Red Hat Enterprise Linux 7

负载平衡器管理

配置 Keepalived 和 HAProxy

Last Updated: 2023-11-27

Red Hat Enterprise Linux 7 负载平衡器管理

配置 Keepalived 和 HAProxy

Steven Levine
Red Hat Customer Content Services
slevine@redhat.com

Stephen Wadeley
Red Hat Customer Content Services
swadeley@redhat.com

法律通告

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

构建负载平衡器系统为生产服务提供高度可用且可扩展的解决方案，使用专门的 Linux Virtual Server(LVS)进行通过 Keepalived 和 HAProxy 配置的路由和负载平衡技术。本书讨论使用红帽企业 Linux 7 中的负载平衡器技术配置高性能系统和服务。

目录

第 1 章 LOAD BALANCER 概述

负载平衡器是一组集成软件组件，用于跨一组实际服务器平衡 IP 流量。它由两种主要技术组成，可用于监控群集成员和群集服务：Keepalived 和 HAProxy。keepalived 使用 *Linux 虚拟服务器* (LVS)在主动和被动路由器上执行负载平衡和故障转移任务，而 HAProxy 为 TCP 和 HTTP 应用程序执行负载平衡和高可用性服务。

1.1. KEEPALIVED

keepalived 守护进程在主动和被动 LVS 路由器上运行。所有运行 **keepalived** 的路由器都使用 *虚拟冗余路由协议* (VRRP)。活动路由器定期发送 VRRP 公告；如果备份路由器无法接收这些公告，则会选择一个新的活跃路由器。

在活跃的路由器上，**keepalived** 还可以为实际服务器执行负载平衡任务。

keepalived 是与 LVS 路由器相关的控制流程。在引导时，守护进程由 **systemctl** 命令启动，该命令读取配置文件 **/etc/keepalived/keepalived.conf**。在活跃的路由器上，**keepalived** 守护进程启动 LVS 服务，并根据配置的拓扑监控服务的健康状况。使用 VRRP 时，活动路由器会将定期公告发送到备份路由器。在备份路由器上，VRRP 实例决定了活动路由器的运行状态。如果活跃路由器在用户可配置间隔后无法公告，Keepalived 会启动故障转移。在故障转移期间，会清除虚拟服务器。*新活动路由器控制虚拟 IP 地址* (VIP)、发送 ARP 消息、设置 IPVS 表条目（虚拟服务器），启动健康检查，然后开始发送 VRRP 公告。

keepalived 在第 4 层或传输层执行故障转移，TCP 在其上执行基于连接的数据传输。当实际的服务器无法回复简单超时 TCP 连接时，**keepalived** 检测到服务器已失败，并从服务器池中删除它。

1.2. HAPOXY

HAProxy 为 HTTP 和基于 TCP 的服务（如互联网连接的服务和基于 web 的应用）提供负载平衡服务。根据所选的负载均衡器调度算法，**haproxy** 可以在多个实际服务器池中处理多个事件，用作一个虚拟服务器。调度程序决定连接卷，并在非加权调度中平均分配连接量，或者为可处理加权算法容量较高的服务器分配较高的连接卷。

HAProxy 允许用户定义多个代理服务，并为代理执行流量的负载平衡服务。代理由前端系统和一个或多个后端系统组成。前端系统定义代理侦听的 IP 地址(VIP)和端口，以及用于特定代理的后端系统。

后端系统是实际服务器池，并且定义负载平衡算法。

HAProxy 在第 7 层或应用层上执行负载平衡管理。在大多数情况下，管理员为基于 HTTP 的负载平衡（如生产 Web 应用程序）部署 HAProxy，高可用性基础架构是业务连续性的必要部分。

1.3. KEEPALIVED 和 HAPROXY

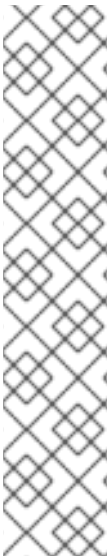
管理员可以将 Keepalived 和 HAProxy 同时用于更加强大和可扩展的高可用性环境。利用 HAProxy 的速度和可扩展性为 HTTP 和其他基于 TCP 的服务执行负载平衡，并与 Keepalived 故障转移服务相结合，管理员可以在实际服务器之间分配负载，并通过执行对备份路由器的故障转移来确保路由器不可用时的连续性。

第 2 章 KEEPALIVED 概述

keepalived 在活动 LVS 路由器以及一个或多个可选的 备份 LVS 路由器上运行。活跃的 LVS 路由器有两个角色：

- 在实际服务器之间平衡负载。
- 检查每个实际服务器上服务的完整性：

活动（主）路由器使用虚拟路由器冗余协议(VRRP)告知备份路由器其活动状态，该协议要求主路由器定期发出公告。如果活动路由器停止发送公告，则会选择新的主路由器。



注意

红帽不支持在配置更改要使用的 VRRP 版本时对 keepalived 进行滚动更新。所有路由器都必须在 keepalived 负载平衡器配置中运行相同的 VRRP 版本。VRRP 版本不匹配将导致以下信息：

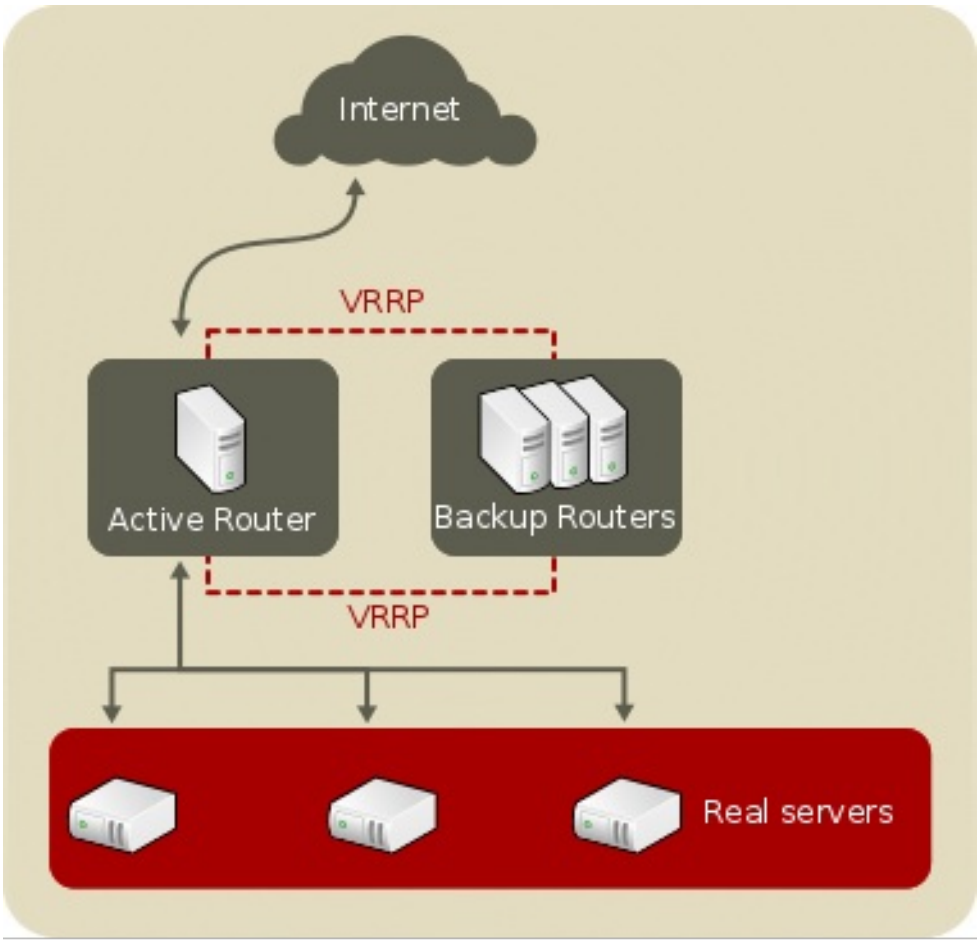
```
Aug  3 17:07:19 hostname Keepalived_vrrp[123]: receive an invalid ip number count associated with VRID!
Aug  3 17:07:19 hostname Keepalived_vrrp[123]: bogus VRRP packet received on em2 !!!
Aug  3 17:07:19 hostname Keepalived_vrrp[123]: VRRP_Instance(vrrp_ipv6) ignoring received advertisment...
```

红帽建议所有系统都运行相同的 keepalived 版本，并且 keepalived 配置应尽可能一致，以避免兼容性问题。

2.1. 基本 KEEPALIVED 负载平衡器配置

图 2.1 “基本负载平衡器配置” 显示由两个层组成的简单的 Keepalived 负载平衡器配置：第一层上有一个活动，多个备用 LVS 路由器。每个 LVS 路由器具有两个网络接口，一个位于互联网，一个在专用网络上，使它们能规范两个网络之间的流量。在本例中，活动路由器使用网络地址转换或 NAT 将流量从互联网定向到第二层上的可变实际服务器，后者又提供必要的服务。因此，本例中的实际服务器连接到专用专用网络段，并通过活动 LVS 路由器来回传递所有公共流量。对外部世界而言，服务器显示为一个实体。

图 2.1. 基本负载平衡器配置



[D]

到达 LVS 路由器的服务请求被寻址到虚拟 IP 地址或 VIP。这是站点管理员可公开路由的地址，与完全限定域名（如 www.example.com）相关联，并分配给一个或多个虚拟服务器。虚拟服务器是配置为侦听特定虚拟 IP 的服务。VIP 地址在故障转移期间从一个 LVS 路由器迁移到另一个 LVS 地址，从而保持在该 IP 地址上的存在性。VIP 也称为浮动 IP 地址。

VIP 地址可以分配到将 LVS 路由器连接到互联网的同一设备。例如，如果 **eth0** 连接到互联网，则可以将多个虚拟服务器分配给 **eth0**。此外，每个虚拟服务器也可与每个服务单独的设备关联。例如，可以在位于 192.168.1.111 的 **eth0** 上处理 HTTP 流量，而 FTP 流量则可在 **eth0** 上处理，地址为 192.168.1.222。

在同时涉及一个主动路由器和一个被动路由器的部署场景中，活动路由器的角色是将服务请求从虚拟 IP 地址重定向到实际服务器。重定向基于 第 2.3 节 "keepalived 计划概述" 中进一步描述的 8 个支持的负载平衡算法之一。

活动路由器还通过三个内置健康检查动态监控实际服务器上特定服务的整体健康状况：简单 TCP 连接、HTTP 和 HTTPS。对于 TCP 连接，活动路由器将定期检查它可以连接到特定端口上的实际服务器。对于 HTTP 和 HTTPS，活动路由器将定期获取实际服务器上的 URL 并验证其内容。

备份路由器充当备用系统的角色。路由器故障转移由 VRRP 处理。启动时，所有路由器将加入多播组。此多播组用于发送和接收 VRRP 公告。由于 VRRP 是基于优先级的协议，因此选择优先级最高的路由器。路由器被选为主路由器后，它负责定期向多播组发送 VRRP 公告。

