ZStack 技术白皮书精选

ZStack--虚拟路由网络服务提供模块

扫一扫二维码，获取更多技术干货吧

版权声明

摘要

大道至简·极速部署，ZStack 致力于产品化私有云和混合云。

ZStack 是新一代创新开源的云计算 IaaS 软件，由英特尔、微软、CloudStack 等世界上最早一批虚拟化工程师创建，拥有 KVM、Xen、Hyper-V 等成熟的技术背景。

ZStack 创新提出了云计算 4S 理念，即 Simple （简单）、Strong （健壮）、Smart （智能）、Scalable （弹性），通过全异步架构，无状态服务架构，无锁架构等核心技术，完美解决云计算执行效率低，系统不稳定，不能支撑高并发等问题，实现 HA 和轻量化管理。

ZStack 发起并维护着国内最大的自主开源 IaaS 社区——zstack.io，吸引了 6000 多名社区用户，对外公开的 API 超过 1000 个。基于这 1000 多个 API，用户可以自由组装出自己的私有云、混合云，甚至利用 ZStack 搭建公有云对外提供服务。

ZStack 拥有充足的知识产权储备，积极申报多项软著和专利，参与业内标准、白皮书的撰写，入选云计算行业方案目录，还通过了工信部云服务能力认证和信通院可信云认证。ZStack 面向企业用户提供基于 IaaS 的私有云和混合云，是业内唯一一家实现产品化，并领先业内首家推出同时打通数据面和控制面无缝混合云的云服务商。选择 ZStack，用户可以官网直接下载、1 台 PC 也可上云、30 分钟完成从裸机的安装部署。

目前已有 1000 多家企业用户选择了 ZStack 云平台。

ZSTACK--虚拟路由网络服务提供模块

在 ZStack 的网络模型中，OSI 第 4～7 层网络服务被实现为来自不同服务提供模块的小插件。默认提供模块，称为虚拟路由，采用定制的 Linux 虚拟机作为虚拟设备，为每一个 L3 网络提供包括 DHCP、DNS、NAT、EIP 和端口转发在内的网络服务。使用虚拟机作为虚拟路由器的方式的优点有：没有单点故障、对物理设备没有特殊要求，因此用户无需购买昂贵的硬件，就可以在商用设备上实现各种网络服务。

概述

正如“ZStack--网络模型 1：L2 和 L3 网络”中提到的，ZStack 以小插件的方式设计网络服务，供应商可以通过创建网络服务提供模块的方式，选择性地实现他们的硬件或软件支持的网络服务。默认情况下，ZStack 带有一个虚拟路由，它负责使用一个应用虚拟机（Virtual Router VM）实现所有网络服务。

> ***注：****事实上 ZStack 有另一个提供模块称为安全组提供模块，它提供了分布式防火墙功能。我们称虚拟路由为默认的提供模块，因为它提供了最常见的，一个云需要的网络服务。*

在 IaaS 软件中，实现网络服务有几种方法。

一种方式是使用中心的、功能强大的网络节点，它们通常是一些物理服务器；通过聚集来自不同租户的流量，网络节点将负责流量隔离并使用类似 Linux 网络命名空间的技术来提供网络服务。

另一种方法是使用专用的网络硬件，例如，可编程的物理交换机、物理防火墙、物理负载平衡器，它会要求用户去购买特定的硬件。

最后一个方法是使用网络功能虚拟化（NFV）技术，像 ZStack 的虚拟路由虚拟机，就是在商用 x86 服务器上虚拟化网络服务。

每种方法都有优点和缺点；我们选择 NFV 作为我们的解决方案是出于如下考虑：

1.需要最少的基础设施：解决方案应该对用户的物理基础设施需求很少甚至为零；也就是说，用户不应该改变现有的基础设施或购买特殊的基础设施来迎合 IaaS 软件的网络模型。我们不想强迫用户购买特定的硬件或要求他们在一组主机前放置一些特殊的功能服务器。

