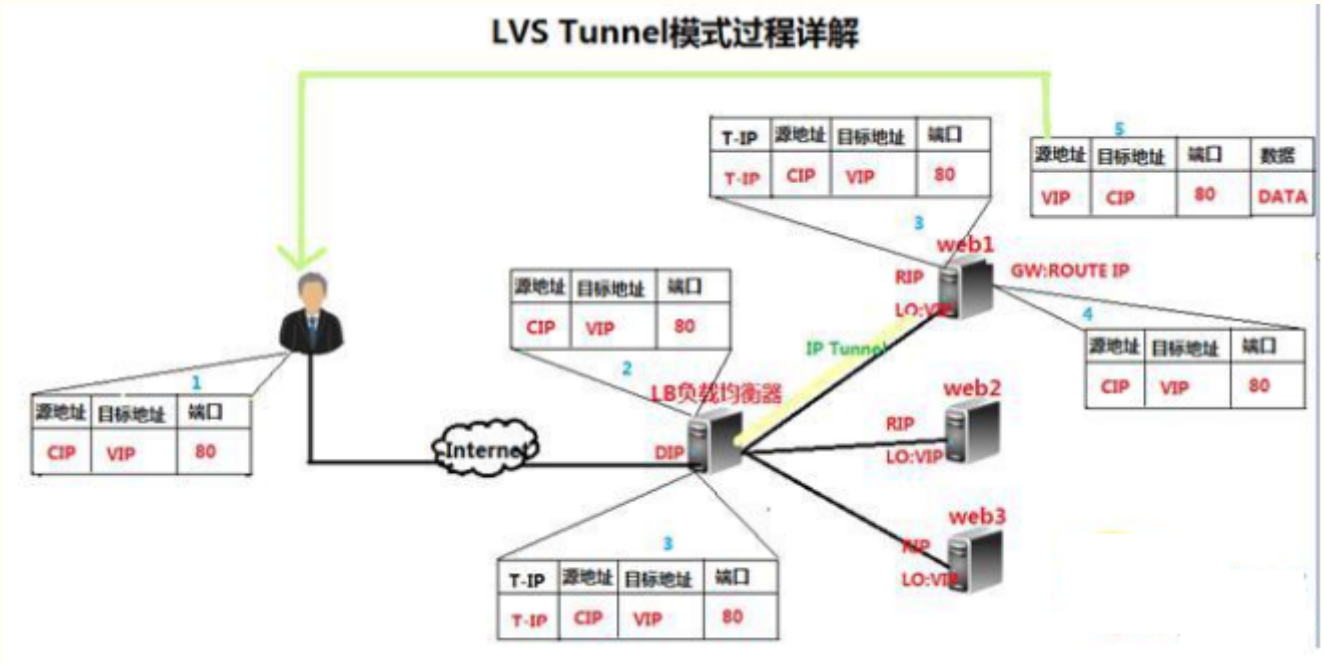
IP隧道主要用于移动主机和虚拟私有网络(Virtual Private Network)，在其中隧道都是静态建立的，隧道一端有一个IP地址，另一端也有唯一的IP地址。它的连接调度和管理与VS/NAT中的一样，只是它的报文转发方法不同。调度器根据各个服务器的负载情况，动态地选择一台服务器，将请求报文封装在另一个IP报文中，再将封装后的IP报文转发给选出的服

务器；服务器收到报文后，先将报文解封获得原来目标地址为 VIP 的报文，服务器发现VIP地址被配置在本地的IP隧道设备上，所以就处理这个请求，然后根据路由表将响应报文直接返回给客户。



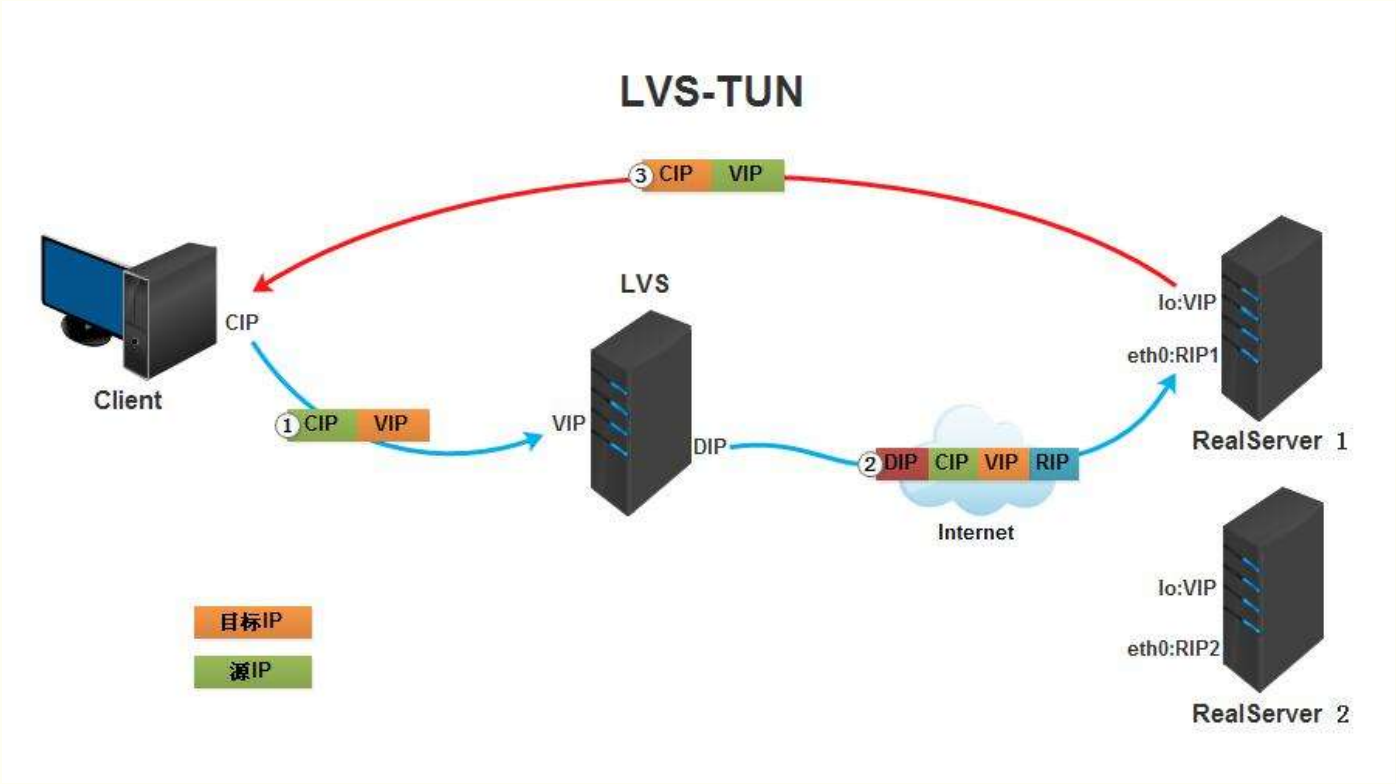

采用NAT模式时，由于请求和响应的报文必须通过调度器地址重写，当客户请求越来越多时，调度器处理能力将成为瓶颈。为了解决这个问题，调度器把请求的报文通过IP隧道转发到真实的服务器。真实的服务器将响应处理后的数据直接返回给客户端。这样调度器就只处理请求入站报文，由于一般网络服务应答数据比请求报文大很多，采用VS/TUN模式后，集群系统的最大吞吐量可以提高10倍。