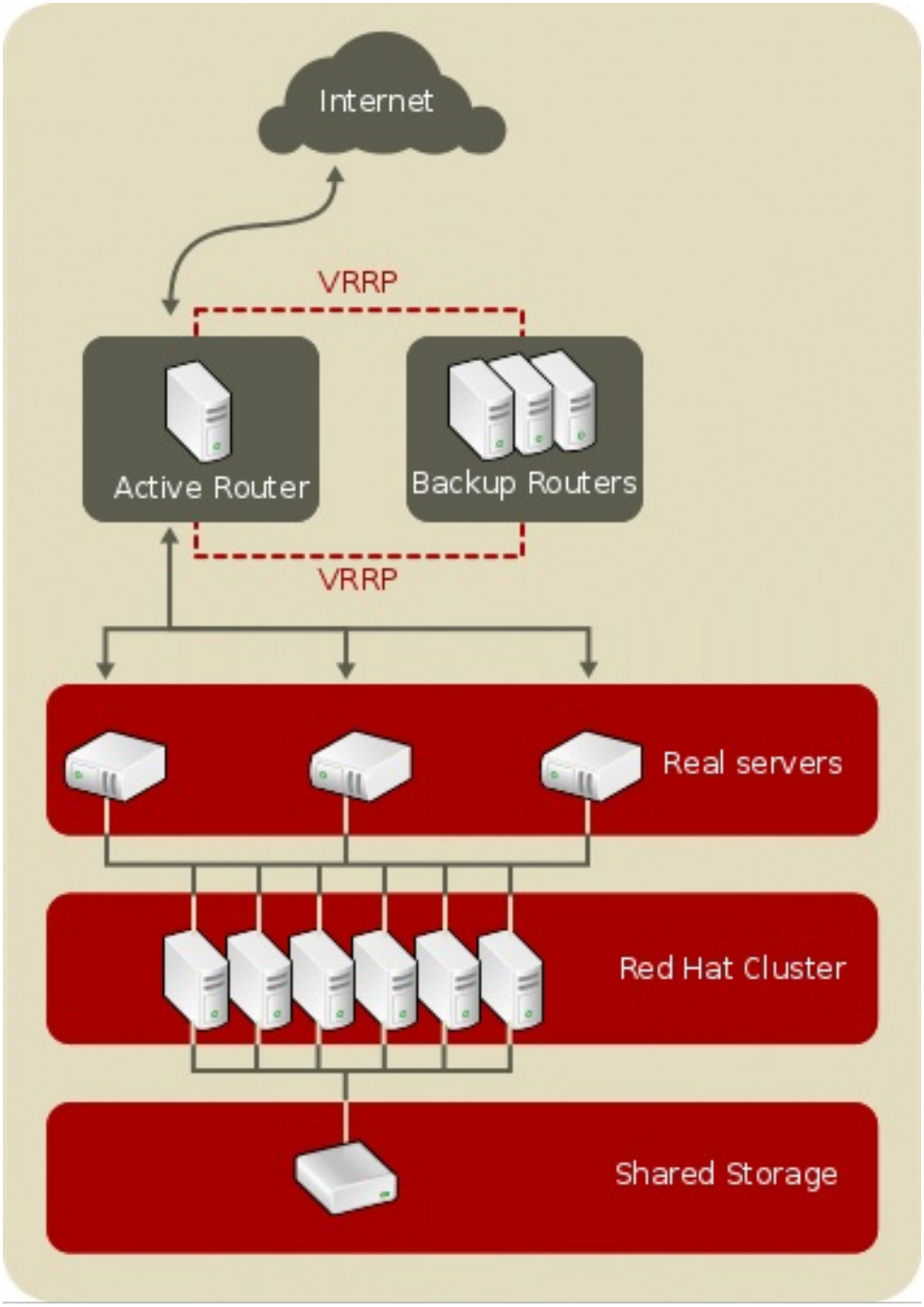
如果备份路由器在特定时间段内（基于广告间隔）无法接收公告，则会选择新的主路由器。新主将接管 VIP 并发送地址解析协议 (ARP) 消息。当路由器返回到活动服务时，它可能成为备份或主设备。该行为由路由器的优先级决定。

图 2.1 “基本负载平衡器配置” 中使用简单、双层的配置最适合提供不经常更改的数据（如静态网页），因为单个实际服务器不会在每个节点间自动同步数据。

2.2. 三层 KEEPALIVED LOAD BALANCER 配置

图 2.2 “三层负载平衡器配置” 显示典型的三层 Keepalived 负载平衡器拓扑。在本例中，活动 LVS 路由器将请求从互联网路由到实际服务器的池。然后，每个实际服务器都会通过网络访问共享数据源。

图 2.2. 三层负载平衡器配置



[D]

此配置非常适合繁忙的 FTP 服务器，其中可访问的数据存储在中央、高度可用的服务器上，并且每个实际服务器通过导出的 NFS 目录或 Samba 共享进行访问。此拓扑也建议用于访问中央、高度可用的事务数据库的网站。此外，管理员可以将主动配置与 Load Balancer 搭配使用，配置一个高可用性群集以同时服务这两个角色。

上例中的第三层不需要使用负载平衡器，但如果无法使用高可用性解决方案，则会引入关键单点故障。

2.3. KEEPALIVED 计划概述

使用 Keepalived 为跨实际服务器分发流量提供了很大的灵活性，其部分原因在于支持的调度算法多种多样。负载平衡优于不灵活的方法，例如 round-Robin DNS，其中 DNS 的层次结构以及客户端机器的缓存会导致负载下降。此外，LVS 路由器使用的低级别过滤优于应用程序级请求转发，因为平衡网络数据包级别的负载会导致计算开销最小，并实现更大的可扩展性。

使用分配的权重为各个机器带来任意优先级。利用这种形式的调度，可以创建一组使用各种硬件和软件组合的实际服务器，活动路由器可以平均加载每个实际服务器。

Keepalived 的调度机制由名为 IP Virtual Server 或 IP VS 模块的内核补丁集合提供。这些模块支持第 4 层(L4)传输层交换，其设计可在单个 IP 地址上与多个服务器配合工作。

为了有效地跟踪和路由数据包到实际服务器，IPVS 在内核中构建 IPVS 表。此表供活动 LVS 路由器用于将来自虚拟服务器地址的请求重定向到 并从池中实际服务器返回。

2.3.1. keepalived Scheduling Algorithms

IPVS 表采用的结构取决于管理员为给定虚拟服务器选择的调度算法。为了最大限度地提高您可以集群的服务类型和这些服务的调度方式，Keepalived 支持以下列出的调度算法。

round-Robin Scheduling

在实际服务器池之间按顺序分发每个请求。使用这种算法时，所有实际服务器都被视为相等，而不考虑容量或负载。此调度模型类似于循环 DNS，但由于它是基于网络连接，而非基于主机，因此更加精细。Load Balancer round-robin 调度也不会遭遇缓存的 DNS 查询导致的静默。

加权轮询调度

在实际服务器池之间按顺序分发每个请求，但为具有更大容量的服务器提供更多作业。容量由用户分配的权重因数表示，然后通过动态负载信息向上或向下调整。

如果池中实际服务器的容量存在显著差异，则加权循环调度是首选。但是，如果请求负载有显著变化，权重较高的服务器可能会回答超过其请求份额。

minimum-Connection

将更多请求分发到活动连接较少的实际服务器。由于它通过 IPVS 表跟踪与实际服务器的实时连接，因此最小连接是一种动态调度算法，因此在请求负载中存在高度变化时做出更好的选择。它最适合一个实际的服务器池，每个成员节点具有大致相同的容量。如果一组服务器具有不同的功能，则优先选择权重最小连接调度。

加权 Least-Connections

将更多请求分发到与其容量相关的活动连接较少的服务器。容量由用户分配的权重指示，然后通过动态负载信息向上或向下调整。添加权重使得这种算法在实际服务器池中包含不同容量的硬件时非常理想。

基于本地性的 Least-Connection 调度

将更多请求发送到与其目标 IP 相关的活动连接较少的服务器。此算法设计为用于代理缓存服务器群集。它将 IP 地址的数据包路由到该地址的服务器，除非该服务器超过其容量，并且具有半负载的服务器，在这种情况下，它会将该 IP 地址分配给最小加载的实际服务器。

使用复制调度基于本地的 Least-Connection 计划

将更多请求发送到与其目标 IP 相关的活动连接较少的服务器。此算法也设计为用于代理缓存服务器群集。它与基于位置的 Least-Connection 调度不同，方法是将目标 IP 地址映射到实际服务器节点的子集。然后，请求被路由到连接数量最低的子集中的服务器。如果目标 IP 的所有节点都超过容量，它会通过将实际服务器的总体连接最少的实际服务器添加到该目标 IP 的实际服务器的子集来复制该目标 IP 的新服务器。然后，最加载的节点从实际的服务器子集中丢弃，以防止过度复制。

目标哈希调度

通过在静态散列表中查找目标 IP，将请求分发到实际服务器池中。此算法设计为用于代理缓存服务器群集。

Source Hash Scheduling

通过在静态散列表中查找源 IP，将请求分发到实际服务器池中。此算法专为具有多个防火墙的 LVS 路由器设计。

最短的预期 Delay

将连接请求分发到服务器，根据给定服务器上的连接数预期最短的延时值，除以其分配的权重。

Never Queue

个双管调度程序，首先查找连接请求并将其发送到闲置或没有连接的服务器。如果没有闲置服务器，调度程序会默认使用与 Shortest 预期 Delay 相同的延迟最小的服务器。

2.3.2. 服务器每周和调度

负载平衡器管理员可以为真实服务器池中的每个节点分配一个权重。这个权重是一个整数值，它被算入任何重量感知调度算法 （如加权最小连接）中，并帮助 LVS 路由器更加均匀地加载具有不同能力的硬件。

权重按照相对于相互的比率来工作。例如，如果一个实际服务器具有 1 的权重，而另一服务器权重为 5，那么权重为 5 的服务器会为另一服务器获得的每 1 连接获得 5 连接。实际服务器权重的默认值为 1。

虽然为实际服务器池中的各种硬件配置添加权重可以帮助更有效地对群集进行负载平衡，但在将实际服务器引入实际服务器池时，并且使用加权最小连接调度虚拟服务器时，可能会导致临时问题。例如，假设实际的服务器池中有三个服务器：服务器 A 和 B 的权重为 1，第三台服务器 C 的权重为 2。如果服务器 C 因任何原因而停机，服务器 A 和 B 平均分配负载。但是，一旦服务器 C 重新上线，LVS 路由器看到它的连接为零，并填充了服务器的所有传入请求，直到它与服务器 A 和 B 相同。

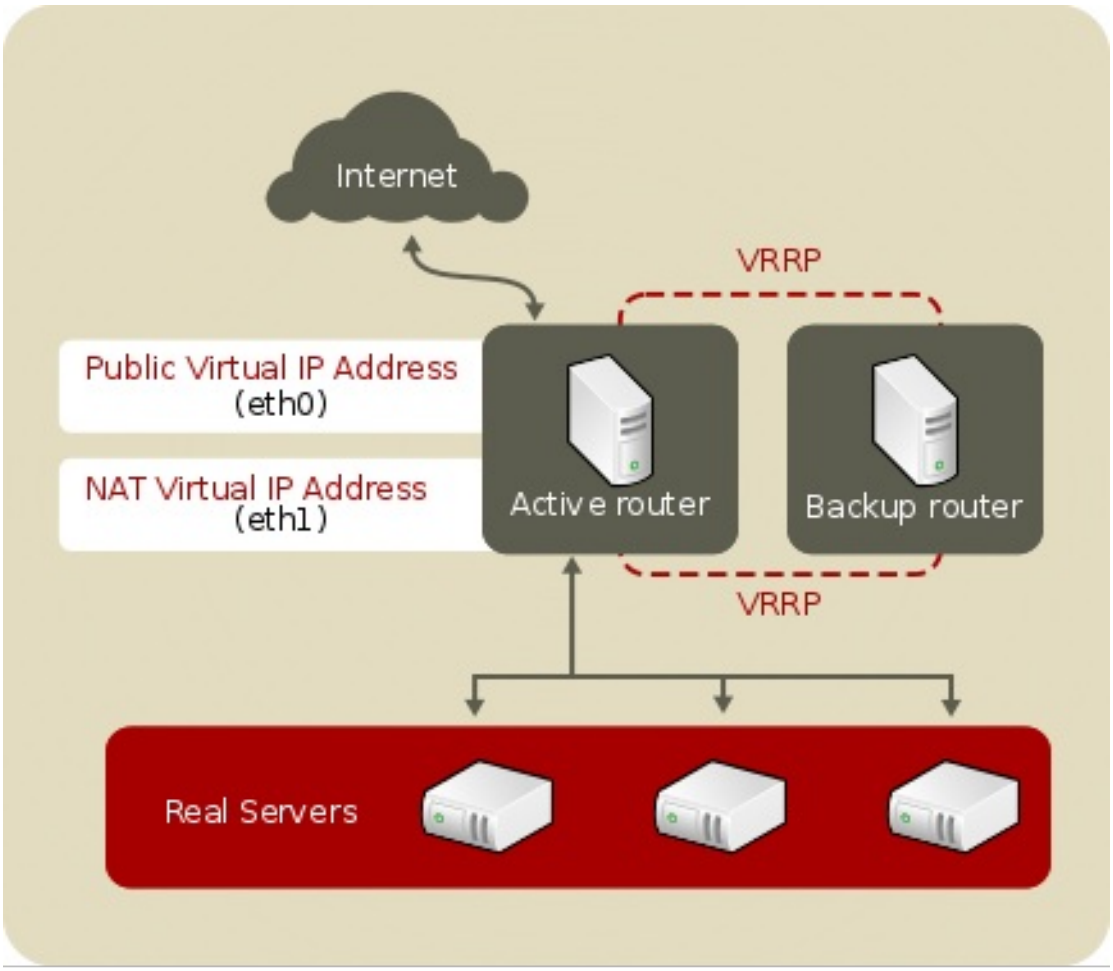
2.4. 路由方法

红帽企业 Linux 使用网络地址转换（NAT 路由）或Keepalived 的直接路由。这样，管理员可以在利用现有硬件时提供巨大的灵活性，并将负载平衡器集成到现有网络中。