2.没有单点故障：解决方案应该是采用没有单点故障的分布式的方式。一个网络节点崩溃应该只能影响拥有它的租户，不应该影响任何其他租户。

3.无状态：网络节点应该是无状态的，这样 IaaS 软件在发生无法预料的错误后，可以轻易摧毁并重新创建它们。

4.利于高可用性（HA）：解决方案应该易于实现高可用，这样租户可以要求部署富余的网络节点。

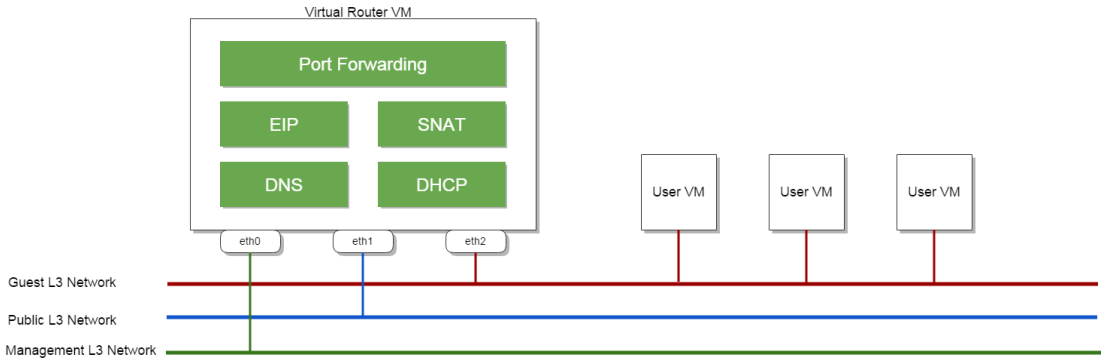
5.不依赖虚拟机管理程序：解决方案不应该依赖于 Hypervisor，并且应该和主流的 Hypervisor 完美结合工作，包括 KVM、XEN、VMWare 和 Hyper-V。

6.较好的性能：解决方案应该为大多数使用场景提供合理的网络性能。

基于虚拟路由的 NFV 解决方案满足上述所有考虑。我们选择它作为默认的实现，同时也为开发人员提供了采用其他解决方案的可能。

虚拟路由

应用虚拟机（Appliance VMs）是一种特别的虚拟机，运行着定制的 Linux 操作系统，以及特别的帮助管理云的 agents。*虚拟路由虚拟机*是应用虚拟机概念的第一个实现。这个想法，简单的说，在用户虚拟机第一次被创建的时候，去创建一个为某个 L3 网络提供全部网络服务的*虚拟路由虚拟机*，只要这个 L3 网络上开启了虚拟路由提供的网络服务。每个*虚拟路由虚拟机*包含一个 Python agent，它通过 HTTP 协议接收来自 ZStack 管理节点的命令，并在同一 L3 网络给用户虚拟机提供包括 DHCP、DNS、NAT、EIP 和端口转发的网络服务。



上图，显示了在客户 L3 网络上启用了所有网络服务的网络拓扑结构。一个虚拟路由通常有三种 L3 网络：

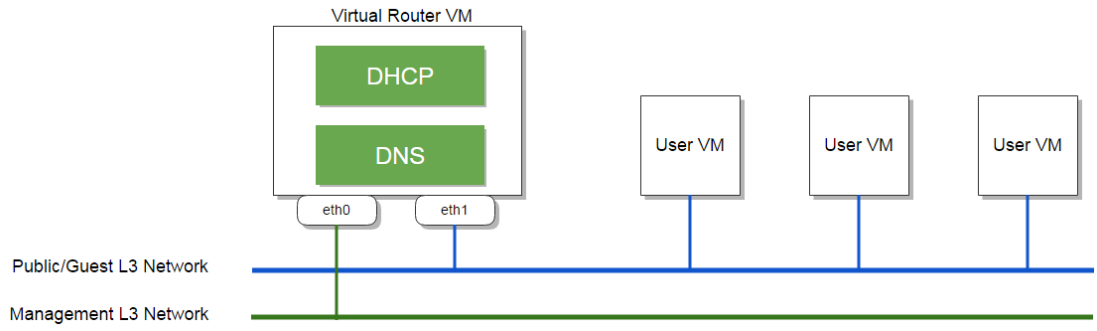
1.**一个 L3 管理网络**指的是 ZStack 管理节点和在*虚拟路由虚拟机*内的 Python agent 通过 HTTP 协议进行通信的网络，是一个必须的网络，每个虚拟路由都有。

