**[Kubernetes网络原理及方案](https://www.kubernetes.org.cn/2059.html)**

2017-05-26 21:09 发起人: [有容云](https://www.kubernetes.org.cn/author/youruncloud) 分类：[Kubernetes实践分享/开发实战](https://www.kubernetes.org.cn/practice) 阅读(4840) 作者：阳运生/[原文](http://www.youruncloud.com/blog/131.html) 评论(0)

大家好，说到容器、[Docker](https://www.kubernetes.org.cn/tags/docker" \t "_blank)，大家一定会想到Kubernetes，确实如此，在2016年ClusterHQ容器技术应用调查报告显示，Kubernetes的使用率已经达到了40%，成为最受欢迎的容器编排工具；那么Kubernetes到底是什么呢？它是一个用于容器集群的自动化部署、扩容以及运维的开源平台；那么通过Kubernetes能干什么呢？它能快速而有预期地部署你的应用，极速地扩展你的应用，无缝对接新的应用功能，节省资源，优化硬件资源的使用。

随着Kubernetes王者时代的到来，计算、[网络](https://www.kubernetes.org.cn/tags/%e7%bd%91%e7%bb%9c" \t "_blank)、[存储](https://www.kubernetes.org.cn/tags/%e5%ad%98%e5%82%a8)、安全是Kubernetes绕不开的话题，本次主要分享Kubernetes网络原理及方案，后续还会有Kubernetes其它方面的分享，另外有容云5.22发布了基于Kubernetes的容器云平台产品[UFleet](http://www.youruncloud.com/ufleet.html" \t "_blank)，想要获取新品试用，欢迎联系有容云。

**一、Kubernetes网络模型**

在Kubernetes网络中存在两种IP（Pod IP和Service Cluster IP），Pod IP 地址是实际存在于某个网卡(可以是虚拟设备)上的，Service Cluster IP它是一个虚拟IP，是由kube-proxy使用Iptables规则重新定向到其本地端口，再均衡到后端Pod的。下面讲讲Kubernetes Pod网络设计模型：

**1、基本原则：**

每个Pod都拥有一个独立的IP地址（IPper Pod），而且假定所有的pod都在一个可以直接连通的、扁平的网络空间中。

**2、设计原因：**

用户不需要额外考虑如何建立Pod之间的连接，也不需要考虑将容器端口映射到主机端口等问题。

**3、网络要求：**

所有的容器都可以在不用NAT的方式下同别的容器通讯；所有节点都可在不用NAT的方式下同所有容器通讯；容器的地址和别人看到的地址是同一个地址。

**二、Docker网络基础**

* **Linux网络名词解释：**

**1、网络的命名空间：**Linux在网络栈中引入网络命名空间，将独立的网络协议栈隔离到不同的命令空间中，彼此间无法通信；docker利用这一特性，实现不容器间的网络隔离。

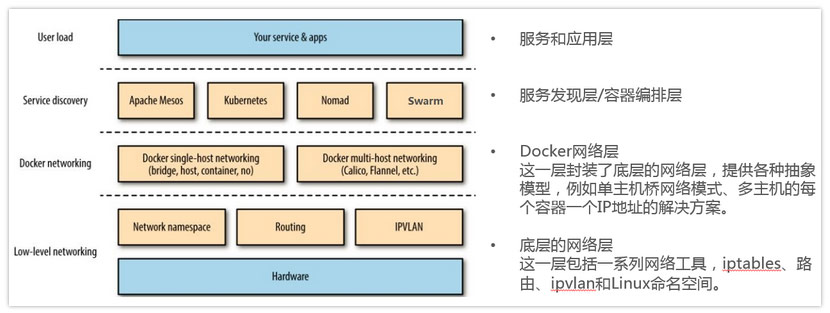
**2、Veth设备对：**Veth设备对的引入是为了实现在不同网络命名空间的通信。

**3、Iptables/Netfilter：**Netfilter负责在内核中执行各种挂接的规则(过滤、修改、丢弃等)，运行在内核 模式中；Iptables模式是在用户模式下运行的进程，负责协助维护内核中Netfilter的各种规则表；通过二者的配合来实现整个Linux网络协议栈中灵活的数据包处理机制。