2.4.1. NAT 路由

图 2.3 “通过 NAT 路由实施负载均衡器”，演示了利用 NAT 路由在互联网和专用网络之间移动请求的负载平衡器。

图 2.3. 通过 NAT 路由实施负载均衡器



[D]

在该示例中，活动 LVS 路由器中有两个 NIC。互联网的 NIC 在 eth0 上具有实际的 IP 地址和浮动 IP 地址。专用网络接口的 NIC 在 eth1 上具有实际的 IP 地址和浮动 IP 地址。在故障转移时，面向互联网和面向虚拟接口的虚拟接口同时被备份 LVS 路由器接管。位于专用网络中的所有实际服务器都将 NAT 路由器的浮动 IP 用作其默认路由以与活动 LVS 路由器通信，以便它们响应来自互联网的请求的能力不会受到影响。

在本例中，LVS 路由器的公共浮动 IP 地址和专用 NAT 浮动 IP 地址分配到物理 NIC。虽然可以将每个浮动 IP 地址关联到 LVS 路由器节点上自己的物理设备，但并不要求拥有超过两个 NIC。

使用此拓扑时，活动 LVS 路由器接收请求并将其路由到适当的服务器。然后，实际服务器处理请求并将数据包返回到 LVS 路由器，该路由器使用网络地址转换将数据包中实际服务器的地址替换为 LVS 路由器的公共 VIP 地址。此过程称为 IP 伪装，因为实际服务器的实际 IP 地址在请求的客户端中隐藏。

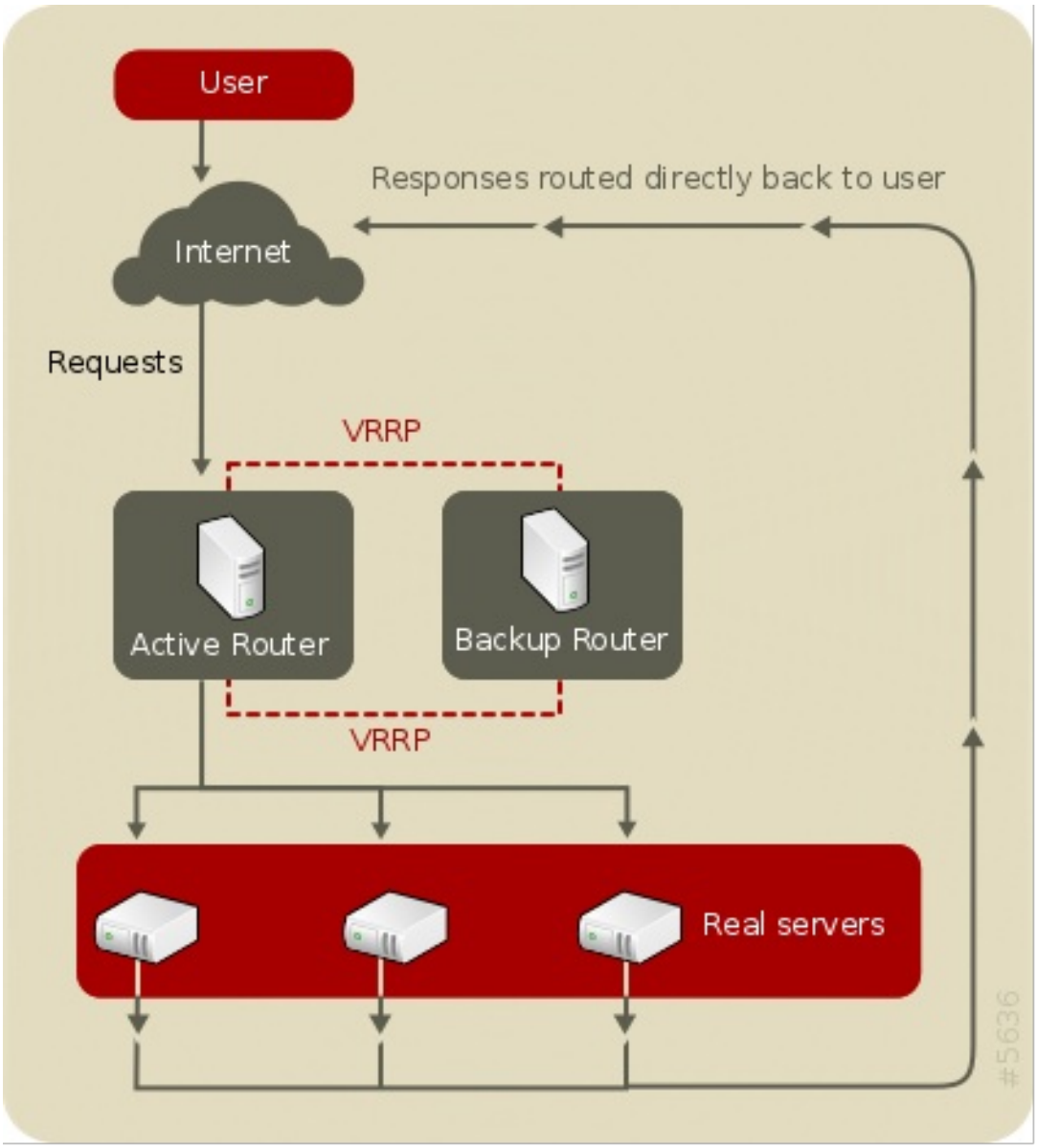
使用此 NAT 路由时，实际服务器可能是运行各种操作系统的任何机器。最大的缺点是 LVS 路由器可能会成为大型集群部署中的瓶颈，因为它必须处理传出和传入的请求。

***ipvs** 模块使用独立于 iptables 和 ip6tables NAT 自己的内部 NAT 例程。当为 NAT 配置真实服务器而不是 **/etc/keepalived/keepalived.conf** 文件中的 DR 时，这将有助于 IPv4 和 IPv6 NAT。*

2.4.2. 直接路由

构建使用直接路由的负载平衡器设置可提高其他负载平衡器网络拓扑的性能优势。直接路由允许实际服务器直接处理数据包并路由到请求的用户，而不是通过 LVS 路由器传递所有传出数据包。直接路由通过委派 LVS 路由器的任务仅处理传入的数据包，从而减少了网络性能问题的可能性。

图 2.4. 使用直接路由实施负载均衡器



[D]

在典型的直接路由负载平衡器设置中，LVS 路由器通过虚拟 IP(VIP)接收传入服务器请求，并使用调度算法将请求路由到实际服务器。实际的服务器处理请求并将响应直接发送到客户端，绕过 LVS 路由器。这种路由方法实现了可扩展性，这样可以添加实际服务器，而 LVS 路由器不会增加负担，将来自实际服务器传出的数据包路由到客户端，这可能会成为繁重网络负载下的瓶颈。

2.4.2.1. 直接路由和 ARP 限制

虽然在负载平衡器中使用直接路由有很多优点，但也存在限制。负载平衡器通过直接路由最常见的问题在于地址解析协议(ARP)。

在典型的情形中，互联网上的客户端发送请求到 IP 地址。网络路由器通常通过相关的 IP 地址将请求发送到其目的地，以使用 ARP 将计算机的 MAC 地址发送到其目的地。ARP 请求广播到网络上的所有连接的计算机，具有正确 IP/MAC 地址组合的计算机将接收数据包。IP/MAC 关联存储在 ARP 缓存中，该缓存定期清除（通常每 15 分钟一次）并重新填充 IP/MAC 关联。

直接路由负载平衡器设置中的 ARP 请求的问题在于，因为对 IP 地址的客户端请求必须与要处理的请求的 MAC 地址关联，负载平衡器系统的虚拟 IP 地址也必须与 MAC 关联。但是，由于 LVS 路由器和实际服务器都具有相同的 VIP，因此 ARP 请求将广播到与 VIP 关联的所有计算机。这可能导致几个问题，例如 VIP

直接与其中一个实际服务器关联并直接处理请求，完全绕过 LVS 路由器并消除负载平衡器设置的目的。

要解决这个问题，请确保传入的请求始终发送到 LVS 路由器，而不是实际的服务器之一。这可以通过过滤 ARP 请求或过滤 IP 数据包来完成。可以使用 **arptables** 实用程序和 IP 数据包过滤 ARP 过滤，可使用 **iptables** 或 **firewalld** 进行过滤。这两种方法的不同之处如下：

- ARP 过滤方法阻止到达实际服务器的请求。这可以防止 ARP 将 VIP 与实际服务器关联，让活动虚拟服务器能够响应 MAC 地址。
- IP 数据包过滤方法允许将数据包路由到与其他 IP 地址相关的实际服务器。这完全消除了 ARP 问题，最初未在实际服务器上配置 VIP。

2.5. KEEPALIVED 的持久性和防火墙标记

在某些情况下，客户端可能需要重复地重新连接同一实际服务器，而不是让负载平衡算法将该请求发送到最佳可用服务器。此类情况的示例包括多屏幕 Web 表单、cookies、SSL 和 FTP 连接。在这些情况下，除非事务由同一服务器处理来保留上下文，否则客户端可能无法正常工作。keepalived 提供了两种不同的功能来解决这个问题：持久性和 防火墙标记。

2.5.1. 持久性

启用后，持久性类似于计时器。当客户端连接到服务时，Load Balancer 会记住指定时间段内的最后连接。如果同一客户端 IP 地址在该期间内再次连接，它将发送到它之前连接到的同一服务器，绕过负载平衡机制。当连接在时间窗之外时，它会根据调度规则进行处理。

借助持久性，管理员可以指定子网掩码以应用到客户端 IP 地址测试，作为控制哪些地址具有更高持久性的工具，从而对该子网的连接进行分组。

对于使用多个端口进行通信的协议（如 FTP）而言，为不同端口对连接进行分组可能很重要。但是，持久性不是解决针对不同端口将连接分组的问题的最有效方式。对于这些情况，最好使用防火墙标记。

2.5.2. firewall Marks

防火墙标记是一种对用于协议或相关协议的端口进行分组的简单而高效的方式。例如，如果部署了 Load Balancer 运行电子商务站点，可使用防火墙标记在端口 80 上以及端口 443 上安全的 HTTPS 连接捆绑 HTTP 连接。通过为每个协议分配相同的防火墙标记，可以保留事务的状态信息，因为 LVS 路由器将在连接打开后将所有请求转发到同一实际服务器。