2.**一个公有 L3 网络**是一个可以连接互联网的可选网络，它在虚拟路由虚拟机内提供了默认的路由。如果省略，*L3 管理网络*将同时被作为管理网络和公有网络使用。

公有网络不需要允许公开访问：*把用户虚拟机和外部世界（数据中心的其他网络或互联网）相连的公有网络不需要允许公开访问。例如，当桥接被 VLAN 和数据中心的其他网络(10.x.x.x/x)隔离的客户 L3 网络（192.168.1.0 / 24）时，网络 10.0.1.0/24 可以是一个公有网络，即使它不能被互联网访问。*

3.**一个客户 L3 网络**是用户虚拟机连接的网络；和网络服务相关的流量在用户虚拟机和虚拟路由虚拟机内流动。

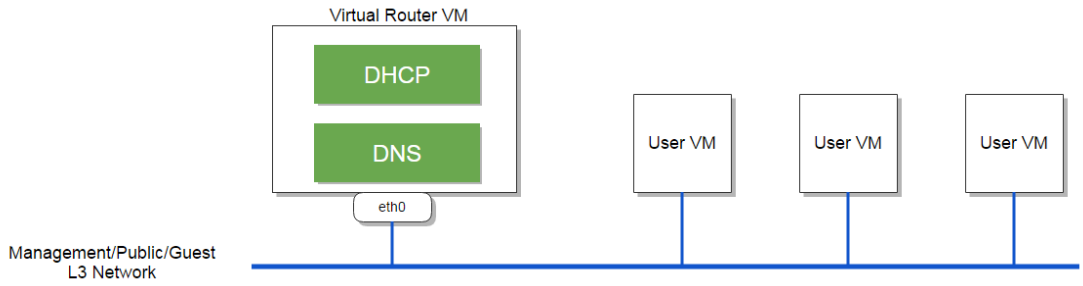
对不同的网络服务组合，L3 网络数量是可变的。例如，如果 DHCP 和 DNS 被启用，网络的拓扑结构变为：



因为没有 NAT 相关的服务（例如 SNAT，EIP），用户的虚拟机不需要一个单独的、隔离的客户 L3 网络，但可以直接连接到公共网络。

注意：*当然，你可以创建一个只有 DHCP 和 DNS 服务的、隔离的客户 L3 网络，该网络上的虚拟机可以获得 IP，但不能访问外部网络，因为缺失了 SNAT 服务。*