**4、网桥：**网桥是一个二层网络设备,通过网桥可以将linux支持的不同的端口连接起来,并实现类似交换机那样的多对多的通信。

**5、路由：**Linux系统包含一个完整的路由功能，当IP层在处理数据发送或转发的时候，会使用路由表来决定发往哪里。

* **Docker生态技术栈**

下图展示了Docker网络在整个Docker生态技术栈中的位置：

* **Docker网络实现**

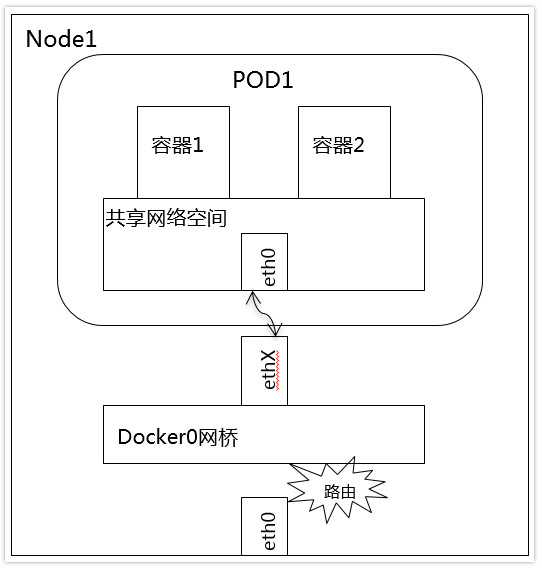
**1、单机网络模式：**Bridge 、Host、Container、None，这里具体就不赘述了。

**2、多机网络模式：**一类是 Docker 在 1.9 版本中引入Libnetwork项目，对跨节点网络的原生支持；一类是通过插件（plugin）方式引入的第三方实现方案，比如 Flannel，Calico 等等。

**三、Kubernetes网络基础**

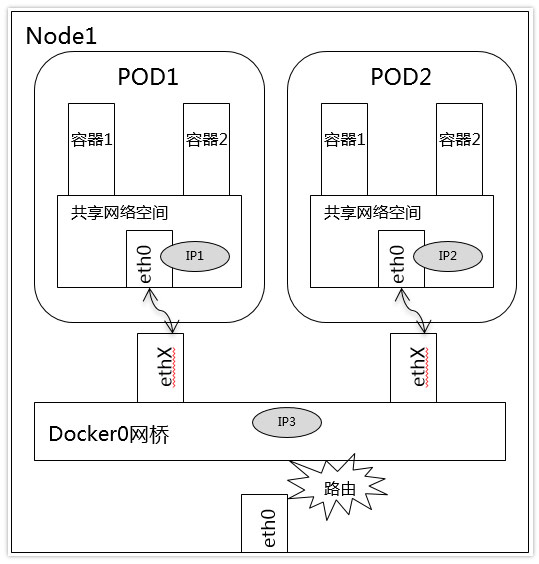
**1、容器间通信：**

同一个[Pod](https://www.kubernetes.org.cn/tags/pod" \t "_blank)的容器共享同一个网络命名空间，它们之间的访问可以用localhost地址 + 容器端口就可以访问。