VS/TUN的工作原理流程图如下所示，它和NAT模式不同的是，它在LB和RS之间的传输不用改写IP地址。而是把客户请求包封装在一个IP tunnel里面，然后发送给RS节点服务器，节点服务器接收到之后解开IP tunnel后，进行响应处理。并且直接把包通过自己的外网地址发送给客户不用经过LB服务器。



TUN模式下的以上原理图过程简述：
1）客户请求数据包，目标地址VIP发送到LB上；
2）LB接收到客户请求包，进行IP Tunnel封装。即在原有的包头加上IP Tunnel的包头。然后发送出去；
3）RS节点机器根据IP Tunnel包头信息 (此时就又一种逻辑上的隐形隧道，只有LB和RS之间懂）收到请求包，然后解开IP Tunnel包头信息，得到客户的请求包并进行响应处理。
4）响应处理完毕之后，RS服务器使用自己的出公网的线路，将这个响应数据包发送给客户端。源IP地址还是VIP地址。（RS节点服务器需要在本地回环接口配置VIP)；

其实TUN模式和下面的DR模式差不多，但是比DR多了一个隧道技术以支持realserver不在同一个物理环境中。就是realserver一个在北京，一个工作在上海。在原有的IP报文外再次封装多一层IP首部，内部IP首部(源地址为CIP，目标IIP为VIP)，外层IP首部(源地址为DIP，目标IP为RIP. 再看下面的TUN模式图:



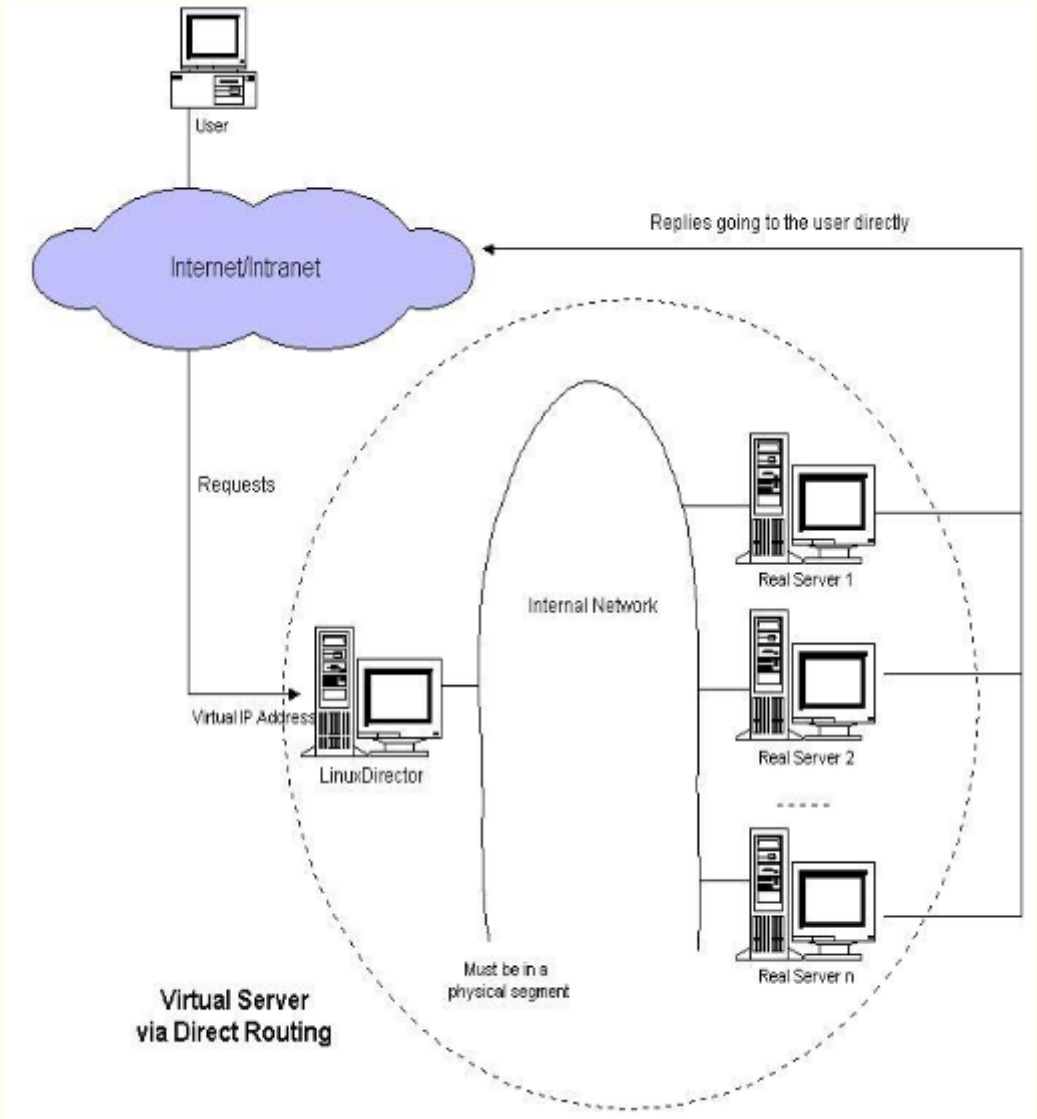

1）当用户请求到达Director Server，此时请求的数据报文会先到内核空间的PREROUTING链。 此时报文的源IP为CIP，目标IP为VIP 。
2）PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链；
3）IPVS比对数据包请求的服务是否为集群服务，若是，在请求报文的首部再次封装一层IP报文，封装源IP为为DIP，目标IP为RIP。然后发至POSTROUTING链。 此时源IP为DIP，目标IP为RIP；
4）POSTROUTING链根据最新封装的IP报文，将数据包发至RS（因为在外层封装多了一层IP首部，所以可以理解为此时通过隧道传输）。 此时源IP为DIP，目标IP为RIP；
5）RS接收到报文后发现是自己的IP地址，就将报文接收下来，拆除掉最外层的IP后，会发现里面还有一层IP首部，而且目标是自己的lo接口VIP，那么此时RS开始处理此请求，处理完成之后，通过lo接口送给eth0网卡，然后向外传递。 此时的源IP地址为VIP，目标IP为CIP；
6）响应报文最终送达至客户端；

LVS-TUN (ip隧道) 模式特点和注意事项
1) RIP、VIP、DIP全是公网地址
2) RS的网关不会也不可能指向DIP
3) 不支持端口映射
4) RS的系统必须支持隧道