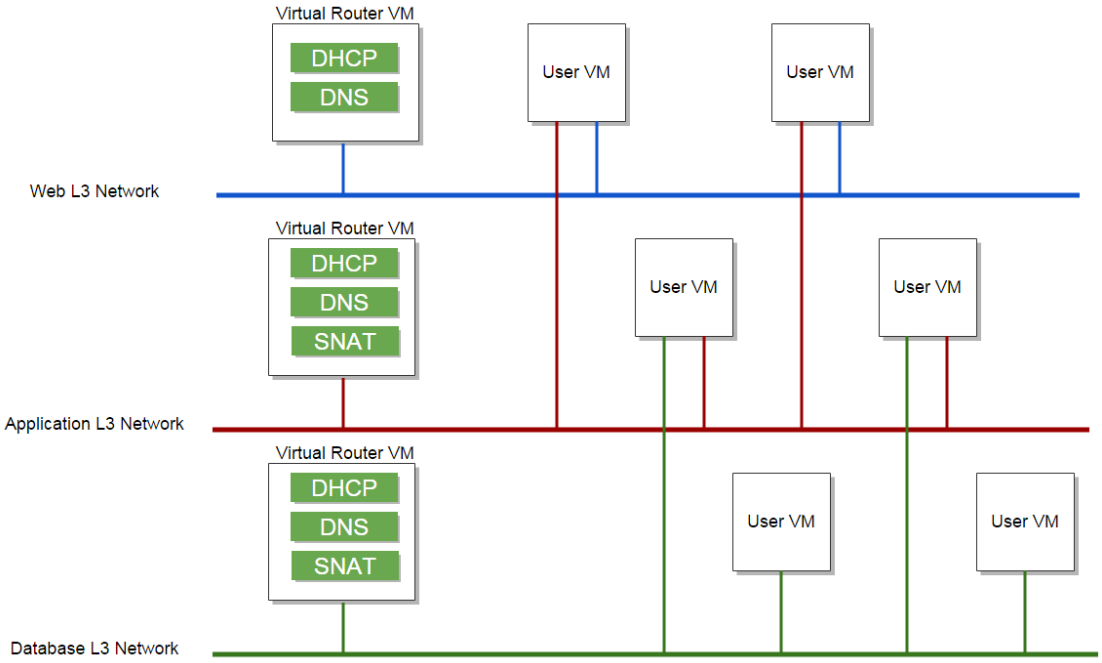
如果我们省略了上图中的 L3 公有网络，那么网络拓扑将变成：



用户可以使用一个虚拟路由计算规格来配置一个虚拟路由虚拟机的 L3 管理网络、L3 公有网络、CPU 速度、以及内存大小。当创建一个虚拟路由虚拟机的时候，ZStack 将会尝试去找到一个合适的虚拟路由计算规格。一个系统标签 `guestL3Network::{l3NetworkUuid},` 可以被用于为一个 L3 客户网络指定一个虚拟路由计算规格，如果没有指定的规格被找到，将会使用一个默认的规格。

注意：关于系统标签，请参阅 *The Tag System*。

在这个 ZStack 版本中（0.6），一个 L3 客户网络可以含有并只能含有一个虚拟路由虚拟机，对于一个多层网络的环境，不同的虚拟路由虚拟机将会服务不同的层：



ZStack 管理节点将会向处于一个虚拟路由虚拟机内部的 Python agent 发送命令，当用户虚拟机启动会停止的时候。以通过 dnsmasq 和 iptables 实现网络服务。Iptables 规则的一小段像这样：

```
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [1828:141919]
:appliancevm - [0:0]
:snat-fwd-eth1 - [0:0]
-A INPUT -p tcp -m tcp --dport 7272 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 7759 -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -d 192.168.0.236/32 -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -j appliancevm
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth0 -o eth1 -j snat-fwd-eth1
-A FORWARD -i eth1 -o eth0 -j snat-fwd-eth1
-A FORWARD -i eth1 -o eth1 -j snat-fwd-eth1
-A appliancevm -i eth0 -p tcp -m state --state NEW -m tcp --dport 7272 -j ACCEPT
-A appliancevm -i eth0 -p tcp -m state --state NEW -m tcp --dport 9393 -j ACCEPT
-A appliancevm -i eth1 -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A appliancevm -i eth1 -p udp -m state --state NEW -m udp --dport 67:68 -j ACCEPT
-A snat-fwd-eth1 -j ACCEPT
COMMIT
```

注意：*在未来的 ZStack 版本中，网络服务：负载均衡，VPN，GRE 隧道，也将会通过虚拟路由虚拟机来实现。另外虚拟路由虚拟机也将会成为虚拟专用云 VPC 的核心实现元素。*