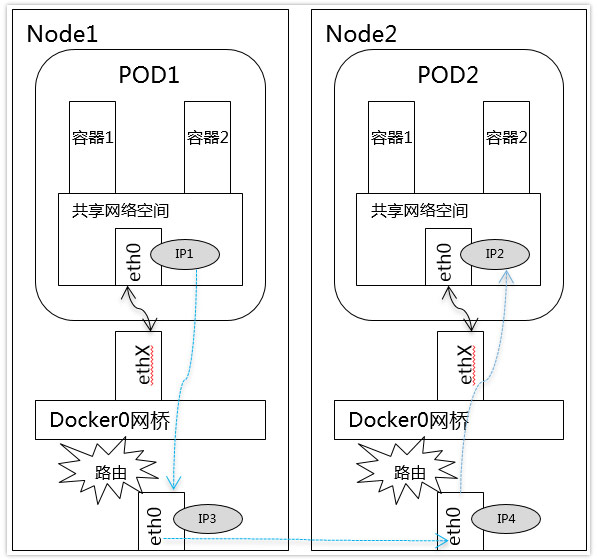
**2、同一Node中Pod间通信：**

同一Node中Pod的默认路由都是docker0的地址，由于它们关联在同一个docker0网桥上，地址网段相同，所有它们之间应当是能直接通信的。



**3、不同Node中Pod间通信：**

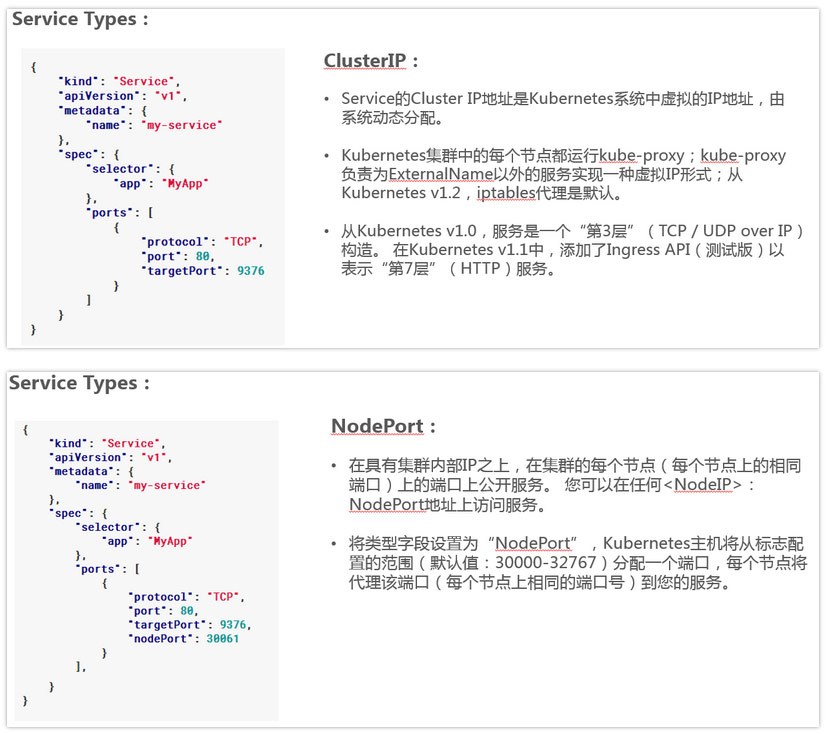
不同Node中Pod间通信要满足2个条件： Pod的IP不能冲突； 将Pod的IP和所在的Node的IP关联起来，通过这个关联让Pod可以互相访问。



**4、Service介绍：**

Service是一组Pod的服务抽象，相当于一组Pod的LB，负责将请求分发给对应的

Pod；Service会为这个LB提供一个IP，一般称为ClusterIP。





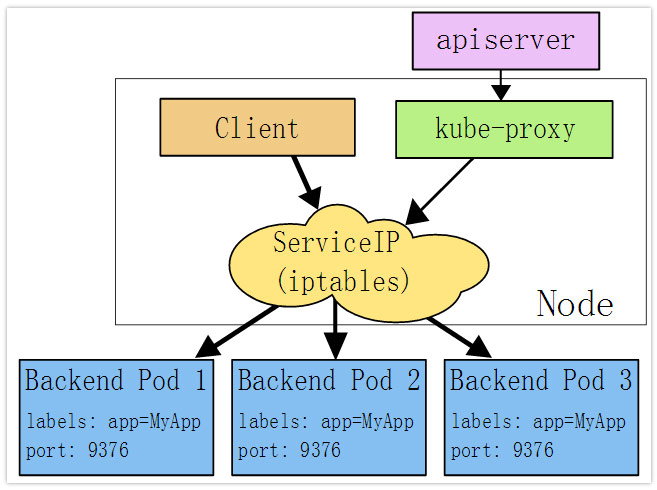
**5、Kube-proxy介绍：**

Kube-proxy是一个简单的网络代理和负载均衡器，它的作用主要是负责Service的实现，具体来说，就是实现了内部从Pod到Service和外部的从NodePort向Service的访问。

**实现方式：**

* userspace是在用户空间，通过kuber-proxy实现LB的代理服务，这个是kube-proxy的最初的版本，较为稳定，但是效率也自然不太高。
* iptables是纯采用iptables来实现LB，是目前kube-proxy默认的方式。