### 3）DR模式-直接路由模式　　Virtual Server via Direct Routing(VS/DR)

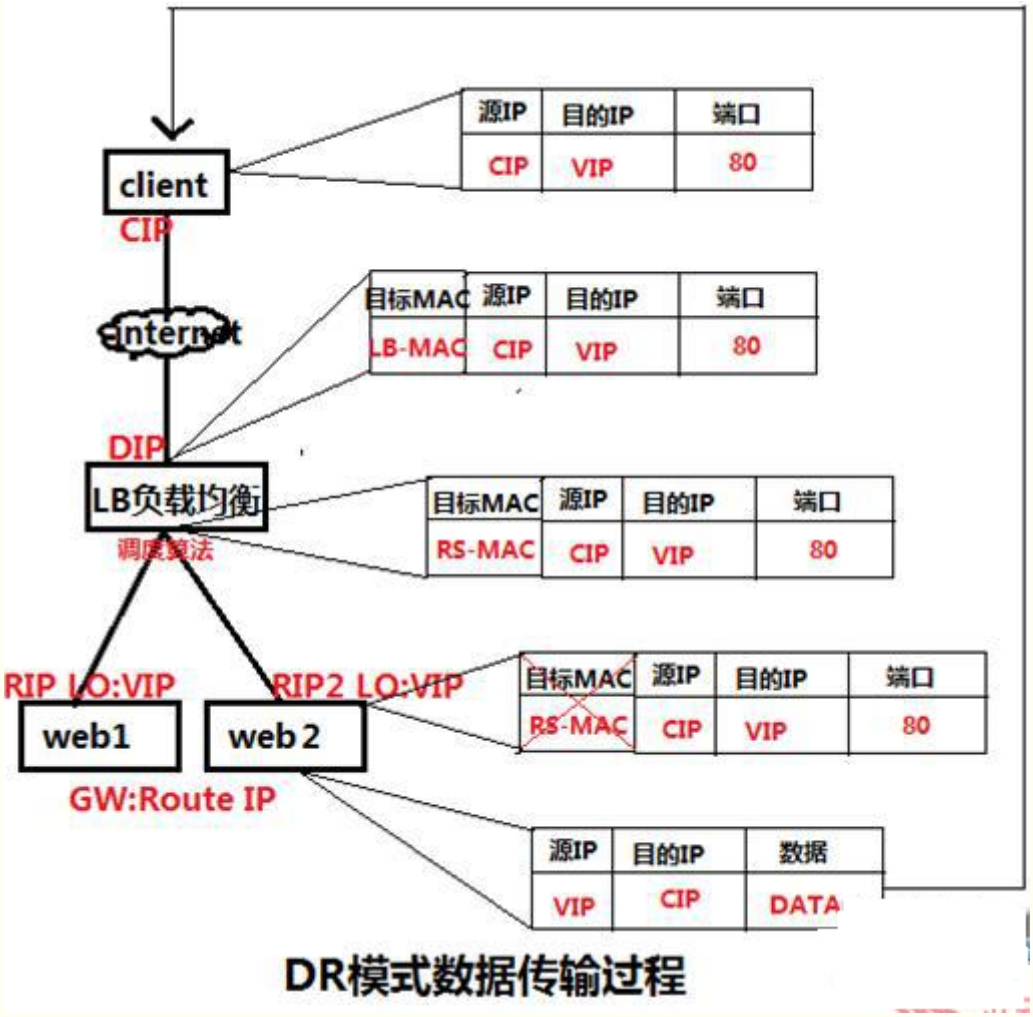
DR模式也就是用直接路由技术实现虚拟服务器。它的连接调度和管理与VS/NAT和VS/TUN中的一样，但它的报文转发方法又有不同，VS/DR通过改写请求报文的MAC地址，将请求发送到Real Server，而Real Server将响应直接返回给客户，免去了VS/TUN中的IP隧道开销。这种方式是三种负载调度机制中性能最高最好的，但是必须要求Director Server与Real Server都有一块网卡连在同一物理网段上。

Director和RealServer必需在物理上有一个网卡通过不间断的局域网相连。 RealServer上绑定的VIP配置在各自Non-ARP的网络设备上(如lo或tunl),Director的VIP地址对外可见，而RealServer的VIP对外是不可见的。RealServer的地址即可以是内部地址，也可以是真实地址。




DR模式是通过改写请求报文的目标MAC地址，将请求发给真实服务器的，而真实服务器响应后的处理结果直接返回给客户端用户。同TUN模式一样，DR模式可以极大的提高集群系统的伸缩性。而且DR模式没有IP隧道的开销，对集群中的真实服务器也没有必要必须支持IP隧道协议的要求。但是要求调度器LB与真实服务器RS都有一块网卡连接到同一物理网段上，必须在同一个局域网环境。

**DR模式是互联网使用比较多的一种模式**，DR模式原理图如下：

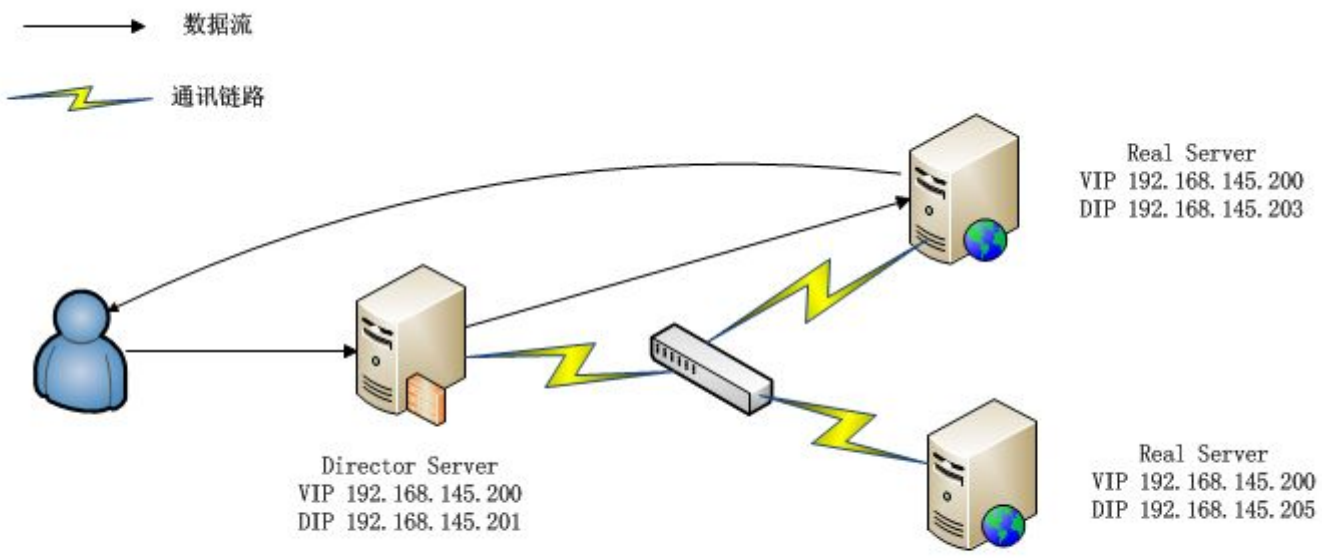



**DR模式以上原理过程简述：**

VS/DR模式的工作流程图如上图所示，它的连接调度和管理与NAT和TUN中的一样，它的报文转发方法和前两种不同。DR模式将报文直接路由给目标真实服务器。在DR模式中，调度器根据各个真实服务器的负载情况，连接数多少等，动态地选择一台服务器，不修改目标IP地址和目标端口，也不封装IP报文，而是将请求报文的数据帧的目标MAC地址改为真实服务器的MAC地址。然后再将修改的数据帧在服务器组的局域网上发送。因为数据帧的MAC地址是真实服务器的MAC地址，并且又在同一个局域网。那么根据局域网的通讯原理，真实复位是一定能够收到由LB发出的数据包。真实服务器接收到请求数据包的时候，解开IP包头查看到的目标IP是VIP。（此时只有自己的IP符合目标IP才会接收进来，所以我们需要在本地的回环借口上面配置VIP。

另外：由于网络接口都会进行ARP广播响应，但集群的其他机器都有这个VIP的lo接口，都响应就会冲突。所以我们需要把真实服务器的lo接口的ARP响应关闭掉。）然后真实服务器做成请求响应，之后根据自己的路由信息将这个响应数据包发送回给客户，并且源IP地址还是VIP。