由于其效率和易用性，负载平衡器的管理员应尽可能使用防火墙标记而不是持久性来分组连接。但是，管理员仍应结合防火墙标记为虚拟服务器添加持久性，以确保客户端在适当的时间段内与同一服务器重新连接。

第 3 章 为 KEEPALIVED 设置负载均衡器先决条件

使用 **keepalived** 的 Load Balancer 由两个基本组组成：LVS 路由器和真实服务器。为防止单点故障，每个组应至少有两个成员。

LVS 路由器组应当包含两个运行 Red Hat Enterprise Linux 的相同或非常类似的系统。个用作活动的 LVS 路由器，而另一个则保持热备用模式，因此它们需要尽可能接近相同的功能。

在为实际服务器组选择和配置硬件之前，请确定要使用的三种负载平衡器拓扑中的哪一种。

3.1. NAT 负载平衡器网络

NAT 拓扑允许在使用现有硬件时有很大的灵活性，但处理大型负载的能力有限，因为所有数据包都通过负载平衡器路由器进入和传出池。

网络布局

从网络布局视角配置使用 NAT 路由的 Load Balancer 拓扑最为简便，因为只需要一个访问点即可访问公共网络。实际的服务器位于专用网络上，并通过 LVS 路由器响应所有请求。

硬件

在 NAT 拓扑中，每个实际服务器只需要一个 NIC，因为它将仅响应 LVS 路由器。另一方面，LVS 路由器需要两个 NIC 来路由两个网络之间的流量。由于此拓扑在 LVS 路由器上创建网络瓶颈，因此可在每个 LVS 路由器上利用千兆位以太网 NIC 来增加 LVS 路由器可以处理的带宽。如果 LVS 路由器上使用了千兆位以太网，任何将实际服务器连接到 LVS 路由器的交换机都必须至少具有两个千兆位以太网端口才能有效地处理负载。

软件

由于 NAT 拓扑需要将 **iptables** 用于某些配置，所以 Keepalived 之外可能存在大量软件配置。特别是，FTP 服务和使用防火墙标记需要额外手动配置 LVS 路由器以正确路由请求。

3.1.1. 使用 NAT 为负载平衡器配置网络接口

要使用 NAT 设置负载平衡器，您必须首先为公共网络和 LVS 路由器上的专用网络配置网络接口。在本例中，LVS 路由器的公共接口(**eth0**)将位于 203.0.113.0/24 网络上，并且链接到实际服务器(**eth**1)的私有接口将位于 10.11.12.0/24 网络中。

重要

编写本文时，**NetworkManager** 服务与 Load Balancer 不兼容。特别是，当 SLAAC 分配了 IPv6 地址时，IPv6 VIP 将无法正常工作。因此，这里显示的示例使用配置文件和 **网络服务**。

在 active 或 primary LVS 路由器节点上，公共接口的网络配置文件 **/etc/sysconfig/network-scripts/ifcfg-eth0** 类似如下：

```
DEVICE=eth0
BOOTPROTO=static
ONBOOT=yes
IPADDR=203.0.113.9
NETMASK=255.255.255.0
GATEWAY=203.0.113.254
```

LVS 路由器上的专用 NAT 接口的配置文件 **/etc/sysconfig/network-scripts/ifcfg-eth1** 类似如下：

```
DEVICE=eth1
BOOTPROTO=static
ONBOOT=yes
IPADDR=10.11.12.9
NETMASK=255.255.255.0
```

VIP 地址必须与静态地址不同，但在同一范围内。在本例中，LVS 路由器的公共接口的 VIP 可以配置为 203.0.113.10，私有接口的 VIP 可以是 10.11.12.10。VIP 地址由 **/etc/keepalived/keepalived.conf** 文件中的 **virtual_ipaddress** 选项设置。如需更多信息，请参阅 第 4.1 节 “基本 Keepalived 配置”。

另外，请确保真实服务器将请求路由回 NAT 接口的 VIP。

重要

本节中的示例以太网接口配置设置用于 LVS 路由器的实际 IP 地址，而非浮动 IP 地址。

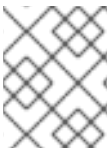
在配置主 LVS 路由器节点的网络接口后，配置备份 LVS 路由器的实际网络接口（请注意，IP 地址都不与网络上的任何其他 IP 地址冲突）。

重要

确保备份节点上的每个接口服务与主节点上的接口相同的网络。例如，如果 eth0 连接到主节点上的公共网络，它还必须连接到备份节点上的公共网络。

3.1.2. 实时服务器上的路由

在 NAT 拓扑中配置实际服务器网络接口时，要记住最重要的事项是为 LVS 路由器的 NAT 浮动 IP 地址设置网关。在本例中，该地址为 10.11.12.10。



注意

网络接口在实际服务器上启动后，计算机将无法以其他方式 ping 或连接到公共网络。这是正常的。但是，您将能够 ping LVS 路由器专用接口的实际 IP，本例中为 10.11.12.9。

实际服务器的配置文件 **/etc/sysconfig/network-scripts/ifcfg-eth0** 文件可能类似如下：

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=10.11.12.1
NETMASK=255.255.255.0
GATEWAY=10.11.12.10
```

警告

如果一个真实服务器配置了 **GATEWAY=** 行，则第一个网络接口将获得网关。因此，如果同时配置了 **eth0** 和 **eth1**，并且 **eth1** 用于 Load Balancer，则实际服务器可能无法正确路由请求。

最好在 **/etc/sysconfig/network-scripts/** 目录中设置 **ONBOOT=no** 来关闭额外的网络接口，或者确保在接口中正确设置了网关。

3.1.3. 在 LVS 路由器中启用 NAT 路由

在简单的 NAT 负载平衡器配置中，每个集群服务仅使用一个端口，如 HTTP 在端口 80 上，管理员只需要启用 LVS 路由器上的数据包转发，以便在外部世界和实际服务器之间正确路由请求。但是，当集群服务需要多个端口在用户会话期间进入同一实际服务器时，需要进行更多的配置。

在 LVS 路由器上启用转发并且设置了实际服务器并设置了集群服务后，请使用 **keepalived** 配置 IP 信息。

警告

不要手动编辑网络配置文件或使用网络配置工具，为 **eth0** 或 **eth1** 配置浮动 IP。反之，使用 **keepalived.conf** 文件配置它们。

完成后，启动 **keepalived** 服务。启动并运行后，活动 LVS 路由器将开始将请求路由到实际服务器的池。

3.2. 使用直接路由的负载均衡器

直接路由允许实际服务器直接处理数据包并路由到请求的用户，而不是通过 LVS 路由器传递传出数据包。直接路由要求实际服务器通过 LVS 路由器物理连接到网络段，并且能够处理和定向传出数据包。

网络布局

在直接路由负载平衡器设置中，LVS 路由器需要接收传入的请求，并将它们路由到正确的实际服务器以进行处理。然后，实际服务器需要将响应直接路由到客户端。因此，例如，如果客户端位于 Internet 上，并且通过 LVS 路由器将数据包发送到实际服务器，则实际服务器必须能够通过互联网直接连接到客户端。这可以通过配置实际服务器的网关以将数据包传递到互联网来完成。服务器池中的每一实际服务器都可拥有自己的独立网关（以及具有自己连接到互联网的每个网关），实现最大吞吐量和可扩展性。但是，对于典型的负载平衡器设置，实际服务器可以通过一个网关（因此也有一个网络连接）进行通信。

硬件

使用直接路由的负载平衡器系统的硬件要求与其他负载平衡器拓扑类似。虽然 LVS 路由器需要运行红帽企业 Linux，以处理传入请求并为真实服务器执行负载平衡，但实际服务器不需要是 Linux 机器才能正常工作。LVS 路由器需要一个或两个 NIC（取决于是否有备份路由器）。您可以使用两个 NIC 轻松配置并区分不同的流量；传入的请求由一个 NIC 处理，并将数据包路由到另一 NIC 上的实际服务器。

由于实际服务器绕过 LVS 路由器并将传出数据包直接发送到客户端，因此需要网关到互联网。为了获得最佳性能和可用性，每个实际服务器可以连接到其自身的独立网关，这些网关拥有其自身专用连接到客户端的网络（如互联网或 Intranet）的专用连接。

软件

keepalived 之外有一些配置需要完成，特别是管理员通过直接路由使用 Load Balancer 时遇到 ARP 问题。如需更多信息，请参阅 第 3.2.1 节 "使用 arptables 直接路由" 或 第 3.2.3 节 "使用 iptables 直接路由"。

3.2.1. 使用 arptables 直接路由

要使用 **arptables** 配置直接路由，每个真实服务器都必须配置其虚拟 IP 地址，以便可以直接路由数据包。对 VIP 的 ARP 请求会被实际服务器完全忽略，否则可能发送的任何 ARP 数据包都会被操作，以包含实际服务器的 IP 而非 VIP。

使用 **arptables** 方法，应用程序可能会绑定到真实服务器提供服务的每个 VIP 或端口。例如，**arptables** 方法允许运行多个 Apache HTTP 服务器实例，并明确绑定到系统上的不同 VIP。

但是，使用 **arptables** 方法时，无法将 VIP 配置为使用标准 Red Hat Enterprise Linux 系统配置工具在引导时启动。

要将每个实际服务器配置为忽略每个虚拟 IP 地址的 ARP 请求，请执行以下步骤：

1. 在每个真实服务器上为每个虚拟 IP 地址创建 ARP 表条目(real_ip 是 director 用于与实际服务器通信的 IP ；通常是绑定到 **eth0** 的 IP)：

   ```
   arptables -A IN -d <virtual_ip> -j DROP
   arptables -A OUT -s <virtual_ip> -j mangle --mangle-ip-s <real_ip>
   ```

   这将导致真实服务器忽略虚拟 IP 地址的所有 ARP 请求，并更改任何可能包含虚拟 IP 的传出 ARP 响应，以便它们改为包含服务器的实际 IP。唯一应响应任何 VIP 的 ARP 请求的节点是当前活动的 LVS 节点。

2. 且在每个真实服务器上完成此操作，请在每个真实服务器上键入以下命令来保存 ARP 表条目：

   arptables-save > /etc/sysconfig/arptables

   systemctl enable arptables.service

   systemctl enable 命令将在启动网络前在启动时重新载入 arptables 配置。

3. 使用 **ip addr** 在所有真实服务器上配置虚拟 IP 地址，以创建 IP 别名。例如：

   ```
   # ip addr add 192.168.76.24 dev eth0
   ```

4. 为直接路由配置 Keepalived。这可以通过在 **keepalived.conf** 文件中添加 **lb_kind DR** 来实现。如需更多信息，请参阅 第 4 章 使用 Keepalived 初始负载均衡器配置。

3.2.2. 使用 firewalld 直接路由

您还可以通过使用 **firewalld** 创建防火墙规则，使用直接路由方法解决 ARP 问题。要使用 **firewalld** 配置直接路由，您必须添加创建透明代理的规则，以便实际服务器将服务发送到 VIP 地址的数据包，即使系统上不存在 VIP 地址。

firewalld 方法比 **arptables** 方法配置更简单。此方法也完全规避 LVS ARP 问题，因为虚拟 IP 地址或地址仅存在于活动的 LVS director 上。

但是，与 **arptables** 相比，使用 **firewalld** 方法有一些性能问题，因为每个返回数据包都有开销。

您也不能使用 **firewalld** 方法重复使用端口。例如：无法运行绑定到端口 80 的两个独立的 Apache HTTP 服务器服务，因为两者都必须绑定到 **INADDR_ANY** 而不是虚拟 IP 地址。

要使用 **firewalld** 方法配置直接路由，请在每个真实服务器上执行以下步骤：

1. 确保 **firewalld** 正在运行。

   ```
   # systemctl start firewalld
   ```

   确保 **firewalld** 已启用在系统启动时启动。

   ```
   # systemctl enable firewalld
   ```

2. 为每个要为真实服务器提供服务的 VIP、端口和协议（TCP 或 UDP）组合输入以下命令。此命令将导致真实服务器处理以 VIP 和端口指定的数据包。

   ```
   # firewall-cmd --permanent --direct --add-rule ipv4 nat PREROUTING 0 -d vip -p tcp|udp -m tcp|udp --dport port -j REDIRECT
   ```

3. 重新加载防火墙规则，并保留状态信息。

   ```
   # firewall-cmd --reload
   ```

   当前永久配置将成为新的 **firewalld** 运行时配置以及下一次系统启动时的配置。

3.2.3. 使用 iptables 直接路由

您还可以通过创建 **iptables** 防火墙规则，使用直接路由方法解决 ARP 问题。要使用 **iptables** 配置直接路由，您必须添加创建透明代理的规则，以便实际服务器将服务发送到 VIP 地址的数据包，即使系统上不存在 VIP 地址。

iptables 方法的配置比 **arptables** 方法更为简单。此方法也完全规避 LVS ARP 问题，因为虚拟 IP 地址仅存在于活动的 LVS director 上。

但是，与 **arptables** 相比，使用 **iptables** 方法有一些性能问题，因为转发/伪装每个数据包存在开销。

您也不能使用 **iptables** 方法重复使用端口。例如：无法运行绑定到端口 80 的两个独立的 Apache HTTP 服务器服务，因为两者都必须绑定到 **INADDR_ANY** 而不是虚拟 IP 地址。

要使用 **iptables** 方法配置直接路由，请执行以下步骤：

1. 在每个真实服务器中，为旨在为真实服务器提供服务的每个 VIP、端口和协议（TCP 或 UDP）组合输入以下命令：

   iptables -t nat -A PREROUTING -p <tcp|udp> -d <vip> --dport <port> -j REDIRECT

   此命令将导致真实服务器处理以 VIP 和端口指定的数据包。

2. 保存每个真实服务器中的配置：

   -