**下面是iptables模式下Kube-proxy的实现方式：**

****

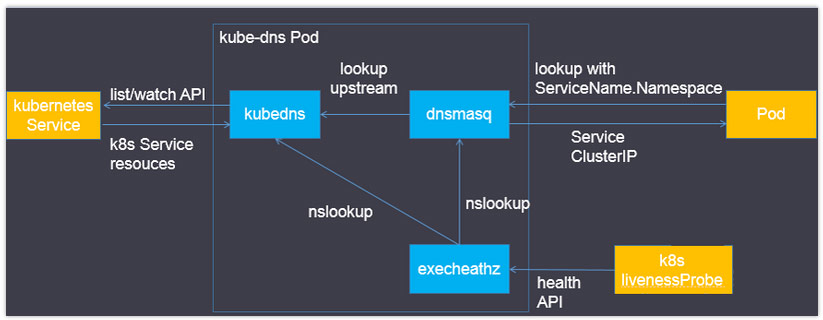
* 在这种模式下，kube-proxy监视Kubernetes主服务器添加和删除服务和端点对象。对于每个服务，它安装iptables规则，捕获到服务的clusterIP（虚拟）和端口的流量，并将流量重定向到服务的后端集合之一。对于每个Endpoints对象，它安装选择后端Pod的iptables规则。
* 默认情况下，后端的选择是随机的。可以通过将service.spec.sessionAffinity设置为“ClientIP”（默认为“无”）来选择基于客户端IP的会话关联。
* 与用户空间代理一样，最终结果是绑定到服务的IP:端口的任何流量被代理到适当的后端，而客户端不知道关于Kubernetes或服务或Pod的任何信息。这应该比用户空间代理更快，更可靠。然而，与用户空间代理不同，如果最初选择的Pod不响应，则iptables代理不能自动重试另一个Pod，因此它取决于具有工作准备就绪探测。

**6、Kube-dns介绍**

Kube-dns用来为kubernetes service分配子域名，在集群中可以通过名称访问service；通常kube-dns会为service赋予一个名为“service名称.namespace.svc.cluster.local”的A记录，用来解析service的clusterip。

**Kube-dns组件：**

* 在Kubernetes v1.4版本之前由“Kube2sky、Etcd、Skydns、Exechealthz”四个组件组成。
* 在Kubernetes v1.4版本及之后由“Kubedns、dnsmasq、exechealthz”三个组件组成。



**Kubedns**

* 接入SkyDNS，为dnsmasq提供查询服务。
* 替换etcd容器，使用树形结构在内存中保存DNS记录。
* 通过K8S API监视Service资源变化并更新DNS记录。
* 服务10053端口。

**Dnsmasq**

* Dnsmasq是一款小巧的DNS配置工具。
* 在kube-dns插件中的作用是：

1. 通过kubedns容器获取DNS规则，在集群中提供DNS查询服务
2. 提供DNS缓存，提高查询性能
3. 降低kubedns容器的压力、提高稳定性

* Dockerfile在GitHub上Kubernetes组织的contrib仓库中，位于dnsmasq目录下。
* 在kube-dns插件的编排文件中可以看到，dnsmasq通过参数–server=127.0.0.1:10053指定upstream为kubedns。

**Exechealthz**

* 在kube-dns插件中提供健康检查功能。
* 源码同样在contrib仓库中，位于exec-healthz目录下。
* 新版中会对两个容器都进行健康检查，更加完善。

**四、Kubernetes网络开源组件**

**1、技术术语：**

**IPAM：**IP地址管理；这个IP地址管理并不是容器所特有的，传统的网络比如说DHCP其实也是一种IPAM，到了容器时代我们谈IPAM，主流的两种方法： 基于CIDR的IP地址段分配地或者精确为每一个容器分配IP。但总之一旦形成一个容器主机集群之后，上面的容器都要给它分配一个全局唯一的IP地址，这就涉及到IPAM的话题。