其实整个DR模式都是停留在第二层的数据链路层，直接修改MAC。实现报文的转发。再看下面的DR模式图：




1）当用户请求到达Director Server，此时请求的数据报文会先到内核空间的PREROUTING链。 此时报文的源IP为CIP，目标IP为VIP；
2）PREROUTING检查发现数据包的目标IP是本机，将数据包送至INPUT链；
3）IPVS比对数据包请求的服务是否为集群服务，若是，将请求报文中的源MAC地址修改为DIP的MAC地址，将目标MAC地址修改RIP的MAC地址，然后将数据包发至POSTROUTING链。 此时的源IP和目的IP均未修改，仅修改了源MAC地址为DIP的MAC地址，目标MAC地址为RIP的MAC地址；
4）由于DS和RS在同一个网络中，所以是通过二层来传输。POSTROUTING链检查目标MAC地址为RIP的MAC地址，那么此时数据包将会发至Real Server；
5）响应报文最终送达至客户端；

LVS-DR模式特点和注意事项

1) 在前端路由器做静态地址路由绑定，将对于VIP的地址仅路由到Director Server
2) arptables：在arp的层次上实现在ARP解析时做防火墙规则，过滤RS响应ARP请求。修改RS上内核参数（arp_ignore和arp_announce）将RS上的VIP配置在网卡接口的别名上，并限制其不能响应对VIP地址解析请求。
3) RS可以使用私有地址；但也可以使用公网地址，此时可以直接通过互联网连入RS以实现配置、监控等；
4) RS的网关一定不能指向DIP；
5) RS跟Dirctory要在同一物理网络内（不能由路由器分隔）；
6) 请求报文经过Directory，但响应报文一定不经过Director
7) 不支持端口映射；
8) RS可以使用大多数的操作系统；

**DR模式小结：**

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接口上（防止IP冲突），并且需要配置ARP机制。
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。

**三种负载均衡方式简单比较：**

1）NAT模式-网络地址转换
VS/NAT 的优点是服务器可以运行任何支持TCP/IP的操作系统，它只需要一个IP地址配置在调度器上，服务器组可以用私有的IP地址。缺点是它的伸缩能力有限，当服务器结点数目升到20时，调度器本身有可能成为系统的新瓶颈，因为在

VS/NAT中请求和响应报文都需要通过负载调度器。如果负载调度器成为系统新的瓶颈，可以有三种方法解决这个问题：混合方法、VS/TUN和 VS/DR。在DNS混合集群系统中，有若干个VS/NAT负调度器，每个负载调度器带自己的服务器集群，同时这些负载调度器又通过RR-DNS组成简单的域名。但VS/TUN和VS/DR是提高系统吞吐量的更好方法。对于那些将IP地址或者端口号在报文数据中传送的网络服务，需要编写相应的应用模块来转换报文数据中的IP地址或者端口号。这会带来实现的工作量，同时应用模块检查报文的开销会降低系统的吞吐率。

2）TUN模式-IP隧道模式
在TUN 的集群系统中，负载调度器只将请求调度到不同的后端服务器，后端服务器将应答的数据直接返回给用户。这样负载调度器就可以处理大量的请求，它甚至可以调度百台以上的服务器(同等规模的服务器)，而它不会成为系统的瓶颈。即使负载调度器只有100Mbps的全双工网卡，整个系统的最大吞吐量可超过 1Gbps。所以，VS/TUN可以极大地增加负载调度器调度的服务器数量。VS/TUN调度器可以调度上百台服务器，而它本身不会成为系统的瓶颈，可以用来构建高性能的超级服务器。VS/TUN技术对服务器有要求，即所有的服务器必须支持"IP Tunneling"或者"IP Encapsulation"协议。目前，VS/TUN的后端服务器主要运行Linux操作系统，我们没对其他操作系统进行测试。因为"IP Tunneling"正成为各个操作系统的标准协议，所以VS/TUN应该会适用运行其他操作系统的后端服务器。

3）DR模式
跟VS/TUN方法一样，VS/DR调度器只处理客户到服务器端的连接，响应数据可以直接从独立的网络路由返回给客户。这可以极大地提高LVS集群系统的伸缩性。跟VS/TUN相比，这种方法没有IP隧道的开销，但是要求负载调度器与实际服务器都有一块网卡连在同一物理网段上，服务器网络设备(或者设备别名)不作ARP响应，或者能将报文重定向(Redirect)到本地的Socket端口上。

三种负载均衡技术比较总结表：

| 工作模式 | VS/NAT | VS/TUN | VS/DR |
|---|---|---|---|
| Real server（节点服务器） | Config dr gw | Tunneling | Non-arp device/tie vip |
| Server Network | Private | LAN/WAN | LAN |
| Server number（节点数量） | Low 10-20 | High 100 | High 100 |
| Real server gateway | Load balance | Own router | Own router |
| 优点 | 地址和端口转换 | Wan环境加密数据 | 性能最高 |
| 缺点 | 效率低 | 需要隧道支持 | 不能跨域LAN |

## 6) LVS负载均衡调度算法

VS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 (如下所列，**LVS总共有10种调度算法，常用的也就四种调度算法，下面会说到**)：
**静态调度算法**：rr，wrr，dh，sh
**动态调度算法**：wlc，lc，lblc，lblcr，sed，nq

静态调度 (也就是固定调度算法)的4种算法：
rr（轮询）
轮询调度：这种是最简单的调度算法，就是将请求A一个，B一个，A一个，B一个 ..... 循环的发。就算A主机挂掉了，调度器还是会将请求发送到A。十分均衡。

wrr（权重，即加权轮询）
加权轮询调度：这种算法是在rr基础上实现的，只不过加了权重，权重范围为1-100，假设A的服务器性能好，就给A的权重设置的高一点，设为2，而B主机是1。这样就实现A二个，B一个，A二个，B一个 ..... 循环的发。这样照顾到了服务器性能。

sh（源地址哈希）
源地址散列：主要是实现将此前的session（会话）绑定。将此前客户的源地址作为散列键，从静态的散列表中找出对应的服务器，只要目标服务器是没有超负荷的就将请求发送过去。就是说某客户访问过A,现在这个客户又来了，所以客户请求会被发送到服务过他的A主机。

dh（目的地址哈希）
目的地址散列：以目的地址为关键字查找一个静态hash表来获得需要的RS。以目标地址为标准挑选。 功能是和sh近似

的，但应用场景不同；举个dh调度算法的例子：假设1号客户访问了web集群的一个动态页面，调度器将请求转发个A服务器，A服务器的PHP将这个动态请求运行了一遍，生成了缓存并回应1号客户。这下2号客户也访问了这个动态页面，调度器应该将请求发给A。毕竟A已经跑过这段程序了，有缓存，对吧。所以这既是dh算法）

动态调度算法，动态算法与静态算法最大的区别就是动态算法考虑了服务器的压力。
活动链接（active）：客户与服务器建立连接并且有数据传送
非活动链接（inactive）：只是建立连接，没有数据传送，没有断开连接

动态调度的6种算法
lc（最少链接）
最少连接调度：这种算法是看A，和B的主机谁的连接少，请求就发给谁。
简单算法：active*256+inactive （谁小发给谁）

wlc（加权最少链接）LVS的理想算法
加权最少链接：这种算法就是比lc多了一个加权。
简单算法：( active*256+inactive )/weight （谁小就发给谁）

sed（最短期望延迟）
基于wlc算法，假设A，B的权重分别是1，2 。而A的链接数为1，B的链接数为2 。这样的话，用wlc算法得出的结果一样，而明显B的权重大，B的能力较强。用sed算法的话，就可以避免wlc出现的问题。
简单算法：（active+1)*256/weight （活动的连接数+1）*256/除以权重 谁小发给谁
A：（1+1）/1
B：（2+1）/2 （B小，交给B）

nq（不用排队）
基于sed算法：在sed的基础上，若谁的链接数为0，直接将请求发送给它!