```
# iptables-save > /etc/sysconfig/iptables
# systemctl enable iptables.service
```

systemctl enable 命令将在启动网络前在启动时重新载入 iptables 配置。

3.2.4. 使用 sysctl 直接路由

使用直接路由时处理 ARP 限制的另一种方法是使用 **sysctl** 接口。管理员可以配置两个 **systcl** 设置，使得真实服务器不会在 ARP 请求中宣布 VIP，且不会回复 VIP 地址的 ARP 请求。要启用此功能，请输入以下命令：

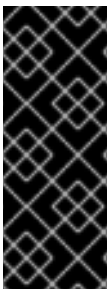
```
echo 1 > /proc/sys/net/ipv4/conf/eth0/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/eth0/arp_announce
```

或者，您可以在 **/etc/sysctl.d/arp.conf** 文件中添加以下行：

```
net.ipv4.conf.eth0.arp_ignore = 1
net.ipv4.conf.eth0.arp_announce = 2
```

3.3. 将配置放在一起

在确定了要使用的上述路由方法后，硬件应连接并配置。



重要

LVS 路由器上的网络适配器必须配置为访问相同的网络。例如，如果 **eth0** 连接到公共网络，并且 **eth1** 连接到专用网络，那么备份 LVS 路由器上的这些相同设备必须连接到同一网络。

另外，第一个接口中列出的网关也会添加到路由表中，其他接口中列出的后续网关将被忽略。在配置真实服务器时，这一点尤为重要。

将硬件连接到网络后，在主路由器和 backup LVS 路由器上配置网络接口。这应当通过手动编辑网络配置文件来完成。有关使用网络配置文件的更多信息，请参阅 Red Hat Enterprise Linux 7 网络指南 。

3.3.1. 常规负载均衡器网络提示

在尝试使用 Keepalived 配置负载平衡器之前，为 LVS 路由器上的公共和专用网络配置实际 IP 地址。每个拓扑中的部分都给出了示例网络地址，但实际的网络地址是必需的。以下是用于启动网络接口或检查其状态的一些有用命令。

启动实时网络接口

要打开实际网络接口，请以 **root** 身份使用以下命令，将 N 替换为与接口对应的数字(**eth0** 和 **eth1**)。

ifup ethN

警告

不要使用 **ifup** 脚本打开任何您可以使用 Keepalived（**eth0:1** 或 **eth1:1**）配置的浮动 IP 地址。使用 **service** 或 **systemctl** 命令启动 **keepalived**。

进行关闭实时网络接口

若要关闭实际网络接口，请以 **root** 身份使用以下命令，将 N 替换为与接口对应的数字（**eth0** 和 **eth1**）。

ifdown ethN

检查网络接口的状态

如果您需要检查在任意给定时间启动哪些网络接口，请输入以下命令：

IP 链接

要查看机器的路由表，请运行以下命令：

IP 路由

3.3.2. 防火墙要求

如果您正在运行防火墙（通过 **firewalld** 或 **iptables**），您必须允许 VRRP 流量在 **keepalived** 节点之间传递。要将防火墙配置为允许使用 **firewalld** 的 VRRP 流量，请运行以下命令：

```
# firewall-cmd --add-rich-rule='rule protocol value="vrrp" accept' --permanent
# firewall-cmd --reload
```

如果省略 区域，则将使用默认区域。

但是，如果您需要允许带有 **iptables** 的 VRRP 流量，请运行以下命令：

```
# iptables -I INPUT -p vrrp -j ACCEPT
# iptables-save > /etc/sysconfig/iptables
# systemctl restart iptables
```

3.4. 多端口服务和负载平衡器

在创建多端口负载平衡器服务时，任何拓扑下的 LVS 路由器都需要额外的配置。通过使用防火墙标记将不同但相关的协议（如 HTTP（端口 80）和 HTTPS（端口 443））或 Load Balancer 与真正的多端口协议一起使用时，多端口服务可以动态创建。在这两种情况下，LVS 路由器使用防火墙标记来识别应使用相同的防火墙标记，以识别针对不同端口指定的数据包，但符合相同的防火墙标记。此外，当与持久性结合使用时，防火墙标记可确保从客户端计算机的连接路由到同一主机，只要连接在 persistence 参数指定的时段内即可。

虽然用于平衡实际服务器上的负载的机制可以识别分配给包的防火墙标记，但它本身无法分配防火墙标记。分配 防火墙标记的作业必须由网络数据包过滤 **iptables** 执行。Red Hat Enterprise Linux 7 中的默认

防火墙管理工具是 **firewalld**，可用于配置 **iptables**。如果首选，可以直接使用 **iptables**。有关在 Red Hat Enterprise Linux 7 中使用 **iptables** 的详情，请查看 Red Hat Enterprise Linux 7 安全指南 。

3.4.1. 使用 firewalld 分配防火墙标记

要将防火墙标记分配给指定特定端口的数据包，管理员可以使用 **firewalld** 的**firewall-cmd** 工具。

如果需要，确认 **firewalld** 正在运行：

```
# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled)
   Active: active (running) since Tue 2016-01-26 05:23:53 EST; 7h ago
```

要启动 **firewalld**，请输入：

```
# systemctl start firewalld
```

确保 **firewalld** 在系统启动时启动：

```
# systemctl enable firewalld
```

本节演示了如何将 HTTP 和 HTTPS 捆绑为示例；但是，FTP 是另一种常见的多端口协议。

使用防火墙标记时要记住的基本规则是，对于在 Keepalived 中使用防火墙标记的每个协议，都必须有一条防火墙规则来为网络数据包分配标记。

在创建网络数据包过滤规则之前，请确保尚未有任何规则。要做到这一点，打开 shell 提示符，以 **root** 身份登录，然后输入以下命令：

```
# firewall-cmd --list-rich-rules
```

如果没有富规则，则提示符将立即重新显示。

如果 **firewalld** 是活跃的且丰富的规则，它会显示一组规则。

如果现有规则非常重要，请检查 **/etc/firewalld/zones/** 的内容，并在继续前将任何需要保留的规则复制到安全位置。使用以下格式的命令删除不需要的富规则：

```
firewall-cmd --zone=zone --remove-rich-rule='rule' --permanent
```

--permanent 选项使设置持久，但命令只会在下次系统启动时生效。如果需要使设置立即生效，请重复省略 **--permanent** 选项的命令。

要配置的第一个负载均衡器相关的防火墙规则是允许 Keepalived 服务的 VRRP 流量正常工作。使用以下命令：

```
# firewall-cmd --add-rich-rule='rule protocol value="vrrp" accept' --permanent
```

如果省略 区域，则将使用默认区域。

以下是将同一防火墙标记 **80** 分配给用于浮动 IP 地址 n.n.n.n 的规则（端口 80 和 443）。

```
# firewall-cmd --add-rich-rule='rule family="ipv4" destination address="n.n.n.n/32" port port="80"
```

```
protocol="tcp" mark set="80"' --permanent
# firewall-cmd --add-rich-rule='rule family="ipv4" destination address="n.n.n.n/32" port port="443"
protocol="tcp" mark set="80"' --permanent
# firewall-cmd --reload
success
# firewall-cmd --list-rich-rules
rule protocol value="vrrp" accept
rule family="ipv4" destination address="n.n.n.n/32" port port="80" protocol="tcp" mark set=80
rule family="ipv4" destination address="n.n.n.n/32" port port="443" protocol="tcp" mark set=80
```

如果省略 区域，则将使用默认区域。

有关使用 **firewalld** 丰富的语言命令的更多信息，请参阅 Red Hat Enterprise Linux 7 安全指南 。

3.4.2. 使用 iptables 分配防火墙标记

要将防火墙标记分配给指定特定端口的数据包，管理员可以使用 **iptables**。

本节演示了如何将 HTTP 和 HTTPS 捆绑为示例；但是，FTP 是另一种常见的多端口协议。

使用防火墙标记时要记住的基本规则是，对于在 Keepalived 中使用防火墙标记的每个协议，都必须有一条防火墙规则来为网络数据包分配标记。

在创建网络数据包过滤规则之前，请确保尚未有任何规则。要做到这一点，打开 shell 提示符，以 **root** 身份登录，然后输入以下命令：

/usr/sbin/service iptables status

如果 **iptables** 没有运行，则提示符将立即重新应用。

如果 **iptables** 处于活动状态，它将显示一组规则。如果存在规则，请输入以下命令：

/sbin/service iptables stop

如果规则已经就位很重要，请检查 **/etc/sysconfig/iptables** 的内容，并在继续操作前将任何需要保留的规则复制到安全位置。

第一个与配置防火墙规则相关的负载均衡器是允许 Keepalived 服务的 VRRP 流量正常工作。

```
/usr/sbin/iptables -I INPUT -p vrrp -j ACCEPT
```

以下是将同一防火墙标记 **80** 分配给用于浮动 IP 地址 n.n.n.n 的规则（端口 80 和 443）。