虚拟路由虚拟机是怎样满足以下考虑的

让我们回顾下我们先前提到的一些考虑，然后看下虚拟路由虚拟机如何能满足它们。

1. **最小的基础设施需求：**虚拟路由虚拟机，对数据中心的物理设备没有任何特别需要。它们只是一些类似于用户虚拟机的虚拟机，可以在物理机上被创建。因为使用它们，管理员不必去为复杂的硬件互联做规划。
2. **没有单点故障：**对每一个 L3 网络都有一个虚拟路由虚拟机，如果它因为某种原因崩溃了，只有对应的 L3 网络上的用户虚拟机会被影响，而不会对其他 L3 网络产生任何影响。在大多数的使用场景中，一个 L3 网络只属于一个租户，这就是说，只有一个租户会遭受到一个虚拟路由虚拟机的失败。当 L3 网络遭到恶意工具的时候，这特别有用。例如，DDOS。攻击者不能通过攻击一个租户而摧毁整个云内的网络。
3. **无状态：**虚拟路由虚拟机是无状态的，所有的配置，来自于 ZStack 管理节点，可以在任何时间被重建。用户有各种各样的选择，用于重建虚拟路由虚拟机中的配置。例如，关闭、启动它们，删除、重建他们，或调用重连 API（ReconnectVirtualRouter API）。
4. **易于实现高可用（HA）：**可以部署两个虚拟路由虚拟机，使用虚拟路由冗余协议在主备模式下工作，以实现 HA。一旦主要的虚拟路由失效了，备用的会自动接管，使得网络的宕机时间微不足道。

   注意：*这个冗余虚拟路由虚拟机的特性在当前版本中不支持（0.6）.*

5. **Hypervisor 无关：**虚拟路由虚拟机不依赖于 Hypervisor。ZStack 有一个脚本，用于为主流的 Hypervisor 构建虚拟路由虚拟机的模板。
6. **合理的性能：**因为使用了 Linux，虚拟路由虚拟机能够实现该 Liunx 能够提供的合理的性能。用户可以配置一个虚拟路由计算规格，通过更多的 CPU 和更大的内存来为虚拟路由虚拟机提供足够的计算能力，以应对沉重的网络流量。性能上主要的关注点在于，虚拟路由虚拟机和用户虚拟机上公有网卡间的流量，在虚拟路由虚拟机提供 NAT 相关的服务，包括 SNAT、EIP、和端口转发时。在大多数场景中，由于一个公网 IP 通常有几十 MB 的带宽，虚拟路由虚拟机足以胜任一个不错的性能。

   然而，当通过虚拟路由虚拟机的流量需要一个极高的带宽的时候，由于虚拟化导致的显著的网络性能下降时不可避免的；然而，有两种技术可以缓解这个问题：

a. LXC/Docker:由于 ZStack 支持多种 Hypervisor，一旦 LXC 或 Docker 被支持，作为一种轻量级的虚拟化技术，作为容器运行的虚拟路由虚拟机可以近乎原生的性能。

b. SR-IOV:虚拟路由虚拟机可以通过 SR-IOV 被分配物理网卡，以达到原生的网络性能。

注意：*LXC/Docker 和 SR-IOV 在当前版本中不支持（0.6）.*

另外，用户可以使用系统标签和虚拟路由计算规格来为虚拟路由虚拟机控制物理主机的分配；更进一步，用户甚至可以指派一个物理服务器给一个虚拟路由虚拟机；在 LXC/Docker 或 SR-IOV 的帮助下，虚拟路由虚拟机能接近一个 Linux 服务器能够提供的原生的网络性能。

不管怎样，软件的解决方案有着天生的性能缺陷；用户可以选择为了网络的高性能而选择混合的解决方案；例如，仅为 DHCP 和 DNS 使用虚拟路由虚拟机，将性能敏感的服务留给使用了物理设备的服务提供器。

总结

在这篇文章中，我们演示了 ZStack 的默认网络服务提供器：虚拟路由提供器。解释了它怎么工作并详细阐述了它是怎样满足了我们关于网络服务的考虑。借助虚拟路由虚拟机，ZStack 取得了一个理想的平衡，在灵活性和性能之间。我们相信 90%的用户可以轻松明确地在商业硬件上构建他们的网络服务。