LBLC（基于局部性的最少连接）类似于dh，目标地址hash
这个算法主要用于Cache集群系统，因为Cache集群的中客户请求报文的目标IP地址的变化，将相同的目标URL地址请求调度到同一台服务器，来提高服务器的访问的局部性和Cache命中率。从而调整整个集群的系统处理能力。但是，如果realserver的负载处于一半负载，就用最少链接算法，将请求发送给活动链接少的主机。

LBLCR（带复制的基于局部性的最少链接）
该算法首先是基于最少链接的，当一个新请求收到后，一定会将请求发给最少连接的那台主机的。但这样又破坏了cache命中率。但这个算法中，集群服务是cache共享的，假设A的PHP跑了一遍，得到缓存。但其他realserver可以去A那里拿缓存，这是种缓存复制机制。

负载调度器是根据各个服务器的负载情况，动态地选择一台Real Server响应用户请求，那么动态选择是如何实现呢，其实也就是这里要说的负载调度算法，根据不同的网络服务需求和服务器配置，IPVS实现了如上的十种负载调度算法，下面详细讲述LVS最常用的四种调度算法：

**- 轮叫调度（Round Robin）**
"轮叫"调度也叫1:1调度，调度器通过"轮叫"调度算法将外部用户请求按顺序1:1的分配到集群中的每个Real Server上，这种算法平等地对待每一台Real Server，而不管服务器
上实际的负载状况和连接状态。

**- 加权轮叫调度（Weighted Round Robin）**
"加权轮叫"调度算法是根据Real Server的不同处理能力来调度访问请求。可以对每台Real Server设置不同的调度权值，对于性能相对较好的Real Server可以设置较高的权值，而对于处理能力较弱的Real Server，可以设置较低的权值，这样保证了处理能力强的服务器处理更多的访问流量。充分合理的利用了服务器资源。同时，调度器还可以自动查询Real Server的负载情况，并动态地调整其权值。

**- 最少链接调度（Least Connections）**
"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。

**- 加权最少链接调度（Weighted Least Connections）**
"加权最少链接调度"是"最少连接调度"的超集，每个服务节点可以用相应的权值表示其处理能力，而系统管理员可以动态的设置相应的权值，缺省权值为1，加权最小连接调度在分配新连接请求时尽可能使服务节点的已建立连接数和其权值成正比。

| 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。 |

**LVS调度算法的生产环境选型：**

1）一般的网络服务，如http，nginx，mysql等常用的LVS调度算法为：

a. 基本轮询调度rr

b. 加权最小连接调度wlc

c. 加权轮询调度wrc

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；

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；

其实对于LVS的理解，主要部分还是在于3种工作方式和8种调度算法，实际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。

**LVS的 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共享或服务器机制：利用单独部署的服务器来统一管理session；

**LVS使用中特别需要注意事项：**

1) 关于时间同步：各节点间的时间偏差不大于1s，建议使用统一的ntp服务器进行更新时间；

2) DR模型中的VIP的MAC广播问题：

在DR模型中，由于每个节点均要配置VIP，因此存在VIP的MAC广播问题，在现在的linux内核中，都提供了相应kernel参数对MAC广播进行管理，具体如下：

arp_ignore：定义接收到ARP请求时的响应级别；

0：只要本地配置的有相应地址，就给予响应；

1：仅在请求的目标地址配置在到达的接口上的时候，才给予响应；DR模型使用

arp_announce：定义将自己地址向外通告时的通告级别；

0：将本地任何接口上的任何地址向外通告；

1：试图仅向目标网络通告与其网络匹配的地址；

2：仅向与本地接口上地址匹配的网络进行通告；DR模型使用

## 五. LVS安装和简单管理（ipvsadm）

LVS全称为Linux Virtual Server，工作在ISO模型中的第四层，由于其工作在第四层，因此与iptables类似，必须工作在内核空间上。因此lvs与iptables一样，是直接工作在内核中的，叫ipvs，主流linux发行版默认都已经集成了ipvs，因

此用户只需安装一个管理工具ipvsadm即可，ipvsadm是LVS在应用层的管理命令，可以通过这个命令去管理LVS的配置。

## 1) 安装LVS

```
先安装依赖
[root@localhost ~]# yum install -y libnl* popt*

查看是否加载lvs模块
[root@localhost ~]# modprobe -l |grep ipvs
kernel/net/netfilter/ipvs/ip_vs.ko
kernel/net/netfilter/ipvs/ip_vs_rr.ko
kernel/net/netfilter/ipvs/ip_vs_wrr.ko
kernel/net/netfilter/ipvs/ip_vs_lc.ko
kernel/net/netfilter/ipvs/ip_vs_wlc.ko
kernel/net/netfilter/ipvs/ip_vs_lblc.ko
kernel/net/netfilter/ipvs/ip_vs_lblcr.ko
kernel/net/netfilter/ipvs/ip_vs_dh.ko
kernel/net/netfilter/ipvs/ip_vs_sh.ko
kernel/net/netfilter/ipvs/ip_vs_sed.ko
kernel/net/netfilter/ipvs/ip_vs_nq.ko
kernel/net/netfilter/ipvs/ip_vs_ftp.ko
kernel/net/netfilter/ipvs/ip_vs_pe_sip.ko

下载并安装LVS
[root@localhost ~]# cd /usr/local/src/
[root@localhost src]# unlink /usr/src/linux
[root@localhost src]# ln -s /usr/src/kernels/2.6.32-431.5.1.el6.x86_64/ /u
[root@localhost src]# wget http://www.linuxvirtualserver.org/software/kern
[root@localhost src]# tar -zvxf ipvsadm-1.26.tar.gz
[root@localhost src]# cd ipvsadm-1.26
[root@localhost ipvsadm-1.26]# make && make install

LVS安装完成，查看LVS集群
[root@localhost ~]# ipvsadm -L -n
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
```

## 2) ipvsadm基本命令说明

```
1) ipvsadm的基本用法：
# ipvsadm COMMAND [protocol] service address
                [scheduling-method] [persistence options]

# ipvsadm command [protocol] service address
                server-address [packet-forwarding-method] [weight options]

第一条命令用于向LVS系统中添加一个用于负载均衡的virtual server（VS）；
```

```
第二条命令用来修改已经存在的VS的配置，service address用来指定涉及的虚拟服务即虚

2) ipvsadm的帮助信息
[root@localhost ~]# ipvsadm --help
ipvsadm v1.26 2008/5/15 (compiled with popt and IPVS v1.2.1)
Usage:
  ipvsadm -A|E -t|u|f service-address [-s scheduler] [-p [timeout]] [-M ne
  ipvsadm -D -t|u|f service-address
  ipvsadm -C
  ipvsadm -R
  ipvsadm -S [-n]
  ipvsadm -a|e -t|u|f service-address -r server-address [options]
  ipvsadm -d -t|u|f service-address -r server-address
  ipvsadm -L|l [options]
  ipvsadm -Z [-t|u|f service-address]
  ipvsadm --set tcp tcpfin udp
  ipvsadm --start-daemon state [--mcast-interface interface] [--syncid sid
  ipvsadm --stop-daemon state
  ipvsadm -h

Commands:
Either long or short options are allowed.
  --add-service     -A        add virtual service with options
  --edit-service    -E        edit virtual service with options
  --delete-service  -D        delete virtual service
  --clear           -C        clear the whole table
  --restore         -R        restore rules from stdin
  --save            -S        save rules to stdout
  --add-server      -a        add real server with options
  --edit-server     -e        edit real server with options
  --delete-server   -d        delete real server
  --list            -L|-l     list the table
  --zero            -Z        zero counters in a service or all services
  --set tcp tcpfin udp        set connection timeout values
  --start-daemon              start connection sync daemon
  --stop-daemon               stop connection sync daemon
  --help            -h        display this help message

Options:
  --tcp-service  -t service-address   service-address is host[:port]
  --udp-service  -u service-address   service-address is host[:port]
  --fwmark-service  -f fwmark         fwmark is an integer greater than ze
  --ipv6         -6                   fwmark entry uses IPv6
  --scheduler    -s scheduler         one of rr|wrr|lc|wlc|lblc|lblcr|dh|s
                                      the default scheduler is wlc.
  --pe            engine              alternate persistence engine may be
                                      not set by default.
  --persistent   -p [timeout]         persistent service
```

```
  --netmask      -M netmask           persistent granularity mask
  --real-server  -r server-address    server-address is host (and port)
  --gatewaying   -g                   gatewaying (direct routing) (default
  --ipip         -i                   ipip encapsulation (tunneling)
  --masquerading -m                   masquerading (NAT)
  --weight       -w weight            capacity of real server
  --u-threshold  -x uthreshold        upper threshold of connections
  --l-threshold  -y lthreshold        lower threshold of connections
  --mcast-interface interface         multicast interface for connection s
  --syncid sid                        syncid for connection sync (default=
  --connection   -c                   output of current IPVS connections
  --timeout                           output of timeout (tcp tcpfin udp)
  --daemon                            output of daemon information
  --stats                             output of statistics information
  --rate                              output of rate information
  --exact                             expand numbers (display exact values
  --thresholds                        output of thresholds information
  --persistent-conn                   output of persistent connection info
  --nosort                            disable sorting output of service/se
  --sort                              does nothing, for backwards compatib
  --ops          -o                   one-packet scheduling
  --numeric      -n                   numeric output of addresses and port
```