```
/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n.n/32 -m multiport --dports 80,443 -j
MARK --set-mark 80
```

请注意，您必须以 **root** 身份登录并加载 **iptables** 模块，然后才能首次发出规则。

在上面的 **iptables** 命令中，n.n.n.n.n 应替换为 HTTP 和 HTTPS 虚拟服务器的浮动 IP。这些命令在适当的端口上为防火墙标记 80 分配指向 VIP 的任何通信，进而得到 IPVS 识别并相应地转发。

警告

上述命令将立即生效，但不会在系统重启后保留。

3.5. 配置 FTP

文件传输协议(FTP)是一款老旧且复杂的多端口协议，对负载平衡器环境构成了一系列截然不同的挑战。要了解这些挑战的性质，您必须首先了解 FTP 如何工作的一些关键事项。

3.5.1. FTP 的工作原理

对于大多数其他服务器客户端关系，客户端计算机会打开与特定端口上的服务器的连接，然后服务器在该端口上响应客户端。FTP 客户端连接到 FTP 服务器时，它会打开与 FTP 控制端口 21 的连接。然后，客户端告知 FTP 服务器是建立 主动连接还是 被动连接。客户端选择的连接类型决定了服务器如何响应以及将发生哪些端口事务。

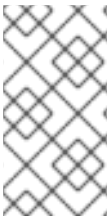
两种类型的数据连接是：

活跃连接

建立活动连接时，服务器会打开与客户端的数据连接，从端口 20 到客户端计算机上的高范围端口。然后，服务器中的所有数据都会通过此连接传递。

被动连接

建立被动连接时，客户端会询问 FTP 服务器建立被动连接端口，该端口可以位于超过 10,000 的端口上。服务器随后为此特定会话绑定到此高编号的端口，并将该端口号转发回客户端。然后，客户端打开用于数据连接的新绑定端口。客户端发出的每一项数据请求都会产生独立的数据连接。大多数现代 FTP 客户端在从服务器请求数据时尝试建立被动连接。



注意

客户端决定连接的类型，而非服务器。这意味着有效集群 FTP，您必须配置 LVS 路由器以处理主动和被动连接。

FTP 客户端-服务器关系可能会打开 Keepalived 的大量端口。

3.5.2. 这如何影响负载平衡器路由

IPVS 数据包转发仅允许基于该集群的端口号或其防火墙标记来回连接。如果来自集群外部的客户端试图打开一个端口 IPVS 未配置为处理，它会丢弃连接。同样，如果实际服务器尝试在端口 IPVS 上打开回互联网的连接，它会丢弃连接。这意味着，来自 Internet 上的 FTP 客户端的所有连接 都必须分配有相同的防火墙标记，并且所有来自 FTP 服务器的连接都必须使用网络包过滤规则正确转发到 Internet。

注意

要启用被动 FTP 连接，您必须加载 **ip_vs_ftp** 内核模块。在 shell 提示符下以管理用户身份运行以下命令来载入此模块，并确保模块在重启时载入：

```
echo "ip_vs_ftp" >> /etc/modules-load.d/ip_vs_ftp.conf
systemctl enable systemd-modules-load
systemctl start systemd-modules-load
```

3.5.3. 创建网络数据包过滤规则

在为 FTP 服务分配任何 **iptables** 规则前，请查看 第 3.4 节 "多端口服务和负载平衡器" 中有关多端口服务和用于检查现有网络数据包过滤规则的技术中的信息。

以下是将同一防火墙标记 **21** 分配给 FTP 流量的规则。

3.5.3.1. Active Connections 规则

活跃连接的规则告诉内核接受并转发到端口 **20** (FTP 数据端口)上 内部 浮动 IP 地址的连接。

以下 **iptables** 命令允许 LVS 路由器接受来自 IPVS 不知道的实际服务器的传出连接：

```
/usr/sbin/iptables -t nat -A POSTROUTING -p tcp -s n.n.n.0/24 --sport 20 -j MASQUERADE
```

在 **iptables** 命令中，n.n.n 应替换为 **keepalived.conf** 文件定义了 **virtual_server** 部分的 NAT 接口内部网络接口的前三个值。

3.5.3.2. 被动连接规则

被动连接的规则为从互联网到服务浮动 IP 地址的连接分配适当的防火墙标记：10,000 到 20,000。

警告

如果要限制被动连接的端口范围，还必须配置 FTP 服务器 **vsftpd** 以使用匹配的端口范围。这可以通过在 **/etc/vsftpd.conf** 中添加以下行来实现：

```
pasv_min_port=10000

pasv_max_port=20000
```

不应使用将 pasv_address 设置为覆盖实际 FTP 服务器地址，因为它已更新为 LVS 的虚拟 IP 地址。

有关其他 FTP 服务器的配置，请参阅相应的文档。

这个范围应该足以满足大多数情况的影响。但是，您可以通过在以下命令将 **10000:20000** 改为 **1024:65535**，将这个数字增加到包含所有可用的非安全端口。

以下 **iptables** 命令效果为为相应端口上浮动 IP 地址分配任何流量，其防火墙标记为 **21**，后者由 IPVS 识别并适当转发：

/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n/32 --dport 21 -j MARK --set-mark 21

/usr/sbin/iptables -t mangle -A PREROUTING -p tcp -d n.n.n/32 --dport 10000:20000 -j MARK --set-mark 21

在 **iptables** 命令中，n.n.n.n.n 应替换为 **keepalived.conf** 文件的 **virtual_server** 子部分中定义的 FTP 虚拟服务器的浮动 IP。

上述命令将立即生效，但不会在系统重启后继续生效，除非已保存。要保存更改，请输入以下命令：

```
# iptables-save > /etc/sysconfig/iptables
```

要确保 **iptables** 服务在系统启动时启动，请输入以下命令：

```
# systemctl enable iptables
```

您可以运行以下命令来验证更改是否在重启后保留，并检查更改是否仍然存在：

```
# systemctl restart iptables
```

3.6. 保存网络数据包过滤器设置

为您的情况配置适当的网络数据包过滤器后，保存设置，以便在重新启动后恢复它们。对于 **iptables**，输入以下命令：

```
# iptables-save > /etc/sysconfig/iptables
```

要确保 **iptables** 服务在系统启动时启动，请输入以下命令：

```
# systemctl enable iptables
```

您可以运行以下命令来验证更改是否在重启后保留，并检查更改是否仍然存在：

```
# systemctl restart iptables
```

有关在 Red Hat Enterprise Linux 7 中使用 **iptables** 的详情，请查看 Red Hat Enterprise Linux 7 安全指南

3.7. 打开 PACKET 转发和非本地绑定

为了使 Keepalived 服务能够将网络数据包正确转发到实际服务器，每个路由器节点必须在内核中开启 IP 转发。以 **root** 身份登录，并将 **/etc/sysctl.conf** 中读取 **net.ipv4.ip_forward = 0** 的行改为以下内容：

```
net.ipv4.ip_forward = 1
```

更改在重启系统时生效。

HAProxy 和 Keepalived 中的负载均衡还需要能够绑定到非本地的 IP 地址，这意味着它不会被分配给本地系统上的设备。这允许正在运行的负载均衡器实例绑定到不是本地的用于故障转移的 IP。

要启用，请编辑 **/etc/sysctl.conf** 中读取 **net.ipv4.ip_nonlocal_bind** 的行：

```
net.ipv4.ip_nonlocal_bind = 1
```

更改在重启系统时生效。

*要检查是否已打开 IP 转发，以 **root** 身份运行以下命令：*

/usr/sbin/sysctl net.ipv4.ip_forward

*要检查是否打开了非本地绑定，以 **root** 身份运行以下命令：*

/usr/sbin/sysctl net.ipv4.ip_nonlocal_bind

*如果上述命令都返回 **1**，则启用相应的设置。*

3.8. 在实时服务器上配置服务

*如果实际服务器是 Red Hat Enterprise Linux 系统，请将相应的服务器守护进程设置为在引导时激活。这些守护进程可以包括用于 Web 服务的 **httpd** 或用于 FTP 或 Telnet 服务的 **xinetd**。*

*远程访问实际服务器也可能有用，因此还应安装并运行 **sshd** 守护进程。*

第 4 章 使用 KEEPALIVED 初始负载均衡器配置

安装 Load Balancer 软件包后，您必须执行一些基本步骤来设置 LVS 路由器和实际服务器以用于 Keepalived。本章详细介绍了这些初始步骤。

4.1. 基本 KEEPALIVED 配置

在这个基本示例中，两个系统配置为负载平衡器。LB1 (Active) 和 LB2 (Backup) 将对运行 **httpd** 的四个 Web 服务器池的路由请求路由到 httpd，其中实际 IP 地址为 192.168.1.20 到 192.168.1.24，共享虚拟 IP 地址 10.0.0.1。每个负载均衡器有两个接口(**eth0** 和 **eth1**)，一个用于处理外部互联网流量，另一个用于将请求路由到真实服务器。使用的负载平衡算法是 round Robin，路由方法将是网络地址转换。

4.1.1. 创建 keapalived.conf 文件

keepalived 通过配置为负载均衡器的每个系统中的 **keepalived.conf** 文件进行配置。要创建类似 第 4.1 节 "基本 Keepalived 配置" 所示的负载均衡器拓扑，请使用文本编辑器在活跃和备份负载均衡器(LB1 和 LB2)中打开 **keepalived.conf**。例如：

```
vi /etc/keepalived/keepalived.conf
```

一个带有配置的基本负载均衡系统，如 第 4.1 节 "基本 Keepalived 配置" 所述，它有一个 **keepalived.conf** 文件，如以下代码部分所述。在本例中，**keepalived.conf** 文件在活动和备份路由器上相同，但 VRRP 实例除外，如 所述 第 4.1.1.2 节 "VRRP 实例"

4.1.1.1. 全局定义

keepalived.conf 文件的 Global Definitions 部分允许管理员在发生负载均衡器更改时指定通知详情。请注意，全局定义是可选的，对于 Keepalived 配置不需要。**keepalived.conf** 文件的这一部分在 LB1 和 LB2 上都相同。

```
global_defs {

  notification_email {
    admin@example.com
  }
  notification_email_from noreply@example.com
  smtp_server 127.0.0.1
  smtp_connect_timeout 60
}
```

notification_email 是负载均衡器的管理员，而 **notification_email_from** 是一个发送负载均衡器状态更改的地址。SMTP 特定配置指定从中发送通知的邮件服务器。

4.1.1.2. VRRP 实例

以下示例显示了 master 路由器和备份路由器中的 **keeplalived.conf** 文件的 **vrrp_sync_group** 小节。请注意，**state** 和 **priority** 值在两个系统之间有所不同。

以下示例显示了 LB1 中 **keepalived.conf** 文件的 **vrrp_sync_group** 小节，即 master 路由器。

```
vrrp_sync_group VG1 {
  group {
    RH_EXT
```

```
      RH_INT
    }
}

vrrp_instance RH_EXT {
    state MASTER
    interface eth0
    virtual_router_id 50
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass passw123
    }
    virtual_ipaddress {
    10.0.0.1
    }
}

vrrp_instance RH_INT {
    state MASTER
    interface eth1
    virtual_router_id 2
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass passw123
    }
    virtual_ipaddress {
        192.168.1.1
    }
}
```

以下示例显示了 LB2 的 ***keepalived.conf*** *文件的* ***vrrp_sync_group*** *小节，即备份路由器。*

```
vrrp_sync_group VG1 {
    group {
      RH_EXT
      RH_INT
    }
}

vrrp_instance RH_EXT {
    state BACKUP
    interface eth0
    virtual_router_id 50
    priority 99
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass passw123
    }
    virtual_ipaddress {
    10.0.0.1
    }
```

```
}

vrrp_instance RH_INT {
  state BACKUP
  interface eth1
  virtual_router_id 2
  priority 99
  advert_int 1
  authentication {
     auth_type PASS
     auth_pass passw123
  }
  virtual_ipaddress {
     192.168.1.1
  }
}
```

在这些示例中，**vrrp_sync_group** 小节定义了通过任何状态更改（如 failover）来保持在一起的 VRRP 组。为与互联网通信的外部接口定义了一个实例(RH_EXT)，也为内部接口(RH_INT)定义了一个实例。

vrrp_instance 行详细说明了 VRRP 服务守护进程的虚拟接口配置，它会创建虚拟 IP 实例。**MASTER 状态指定活动服务器，状态 BACKUP 指定备份服务器。**

interface 参数将物理接口名称分配给这一特定的虚拟 **IP** 实例。

virtual_router_id 是虚拟路由器实例的数字标识符。它在所有参与此虚拟路由器的 **LVS** 路由器系统上必须相同。它用于区分在同一网络接口中运行的 **keepalived** 实例。

优先级指定分配接口在故障切换中接管的顺序；数值越大，优先级越高。此优先级值必须在 **0** 到 **255** 范围内，并且配置为 **状态 MASTER** 的 **Load Balancing** 服务器的优先级值应设置为比配置为 **状态 BACKUP** 的优先级值高的值。

身份验证块指定身份验证类型 **(auth_type)**和密码**(auth_pass)**，用于验证服务器以进行故障转移同步。**PASS** 指定密码身份验证；**Keepalived** 还支持 **AH** 或身份验证标头来实现连接完整性。

最后，**virtual_ipaddress** 选项指定接口虚拟 **IP** 地址。

4.1.1.3. 虚拟服务器定义

keepalived.conf 文件的 **Virtual Server definitions** 部分在 **LB1** 和 **LB2** 上都相同。