命令：

-A, --add-service： 添加一个集群服务. 即为ipvs虚拟服务器添加一个虚拟服务，也就是

-E, --edit-service： 修改一个虚拟服务。

-D, --delete-service： 删除一个虚拟服务。即删除指定的集群服务;

-C, --clear： 清除所有虚拟服务。

-R, --restore： 从标准输入获取ipvsadm命令。一般结合下边的-S使用。

-S, --save： 从标准输出输出虚拟服务器的规则。可以将虚拟服务器的规则保存，在以后通

-a, --add-server： 为虚拟服务添加一个real server（RS）

-e, --edit-server： 修改RS

-d, --delete-server： 删除

-L, -l, --list： 列出虚拟服务表中的所有虚拟服务。可以指定地址。添加-c显示连接表。

-Z, --zero： 将所有数据相关的记录清零。这些记录一般用于调度策略。

--set tcp tcpfin udp： 修改协议的超时时间。

--start-daemon state： 设置虚拟服务器的备服务器，用来实现主备服务器冗余。（注：

--stop-daemon： 停止备服务器。

-h, --help： 帮助。

参数：

以下参数可以接在上边的命令后边。

-t, --tcp-service service-address： 指定虚拟服务为tcp服务。service-address要

-u, --udp-service service-address： 使用udp服务，其他同上。

-f, --fwmark-service integer： 用firewall mark取代虚拟地址来指定要被负载均衡的

-s, --scheduler scheduling-method： 指定调度算法,默认是wlc。调度算法可以指定以

-p, --persistent [timeout]： 设置持久连接，这个模式可以使来自客户的多个请求被送

-M, --netmask netmask： 指定客户地址的子网掩码。用于将同属一个子网的客户的请求转

```
-r, --real-server server-address:  为虚拟服务指定数据可以转发到的真实服务器的地
[packet-forwarding-method]:  此选项指定某个真实服务器所使用的数据转发模式。需要
-g, --gatewaying:  使用网关（即直接路由），此模式是默认模式。
-i, --ipip:  使用ipip隧道模式。
-m, --masquerading:  使用NAT模式。
-w, --weight weight: 设置权重。权重是0~65535的整数。如果将某个真实服务器的权重
-x, --u-threshold uthreshold:  设置一个服务器可以维持的连接上限。0~65535。设置
-y, --l-threshold lthreshold:  设置一个服务器的连接下限。当服务器的连接数低于此
--mcast-interface interface:  指定使用备服务器时候的广播接口。
--syncid syncid:指定syncid,  同样用于主备服务器的同步。

以下选项用于list命令:
-c, --connection:  列出当前的IPVS连接。
--timeout:  列出超时
--daemon:
--stats:  状态信息
--rate:  传输速率
--thresholds:  列出阈值
--persistent-conn:  坚持连接
--sor:  把列表排序。
--nosort:  不排序
-n, --numeric:  不对ip地址进行dns查询
--exact:  单位
-6:  如果fwmark用的是ipv6地址需要指定此选项。


=========其他注意事项=========
如果使用IPv6地址，需要在地址两端加上"[]"。例如：ipvsadm -A -t [2001:db8::80]:8
可以通过设置以下虚拟文件的值来防御DoS攻击:
/proc/sys/net/ipv4/vs/drop_entry
/proc/sys/net/ipv4/vs/drop_packet
/proc/sys/net/ipv4/vs/secure_tcp
```

### 3) ipvsadm 举例说明

```
一. LVS集群服务管理类举例
1) 添加：-A
# ipvsadm -A -t|u|f service-address [-s scheduler]

举例1: 添加集群
[root@lvs ~]# ipvsadm -A -t 172.16.60.111:80 -s wlc

2) 修改：-E
# ipvsadm -E -t|u|f service-address [-s scheduler]

举例2: 修改集群 (修改集群的调度算法)
[root@lvs ~]# ipvsadm -E -t 172.16.60.111:80 -s wrr

```

```
3) 删除：-D
# ipvsadm -D -t|u|f service-address

举例3: 删除集群
[root@lvs ~]# ipvsadm -D -t 172.16.60.111:80


二. 管理LVS集群中的RealServer举例
1) 添加RS : -a
# ipvsadm -a -t|u|f service-address -r server-address [-g|i|m] [-w weight]

举例1: 往VIP资源为172.16.60.111的集群服务里添加两个realserver
[root@lvs ~]# ipvsadm -a -t 172.16.60.111:80 -r 172.16.60.120 –g -w 5
[root@lvs ~]# ipvsadm -a -t 172.16.60.111:80 -r 172.16.60.130 –g -w 10

2) 修改RS : -e
# ipvsadm -e -t|u|f service-address -r server-address [-g|i|m] [-w weight]

举例2: 修改172.16.60.111集群服务里172.16.60.120这个realserver的权重为3
[root@lvs ~]# ipvsadm -e -t 172.16.60.111:80 -r 172.16.60.120 –g -w 3

3) 删除RS : -d
# ipvsadm -d -t|u|f service-address -r server-address

举例3: 删除172.16.60.111集群服务里172.16.60.120这个realserver
[root@lvs ~]# ipvsadm -d -t 172.16.60.111:80 -r 172.16.60.120


三. 管理LVS集群服务的查看
# ipvsadm -L|l [options]
   options可以为：
   -n：数字格式显示
   --stats 统计信息
   --rate：统计速率
   --timeout：显示tcp、tcpinfo、udp的会话超时时长
   -c：连接客户端数量

举例1: 查看lvs集群转发情况
[root@lvs ~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port         Forward Weight ActiveConn InActConn
TCP  172.16.60.111:80 wlc persistent 600
  -> 172.16.60.205:80             Route   1      0          0
  -> 172.16.60.206:80             Route   1      0          0

举例2: 查看lvs集群的连接状态
[root@lvs ~]# ipvsadm -l --stats
IP Virtual Server version 1.2.1 (size=4096)
```

```
Prot LocalAddress:Port               Conns   InPkts  OutPkts  InBytes OutB
  -> RemoteAddress:Port
TCP  172.16.60.111                     4        6        0
  -> 172.16.60.205:80                  0        0        0
  -> 172.16.60.206:80                  4        6        0

说明：
Conns    (connections scheduled)  已经转发过的连接数
InPkts   (incoming packets)       入包个数
OutPkts  (outgoing packets)       出包个数
InBytes  (incoming bytes)         入流量（字节）
OutBytes (outgoing bytes)         出流量（字节）

举例3: 查看lvs集群的速率
[root@lvs ~]# ipvsadm -l --rate
Prot LocalAddress:Port               CPS    InPPS   OutPPS    InBPS   OutBP
  -> RemoteAddress:Port
TCP  172.16.60.111                     0        0        0        0
  -> 172.16.60.205:80                  0        0        0        0
  -> 172.16.60.206:80                  0        0        0        0

说明：
CPS      (current connection rate)   每秒连接数
InPPS    (current in packet rate)    每秒的入包个数
OutPPS   (current out packet rate)   每秒的出包个数
InBPS    (current in byte rate)      每秒入流量（字节）
OutBPS   (current out byte rate)      每秒入流量（字节）

4) 清除计数器：
# ipvsadm -Z [-t|u|f service-address]

5) 清除规则 (删除所有集群服务)，该命令与iptables的-F功能类似，执行后会清除所有规
# ipvsadm -C

6) 保存规则：
# ipvsadm -S > /path/to/somefile
# ipvsadm-save > /path/to/somefile
# ipvsadm-restore < /path/to/somefile

================================================================================
一.  使用NAT模式
1) 添加vip地址：172.16.60.111
[root@lvs ~]# /sbin/ifconfig eth0:0 172.16.60.111 broadcast 172.16.60.111
[root@lvs ~]# /sbin/route add -host 172.16.60.111 dev eth0:0
[root@lvs ~]# /sbin/arping -I eth0 -c 5 -s 172.16.60.111 172.16.60.1 >/dev

2) 比如添加地址为172.16.60.111:80的lvs集群服务，指定调度算法为轮转。
[root@lvs ~]# ipvsadm -A -t 172.16.60.111:80 -s rr
```

1) 添加真实服务器，指定传输模式为NAT

```
[root@lvs ~]# ipvsadm -a -t 172.16.60.111:80 -r 172.16.60.180:80 -m
[root@lvs ~]# ipvsadm -a -t 172.16.60.111:80 -r 172.16.60.181:80 -m
[root@lvs ~]# ipvsadm -a -t 172.16.60.111:80 -r 172.16.60.182:80 -m
```

NAT模式是lvs的三种模式中最简单的一种。此种模式下只需要保证调度服务器与真实服务器

二. 使用DR模式

1) 对于DR模式首先要配置真实服务器：

```
[root@rs-01 ~]# vim /etc/init.d/realserver
#!/bin/sh
VIP=172.16.60.111
. /etc/rc.d/init.d/functions

case "$1" in
# 禁用本地的ARP请求、绑定本地回环地址
start)
    /sbin/ifconfig lo down
    /sbin/ifconfig lo up
    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
    /sbin/sysctl -p >/dev/null 2>&1
    /sbin/ifconfig lo:0 $VIP netmask 255.255.255.255 up
    /sbin/route add -host $VIP dev lo:0
    echo "LVS-DR real server starts successfully.\n"
    ;;
stop)
    /sbin/ifconfig lo:0 down
    /sbin/route del $VIP >/dev/null 2>&1
    echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
    echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
echo "LVS-DR real server stopped.\n"
    ;;
status)
    isLoOn=`/sbin/ifconfig lo:0 | grep "$VIP"`
    isRoOn=`/bin/netstat -rn | grep "$VIP"`
    if [ "$isLoON" == "" -a "$isRoOn" == "" ]; then
        echo "LVS-DR real server has run yet."
    else
        echo "LVS-DR real server is running."
    fi
    exit 3
    ;;
*)
    echo "Usage: $0 {start|stop|status}"
```

```
    exit 1
esac
exit 0


在真实服务器上执行上面的脚本
[root@rs-01 ~]# chmod 755 /etc/init.d/realserver
[root@rs-01 ~]# /etc/init.d/realserver  start

上面脚本执行后，真实服务器上就在lo:0设备上配置了vip地址，可以使用"ifconfig"命令

2）在LVS机器上接着添加ipvs规则：

先添加vip地址：172.16.60.111
[root@lvs ~]# /sbin/ifconfig eth0:0 172.16.60.111 broadcast 172.16.60.111
[root@lvs ~]# /sbin/route add -host 172.16.60.111 dev eth0:0
[root@lvs ~]# /sbin/arping -I eth0 -c 5 -s 172.16.60.111 172.16.60.1 >/dev

添加地址为172.16.60.111:80的lvs集群服务，指定调度算法为轮转。
[root@lvs ~]# ipvsadm -A -t 172.16.60.111:80 -s rr

添加真实服务器，指定传输模式为DR
[root@lvs ~]# ipvsadm -a -t 172.16.60.111:80 -r 172.16.60.180:80 -g
[root@lvs ~]# ipvsadm -a -t 172.16.60.111:80 -r 172.16.60.181:80 -g
[root@lvs ~]# ipvsadm -a -t 172.16.60.111:80 -r 172.16.60.182:80 -g

注意：此处的例子中客户、调度服务器、真实服务器都是位于同一网段的
```