```
virtual_server 10.0.0.1 80 {
   delay_loop 6
```

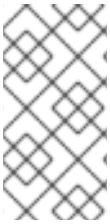
```
    lb_algo rr
    lb_kind NAT
    protocol TCP

    real_server 192.168.1.20 80 {
        TCP_CHECK {
            connect_timeout 10
        }
    }
    real_server 192.168.1.21 80 {
        TCP_CHECK {
            connect_timeout 10
        }
    }
    real_server 192.168.1.22 80 {
        TCP_CHECK {
            connect_timeout 10
        }
    }
    real_server 192.168.1.23 80 {
        TCP_CHECK {
            connect_timeout 10
        }
    }

}
```

在此块中，**virtual_server** 首先使用 IP 地址配置。然后，**delay_loop** 配置健康检查之间的时间（以秒为单位）。**lb_algo** 选项指定用于可用性的算法类型（本例中为 **r-Robin**；对于可能的 **lb_algo** 值列表，请参阅 *表 4.1 “适用于虚拟服务器的 lv_algo 值”*）。**lb_kind** 选项决定路由方法，本例中使用了网络地址转换(**ornat**)。

配置虚拟服务器详细信息后，将再次通过首先指定 IP 地址来配置 **real_server** 选项。**TCP_CHECK** 小节使用 **TCP** 检查实际服务器的可用性。**connect_timeout** 以秒为单位配置超时前的时间。



注意

不支持从负载均衡器或其中一个实际服务器访问虚拟 IP。同样，不支持在与真实服务器相同的计算机上配置负载平衡器。

表 4.1. 适用于虚拟服务器的 lv_algo 值

算法名称	lv_algo 值
round-Robin	**rr**
加权 round-Robin	**wrr**

算法名称	lv_algo 值
minimum-Connection	**lc**
加权 Least-Connection	**wlc**
基于本地性的 Least-Connection	**lblc**
使用复制基于本地的 Least-Connection 计划	**lblcr**
目标哈希	**dh**
源哈希	**sh**
源预期 Delay	**sed**
Never Queue	**nq**

4.2. KEEPALIVED DIRECT ROUTING 配置

Keepalived 的直接路由配置与 NAT 配置类似。在以下示例中，Keepalived 配置为为在端口 80 上运行 HTTP 的一组实际服务器提供负载平衡。要配置直接路由，将 lb_kind 参数改为 DR。其他配置选项在 第 4.1 节 “基本 Keepalived 配置” 中讨论。

以下示例显示了使用直接路由的 Keepalived 配置中的活跃服务器的 keepalived.conf 文件。

```
global_defs {
   notification_email {
     admin@example.com
   }
   notification_email_from noreply_admin@example.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 60
}

vrrp_instance RH_1 {
    state MASTER
    interface eth0
    virtual_router_id 50
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass passw123
    }
    virtual_ipaddress {
        172.31.0.1
    }
```

```
}

virtual_server 172.31.0.1 80 {
   delay_loop 10
   lb_algo rr
   lb_kind DR
   persistence_timeout 9600
   protocol TCP

   real_server 192.168.0.1 80 {
      weight 1
      TCP_CHECK {
       connect_timeout 10
       connect_port    80
      }
   }
   real_server 192.168.0.2 80 {
      weight 1
      TCP_CHECK {
       connect_timeout 10
       connect_port    80
      }
   }
   real_server 192.168.0.3 80 {
      weight 1
      TCP_CHECK {
       connect_timeout 10
       connect_port    80
      }
   }
}
```

以下示例显示了使用直接路由的 Keepalived 配置中的备份服务器的 keepalived.conf 文件。请注意，state 和 priority 值与活动服务器上的 keepalived.conf 文件不同。

```
global_defs {
   notification_email {
    admin@example.com
   }
   notification_email_from noreply_admin@example.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 60
}

vrrp_instance RH_1 {
   state BACKUP
   interface eth0
   virtual_router_id 50
   priority 99
   advert_int 1
   authentication {
      auth_type PASS
      auth_pass passw123
   }
```

```
    virtual_ipaddress {
        172.31.0.1
    }
}

virtual_server 172.31.0.1 80 {
    delay_loop 10
    lb_algo rr
    lb_kind DR
    persistence_timeout 9600
    protocol TCP

    real_server 192.168.0.1 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
    real_server 192.168.0.2 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
    real_server 192.168.0.3 80 {
        weight 1
        TCP_CHECK {
          connect_timeout 10
          connect_port    80
        }
    }
}
```

4.3. 启动服务

在负载均衡器配置的服务器中输入以下命令来启动该服务：

```
# systemctl start keepalived.service
```

要使 Keepalived 服务在重启后持久保留，在负载均衡器配置的服务器中输入以下命令：

```
# systemctl enable keepalived.service
```

第 5 章 HAPROXY 配置

本章解释了基本设置的配置，其中重点介绍了管理员为高可用性环境部署 HAProxy 服务时可能会遇到的常见配置选项。

HAProxy 拥有自己的一组调度算法，用于负载平衡。这些算法在 第 5.1 节 “HAProxy 调度算法” 中描述。

通过编辑 /etc/haproxy/haproxy.cfg 文件来配置 HAProxy。

使用 HAProxy 的负载平衡器配置由五个部分组成，用于配置：

- 第 5.2 节 “全局设置”
- proxys 部分，由 4 个部分组成：
  - 第 5.3 节 “默认设置” 设置
  - 第 5.4 节 “前端设置” 设置
  - 第 5.5 节 “后端设置” 设置

5.1. HAPROXY 调度算法

负载平衡的 HAProxy 调度算法可以在 /etc/haproxy/haproxy.cfg 配置文件的 balance 部分中的 backend 参数中编辑。请注意，HAProxy 支持配置多个后端，每个后端都可以配置调度算法。

Round-Robin (roundrobin)

在实际服务器池之间按顺序分发每个请求。使用这种算法时，所有实际服务器都被视为相等，而不考虑容量或负载。此调度模型类似于循环 DNS，但由于它是基于网络连接，而非基于主机，因此更加精细。Load Balancer round-robin 调度也不会遭遇缓存的 DNS 查询导致的静默。但是，在 HAProxy 中，由于可以使用此调度程序实时配置服务器权重，活动服务器的数量将限制为每个后端 4095 个。

静态 Round-Robin (static-rr)

与 round-Robin 一样，按顺序分发每个请求的实际服务器池，但不允许动态配置服务器权重。但是，由于服务器权重的静态性质，后端中活动服务器的数量没有限制。

least-Connection (leastconn)

将更多请求分发到活动连接较少的实际服务器。具有不同会话或连接长度的动态环境的管理员可能会发现此调度程序更适合其环境。它也适用于一组服务器具有不同容量的环境，因为管理员可以使用此调度程序实时调整权重。

源 (source)

通过对请求源 IP 地址的请求进行哈希并除以所有正在运行的服务器的权重，从而将请求分发到服务器，以确定哪个服务器将获取该请求。在运行所有服务器的场景中，源 IP 请求将持续由相同的实际服务器提供。如果正在运行的服务器的数量或权重有变化，则会话可能会移到另一台服务器，因为哈希/轻量级结果已改变。

URI (uri)

通过哈希整个 URI（或 URI 的可配置部分）将请求分发到服务器，并按所有运行服务器的权重划分，以确定哪个服务器将请求该请求。在运行所有活动服务器的场景中，目标 IP 请求将持续由相同的实际服务器提供。此调度程序可进一步配置，例如 URI 目录部分开始时的字符长度，以计算哈希结果，以及 URI 中的目录深度（通过 URI 中的正斜杠指定）来计算哈希结果。

URL 参数 (url_param)

通过在源 URL 请求中查找特定参数字符串，并将哈希计算除以所有正在运行的服务器的权重，从而向服务器分发请求。如果 URL 中缺少 参数，调度程序默认为 round-robin 调度。修饰符可以基于 POST 参数以及等待限制，具体取决于管理员在计算哈希结果前分配给特定参数的最大八进制数。

标头名称(hdr)

通过检查每个源 HTTP 请求中的特定标头名称并将哈希计算除以所有正在运行的服务器的权重，从而将请求分发到服务器。如果缺少标头，调度程序默认为 round-robin 调度。

RDP Cookie (rdp-cookie)

通过为每个 TCP 请求查找 RDP Cookie 并将哈希计算除以所有正在运行的服务器的权重，向服务器分发请求。如果缺少标头，调度程序默认为 round-robin 调度。此方法适用于持久性，因为它维护会话的完整性。

5.2. 全局设置

global 设置配置应用到运行 HAProxy 的所有服务器的参数。典型的 global 部分可能类似如下：

```
global
    log 127.0.0.1 local2
    maxconn 4000
    user haproxy
    group haproxy
    daemon
```

在上述配置中，管理员已将服务配置为 log 所有条目到本地 syslog 服务器。默认情况下，这可能是 /var/log/syslog 或一些用户指定的位置。

maxconn 参数指定服务的最大并发连接数。默认情况下，最大值为 2000。

user 和 group 参数指定 haproxy 进程所属的用户名和组名称。

最后，daemon 参数指定 haproxy 作为后台进程运行。

5.3. 默认设置

default 设置配置应用到配置中的所有 proxy 子部分的参数(frontend、backend、listen)。典型的 default 部分可能类似如下：

注意

proxy 子部分中配置的任何参数(frontend、backend 或 listen)都优先于 default 中的参数值。

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    retries                 3
    timeout http-request    10s
    timeout queue           1m
```

```
    timeout connect         10s
    timeout client          1m
    timeout server          1m
```

mode 指定 HAProxy 实例的协议。使用 http 模式将源请求连接到基于 HTTP 的实际服务器，非常适合负载平衡 web 服务器。对于其他应用程序，使用 tcp 模式。

log 指定日志条目要写入的日志地址和 syslog 工具。global 值将 HAProxy 实例指代为 log 部分中的 global 参数中指定的任何内容。

option httplog 启用 HTTP 会话的各种值的日志记录，包括 HTTP 请求、会话状态、连接号、源地址和连接计时器。

option dontlognull 禁用 null 连接的日志记录，这意味着 HAProxy 不会记录没有传输数据的连接。对于 Web 应用等环境不建议这么做，其中空连接可能会指出漏洞的开放端口扫描等恶意活动。

retries 指定实际服务器在第一次尝试连接失败后，重试连接请求的次数。

各种 timeout 值指定给定请求、连接或响应的不活跃时间长度。这些值通常以毫秒为单位表示（除非另有明确说明），但是可以通过将单位加到数字值，以任何其他单位表示。支持的单位为 us（微秒）、ms（毫秒）、s（秒）、m（分钟）、h（小时）和 d（天）。http-request 10s 提供 10 秒以等待客户端的完整 HTTP 请求。queue 1m 设置一分钟，作为连接被丢弃前等待的时间，客户端会收到 503 或 "Service Unavailable" 错误。connect 10s 指定服务器成功连接的秒数。client 1m 指定客户端可以保持不活跃的时间（以分钟为单位）（不接受或发送数据）。server 1m 指定服务器在超时发生前接受或发送数据的时间（以分钟为单位）。

5.4. 前端设置

frontend 设置为客户端连接请求配置服务器的侦听套接字。frontend 的典型 HAProxy 配置可能类似如下：

```
frontend  main
  bind 192.168.0.10:80
  default_backend app
```

名为 frontend 的 main 配置为 192.168.0.10 IP 地址，并使用 bind 参数侦听端口 80。连接后，use backend 指定所有会话都连接到 app 后端。

5.5. 后端设置

backend 设置指定实际服务器 IP 地址和负载均衡器调度算法。以下示例显示了一个典型的 backend 部分：

```
backend app
  balance     roundrobin
  server  app1 192.168.1.1:80 check
  server  app2 192.168.1.2:80 check
  server  app3 192.168.1.3:80 check inter 2s rise 4 fall 3
  server  app4 192.168.1.4:80 backup
```

后端服务器名为 app。balance 指定要使用的负载均衡器调度算法，本例中为 Round Robin (roundrobin)，但可以是 HAProxy 支持的任何调度程序。有关在 HAProxy 中配置调度程序的更多信息，请参阅 第 5.1 节 “HAProxy 调度算法”。

server 行指定后端中可用的服务器。app1 app4 是内部分配给每个实际服务器的名称。日志文件将按名称指定服务器消息。该地址是分配的 IP 地址。IP 地址中冒号后面的值是特定服务器上发生连接的端口号。check 选项为定期健康检查标记服务器，以确保它可用，并能够接收和发送数据并获取会话请求。服务器 app3 还将 healthcheck 间隔配置为 2 秒(inter 2s)，必须经过检查 app3 量来确定服务器是否被视为健康(rise 4)，在服务器被视为失败(fall 3)前，服务器连续失败的次数()。

5.6. 启动 HAPROXY

要启动 HAProxy 服务，请输入以下命令：

```
# systemctl start haproxy.service
```

要使 HAProxy 服务在重启后持久保留，请输入以下命令：

```
# systemctl enable haproxy.service
```

5.7. 将 HAPROXY 消息记录到 RSYSLOG

您可以通过写入 /dev/log 套接字，将您的系统配置为将 HAProxy 消息记录到 rsyslog。另外，您也可以将 TCP 回环地址作为目标，但这导致性能下降。

以下流程将 HAProxy 配置为将信息记录到 rsyslog。

1. 在 HAProxy 配置文件的 全局 部分中，使用 log 指令以 /dev/log 套接字为目标。

```
log /dev/log local0
```

2. 更新 前端、后端和 侦听 代理，以便将消息发送到您在 HAProxy 配置文件的 全局 部分中配置的 rsyslog 服务。为此，请将 log global 指令添加到配置文件的 defaults 部分，如下所示。

```
defaults
   log global
   option httplog
```

3. 如果您在 chroot 环境中运行 HAProxy，或者让 HAProxy 使用 chroot 配置指令为您创建 chroot 目录，那么必须在 chroot 目录中提供套接字。您可以通过修改 rsyslog 配置以在 chroot 文件系统内创建新的侦听套接字来完成此操作。为此，请在 rsyslog 配置文件中添加以下行：

```
$ModLoad imuxsock
$AddUnixListenSocket PATH_TO_CHROOT/dev/log
```

4. 要自定义 HAProxy 日志消息的显示位置，您可以使用 rsyslog 过滤器，如 系统管理员指南 中的 Rsyslog 基本配置 中所述。

附录 A. 示例配置：加载使用 HAPROXY 和 KEEPALIVED 的 CEPH 对象网关服务器

本附录提供了一个示例，显示了 HAProxy 和 Keepalived 与 Ceph 集群的配置。Ceph 对象网关允许您将多个对象网关实例分配到一个区域，以便您可以在负载增加时向外扩展。由于每个对象网关实例都有自己的 IP 地址，因此您可以使用 HAProxy 和 keepalived 在 Ceph 对象网关服务器之间平衡负载。

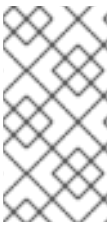
在这种配置中，HAproxy 在 Ceph 对象网关服务器之间执行负载平衡，而 Keepalived 用于管理 Ceph 对象网关服务器的虚拟 IP 地址和监控 HAProxy。

HAProxy 和 keepalived 的另一个用例是在 HAProxy 服务器上终止 HTTPS。Red Hat Ceph Storage(RHCS)1.3.x 使用 Civetweb，而 RHCS 1.3.x 中的实施不支持 HTTPS。您可以使用 HAProxy 服务器在 HAProxy 服务器上终止 HTTPS，并在 HAProxy 服务器和 Civetweb 网关实例之间使用 HTTP。这个示例包括此配置作为流程的一部分。

A.1. 先决条件

要使用 Ceph 对象网关设置 HAProxy，您必须有：

- 正在运行的 Ceph 群集；
- 同一区域中至少有两个 Ceph 对象网关服务器，配置为在端口 80 上运行；
- HAProxy 和 keepalived 至少两台服务器。



注意

此流程假定您至少有两个 Ceph 对象网关服务器运行，并且您在运行测试脚本时通过端口 80 获得有效的响应。

A.2. 准备 HAPROXY 节点

以下设置假定两个名为 haproxy 和 haproxy2 的 HAProxy 节点，以及名为 rgw1 和 rgw2 的 Ceph 对象网关服务器。您可以使用您喜欢的任何命名规则。在两个 HAProxy 节点上执行以下步骤：

1. 安装 Red Hat Enterprise Linux 7.

2. 注册节点。

```
# subscription-manager register
```

3. 启用红帽企业 Linux 7 服务器存储库。

```
# subscription-manager repos --enable=rhel-7-server-rpms
```

4. 更新服务器。

```
# yum update -y
```

5. 根据需要安装管理工具（例如 wget、 vim 等等）。

6. 打开端口 80。

```
# firewall-cmd --zone=public --add-port 80/tcp --permanent
# firewall-cmd --reload
```

7. 对于 HTTPS，打开端口 443。

```
# firewall-cmd --zone=public --add-port 443/tcp --permanent
# firewall-cmd --reload
```

A.3. 安装和配置 KEEPALIVED

在两个 HAProxy 节点上执行以下步骤：

1. 安装 keepalived。

```
# yum install -y keepalived
```

2. 配置 keepalived。

```
# vim /etc/keepalived/keepalived.conf
```

在以下配置中，有一个用于检查 HAProxy 进程的脚本。实例使用 eth0 作为网络接口，并将 haproxy 配置为主服务器，并将 haproxy2 配置为备份服务器。它还分配虚拟 IP 地址 192.168.0.100。

```
vrrp_script chk_haproxy {
  script "killall -0 haproxy" # check the haproxy process
  interval 2 # every 2 seconds
  weight 2 # add 2 points if OK
}

vrrp_instance VI_1 {
  interface eth0 # interface to monitor
  state MASTER # MASTER on haproxy, BACKUP on haproxy2
  virtual_router_id 51
  priority 101 # 101 on haproxy, 100 on haproxy2
  virtual_ipaddress {
    192.168.0.100 # virtual ip address
  }
  track_script {
    chk_haproxy
  }
}
```

3. 启用并开始 keepalived。

```
# systemctl enable keepalived
# systemctl start keepalived
```

A.4. 安装和配置 HAPROXY

在两个 HAProxy 节点上执行以下步骤：

1. 安装 haproxy。

```
# yum install haproxy
```

2. 为 SELinux 和 HTTP 配置 haproxy。

```
# vim /etc/firewalld/services/haproxy-http.xml
```

添加以下行：

```
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>HAProxy-HTTP</short>
<description>HAProxy load-balancer</description>
<port protocol="tcp" port="80"/>
</service>
```

以 root 用户身份，为 haproxy-http.xml 文件分配正确的 SELinux 上下文和文件权限。

```
# cd /etc/firewalld/services
# restorecon haproxy-http.xml
# chmod 640 haproxy-http.xml
```

3. **如果要使用 HTTPS，请为 SELinux 和 HTTPS 配置 haproxy。**

```
# vim /etc/firewalld/services/haproxy-https.xml
```

添加以下行：

```
<?xml version="1.0" encoding="utf-8"?>
<service>
<short>HAProxy-HTTPS</short>
<description>HAProxy load-balancer</description>
<port protocol="tcp" port="443"/>
</service>
```

以 root 用户身份，为 haproxy-https.xml 文件分配正确的 SELinux 上下文和文件权限。

```
# cd /etc/firewalld/services
# restorecon haproxy-https.xml
# chmod 640 haproxy-https.xml
```

4. **如果您打算使用 HTTPS，请为 SSL 生成密钥。如果您没有证书，您可以使用自签名证书。有关生成密钥和自签名证书的详情，请参考 Red Hat Enterprise Linux 系统管理员指南。**

最后，将证书和密钥放入 PEM 文件中。

```
# cat example.com.crt example.com.key > example.com.pem
# cp example.com.pem /etc/ssl/private/
```

5. 配置 HAProxy.

```
# vim /etc/haproxy/haproxy.cfg
```

haproxy.cfg 的 global 和 defaults 部分可能会保持不变。在 defaults 部分后，您需要配置 frontend 和 backend 部分，如下例所示：

```
frontend http_web *:80
   mode http
   default_backend rgw

frontend rgw-https
  bind <insert vip ipv4>:443 ssl crt /etc/ssl/private/example.com.pem
  default_backend rgw

backend rgw
   balance roundrobin
   mode http
   server  rgw1 10.0.0.71:80 check
   server  rgw2 10.0.0.80:80 check
```

6. enable/start haproxy

```
# systemctl enable haproxy
# systemctl start haproxy
```

A.5. 测试您的 HAPROXY 配置

在 HAProxy 节点上，检查以确保来自 keepalived 配置的虚拟 IP 地址会出现。

```
$ ip addr show
```

在 calamari 节点上，查看您是否能通过负载平衡器配置访问网关节点。例如：

```
$ wget haproxy
```

这应该返回相同的结果，如下：

```
$ wget rgw1
```

如果返回包含以下内容的 index.html 文件，则您的配置可以正常工作。

```
<?xml version="1.0" encoding="UTF-8"?>
 <ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
   <ID>anonymous</ID>
   <DisplayName></DisplayName>
  </Owner>
  <Buckets>
  </Buckets>
 </ListAllMyBucketsResult>
```

附录 B. 修订历史记录

修订 4.1-1 为 7.7 GA 发布准备文档.	Wed Aug 7 2019	Steven Levine
修订 3.1-2 为 7.6 GA 发布准备文档.	Thu Oct 4 2018	Steven Levine
修订 2.1-1 为 7.5 GA 发布准备文档.	Thu Mar 15 2018	Steven Levine
修订 2.1-0 为 7.5 Beta 版出版物准备文档.	Thu Dec 14 2017	Steven Levine
修订 0.6-5 7.4 的更新版本.	Wed Nov 22 2017	Steven Levine
修订 0.6-3 发布 7.4 GA 的文件版本.	Thu Jul 27 2017	Steven Levine
修订 0.6-1 为 7.4 Beta 版出版物准备文档.	Wed May 10 2017	Steven Levine
修订 0.5-9 针对 7.3 的更新版本.	Mon Dec 5 2016	Steven Levine
修订 0.5-7 7.3 GA 发布版本.	Mon Oct 17 2016	Steven Levine
修订 0.5-6 为 7.3 Beta 发布准备文档.	Thu Aug 18 2016	Steven Levine
修订 0.3-2 为 7.2 GA 发布准备文档.	Mon Nov 9 2015	Steven Levine
修订 0.3-0 为 7.2 Beta 版出版物准备文档.	Wed Aug 19 2015	Steven Levine
修订 0.2-6 7.1 GA 版本	Mon Feb 16 2015	Steven Levine
修订 0.2-5 7.1 Beta 发行版本的版本	Thu Dec 11 2014	Steven Levine
修订 0.2-4 7.1 Beta 发行版本的版本	Thu Dec 04 2014	Steven Levine
修订 0.1-12 7.0 GA 版本	Tue Jun 03 2014	John Ha
修订 0.1-6 红帽企业 Linux 7 测试版构建	Mon Jun 13 2013	John Ha
修订 0.1-1	Wed Jan 16 2013	John Ha

基于本文档的 Red Hat Enterprise Linux 6 版本进行分支

索引