### 4）小案例分析

172.168.60.208 作为LVS负载代理层，代理后端两个web节点172.16.60.205和172.16.60.206的80端口.
VIP资源为172.16.60.119

```
1）在172.16.60.208服务器上安装LVS （安装方式如上）
[root@lvs-208 ~]# yum install -y libnl* popt*
[root@lvs-208 ~]# cd /usr/local/src/
[root@lvs-208 src]# unlink /usr/src/linux
[root@lvs-208 src]# ln -s /usr/src/kernels/2.6.32-431.5.1.el6.x86_64/ /usr
[root@lvs-208 src]# wget http://www.linuxvirtualserver.org/software/kernel
[root@lvs-208 src]# tar -zvxf ipvsadm-1.26.tar.gz
[root@lvs-208 src]# cd ipvsadm-1.26
[root@lvs-208 ipvsadm-1.26]# make && make install
[root@lvs-208 ipvsadm-1.26]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn

2）在后端两个web节点(realserver)上配置vip （连个realserver节点操作一样）
[root@rs-205 ~]# vim /etc/init.d/realserver
#!/bin/sh
```

```
VIP=172.16.60.119
. /etc/rc.d/init.d/functions

case "$1" in
# 禁用本地的ARP请求、绑定本地回环地址
start)
    /sbin/ifconfig lo down
    /sbin/ifconfig lo up
    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
    /sbin/sysctl -p >/dev/null 2>&1
    /sbin/ifconfig lo:0 $VIP netmask 255.255.255.255 up
    /sbin/route add -host $VIP dev lo:0
    echo "LVS-DR real server starts successfully.\n"
    ;;
stop)
    /sbin/ifconfig lo:0 down
    /sbin/route del $VIP >/dev/null 2>&1
    echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
    echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
echo "LVS-DR real server stopped.\n"
    ;;
status)
    isLoOn=`/sbin/ifconfig lo:0 | grep "$VIP"`
    isRoOn=`/bin/netstat -rn | grep "$VIP"`
    if [ "$isLoON" == "" -a "$isRoOn" == "" ]; then
        echo "LVS-DR real server has run yet."
    else
        echo "LVS-DR real server is running."
    fi
    exit 3
    ;;
*)
    echo "Usage: $0 {start|stop|status}"
    exit 1
esac
exit 0


执行脚本
[root@rs-205 ~]# chmod 755 /etc/init.d/realserver
[root@rs-205 ~]# /etc/init.d/realserver start
LVS-DR real server starts successfully.\n

[root@rs-205 ~]# ifconfig
```

```
......
lo:0      Link encap:Local Loopback
          inet addr:172.16.60.119  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:65536  Metric:1


后端两个web节点的80端口为nginx，nginx安装配置这里省略
[root@rs-205 ~]# ps -ef|grep nginx
root     24154     1  0 Dec25 ?        00:00:00 nginx: master process /usr
nginx    24155 24154  0 Dec25 ?        00:00:02 nginx: worker process
root     24556 23313  0 01:14 pts/1    00:00:00 grep nginx
[root@rs-205 ~]# lsof -i:80
COMMAND   PID  USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
nginx   24154  root    7u  IPv4  85119      0t0  TCP *:http (LISTEN)
nginx   24155 nginx    7u  IPv4  85119      0t0  TCP *:http (LISTEN)


3) 在172.16.60.208服务器上管理LVS
添加LVS集群服务，vip为172.16.60.119
接着添加后面两个realserver，指定传输模式为DR

[root@lvs-208~]# /sbin/iptables -F
[root@lvs-208~]# /sbin/iptables -Z
[root@lvs-208~]# /sbin/ipvsadm -C

[root@lvs-208~]# /sbin/ipvsadm --set 30 5 60
[root@lvs-208~]# /sbin/ifconfig eth0:0 172.16.60.119 broadcast 172.16.60.1
[root@lvs-208~]# /sbin/route add -host 172.16.60.119 dev eth0:0

[root@lvs-208~]# /sbin/ipvsadm -A -t 172.16.60.119:80 -s wlc -p 600
[root@lvs-208~]# /sbin/ipvsadm -a -t 172.16.60.119:80 -r 172.16.60.205:80
[root@lvs-208~]# /sbin/ipvsadm -a -t 172.16.60.119:80 -r 172.16.60.206:80

[root@lvs-208~]# touch /var/lock/subsys/ipvsadm >/dev/null 2>&1
[root@lvs-208~]# /sbin/arping -I eth0 -c 5 -s 172.16.60.119 172.16.60.1 >/

查看vip
[root@lvs-208~]# ifconfig
......

eth0:0    Link encap:Ethernet  HWaddr 00:50:56:AC:5B:56
          inet addr:172.16.60.119  Bcast:172.16.60.119  Mask:255.255.255.2
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

查看lvs集群转发情况
[root@lvs-208~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
```

```
TCP  172.16.60.119:80 wlc persistent 600
  -> 172.16.60.205:80             Route   1      0          0
  -> 172.16.60.206:80             Route   1      0          10

访问http://172.16.60.219/, 就可以负载到两个realserver的80端口了

由于配置了持久化, 则600秒内的客户端请求将会转发到同一个realserver节点上.
如果当前请求转发到172.16.60.206节点上, 则关闭该节点的80端口, 则访问http://172.
因为手动将该节点从lvs集群中踢出去,如下:

[root@lvs-208~]# ipvsadm -d -t 172.16.60.119:80 -r 172.16.60.206

[root@lvs-208~]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  172.16.60.119:80 wlc persistent 600
  -> 172.16.60.205:80             Route   1      0          0

然后访问http://172.16.60.219/ 显示的就是172.16.60.205节点的80端口页面!

以上的LVS没有实现后端realserver节点健康检查机制, 如果要想对后端realserver节点进
则需要结合ldirectord软件,  ldirectord配置里有参数可以实现:
realserver节点故障发生时自动踢出lvs集群;
realserver节点故障恢复后重新加入lvs集群;

=====================================================
ldirectord部分的安装和配置可以参考:  https://www.cnblogs.com/kevingrace/p/10

[root@lvs-208src]# pwd
/usr/local/src
[root@lvs-208src]# ll ldirectord-3.9.5-3.1.x86_64.rpm
-rw-rw-r-- 1 root root 90140 Dec 24 15:54 ldirectord-3.9.5-3.1.x86_64.rpm

[root@lvs-208src]# yum install -y ldirectord-3.9.5-3.1.x86_64.rpm

[root@lvs-208src]# cat /etc/init.d/ldirectord |grep "config file"
#              Using the config file /etc/ha.d/ldirectord.cf
#       It uses the config file /etc/ha.d/ldirectord.cf.

如上查找可知, ldirectord的配置文件为/etc/ha.d/ldirectord.cf

[root@lvs-208src]# cd /usr/share/doc/ldirectord-3.9.5
[root@lvs-208ldirectord-3.9.5]# ll ldirectord.cf
-rw-r--r-- 1 root root 8301 Feb  7  2013 ldirectord.cf
[root@lvs-208ldirectord-3.9.5]# cp ldirectord.cf /etc/ha.d/
[root@lvs-208ldirectord-3.9.5]# cd /etc/ha.d/
[root@lvs-208ha.d]# ll
total 20
```

```
-rw-r--r-- 1 root root 8301 Dec 26 01:44 ldirectord.cf
drwxr-xr-x 2 root root 4096 Dec 26 01:40 resource.d
-rw-r--r-- 1 root root 2082 Mar 24  2017 shellfuncs

配置ldirectord.cf，实现realserver节点的健康检查机制 (根据文件中的配置范例进行修
[root@lvs-208ha.d]# cp ldirectord.cf ldirectord.cf.bak
[root@lvs-208ha.d]# vim ldirectord.cf
checktimeout=3
checkinterval=1
autoreload=yes
logfile="/var/log/ldirectord.log"
quiescent=no                                    #这个参数配置就实现了realserver

virtual=172.16.60.119:80
        real=172.16.60.205:80 gate
        real=172.16.60.206:80 gate
        fallback=127.0.0.1:80 gate     #realserver都故障时，转发请求到lvs本
        service=http
        scheduler=rr
        persistent=600
        #netmask=255.255.255.255
        protocol=tcp
        checktype=negotiate
        checkport=80
        #request="index.html"
        #receive="Test Page"
        #virtualhost=www.x.y.z


重启ldirectord服务
[root@lvs-208ha.d]# /etc/init.d/ldirectord start

[root@lvs-208ha.d]# ps -ef|grep ldirectord
root      4399     1  0 01:48 ?        00:00:00 /usr/bin/perl -w /usr/sbin
root      4428  3750  0 01:50 pts/0    00:00:00 grep ldirectord

这样，后端的realserver就通过ldirectord配置实现了健康检查!

[root@lvs-208ha.d]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  172.16.60.119:80 rr persistent 600
  -> 172.16.60.205:80             Route   1      0          0
  -> 172.16.60.206:80             Route   1      0          0

当172.16.60.205 和 172.16.60.206 中的任意一个节点故障时，该节点就会自动从lvs集
该故障节点恢复后，该节点就会自动重新加入到lvs集群中；整个过程对于前面的客户端访问
```

```
如172.16.60.205节点的80端口挂了，则lvs转发情况:
[root@lvs-208ha.d]# ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  172.16.60.119:80 rr persistent 600
  -> 172.16.60.206:80             Route   1      0          0

当172.16.60.205节点的80端口恢复后，则lvs转发情况
[root@lvs-208ha.d]# ipvsadm -Ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  172.16.60.119:80 rr persistent 600
  -> 172.16.60.205:80             Route   1      0          0
  -> 172.16.60.206:80             Route   1      0          0

这就实现了realserver 层面的高可用了!!!

但是此时lvs层是单点，如果还想实现lvs层的高可用，就要利用keepalived 或 heartbea
```