**Overlay：**在现有二层或三层网络之上再构建起来一个独立的网络，这个网络通常会有自己独立的IP地址空间、交换或者路由的实现。

**IPSesc：**一个点对点的一个加密通信协议，一般会用到Overlay网络的数据通道里。

**vxLAN：**由VMware、Cisco、RedHat等联合提出的这么一个解决方案，这个解决方案最主要是解决VLAN支持虚拟网络数量（4096）过少的问题。因为在公有云上每一个租户都有不同的VPC，4096明显不够用。就有了vxLAN，它可以支持1600万个虚拟网络，基本上公有云是够用的。

**网桥Bridge：**连接两个对等网络之间的网络设备，但在今天的语境里指的是Linux Bridge，就是大名鼎鼎的Docker0这个网桥。

**BGP：** 主干网自治网络的路由协议，今天有了互联网，互联网由很多小的自治网络构成的，自治网络之间的三层路由是由BGP实现的。

**SDN、Openflow：** 软件定义网络里面的一个术语，比如说我们经常听到的流表、控制平面，或者转发平面都是Openflow里的术语。

**2、容器网络方案：**

**隧道方案（ Overlay Networking ）**

隧道方案在IaaS层的网络中应用也比较多，大家共识是随着节点规模的增长复杂度会提升，而且出了网络问题跟踪起来比较麻烦，大规模集群情况下这是需要考虑的一个点。

* **Weave：**UDP广播，本机建立新的BR，通过PCAP互通
* **Open vSwitch（OVS）：**基于VxLan和GRE协议，但是性能方面损失比较严重
* **Flannel：**UDP广播，VxLan
* **Racher：**IPsec

**路由方案**

路由方案一般是从3层或者2层实现隔离和跨主机容器互通的，出了问题也很容易排查。

* **Calico：**基于BGP协议的路由方案，支持很细致的ACL控制，对混合云亲和度比较高。
* **Macvlan：**从逻辑和Kernel层来看隔离性和性能最优的方案，基于二层隔离，所以需要二层路由器支持，大多数云服务商不支持，所以混合云上比较难以实现。

**3、CNM & CNI阵营：**

容器网络发展到现在，形成了两大阵营，就是Docker的CNM和Google、CoreOS、Kuberenetes主导的CNI。首先明确一点，CNM和CNI并不是网络实现，他们是网络规范和网络体系，从研发的角度他们就是一堆接口，你底层是用Flannel也好、用Calico也好，他们并不关心，CNM和CNI关心的是网络管理的问题。

**CNM（Docker LibnetworkContainer Network Model）:**

Docker Libnetwork的优势就是原生，而且和Docker容器生命周期结合紧密；缺点也可以理解为是原生，被Docker“绑架”。

* Docker Swarm overlay
* Macvlan & IP networkdrivers
* Calico
* Contiv
* Weave

**CNI（Container NetworkInterface）：**

CNI的优势是兼容其他容器技术（e.g. rkt）及上层编排系统（Kubernetes & Mesos)，而且社区活跃势头迅猛，Kubernetes加上CoreOS主推；缺点是非Docker原生。

* Kubernetes
* Weave
* Macvlan
* Calico
* Flannel
* Contiv
* Mesos CNI

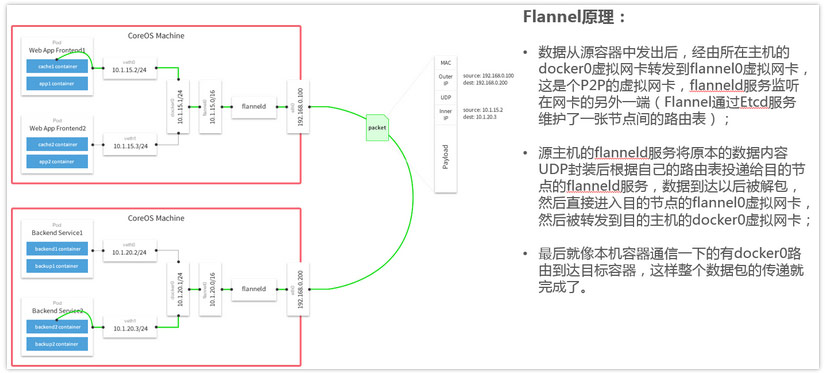
**4、Flannel容器网络：**

Flannel之所以可以搭建kubernets依赖的底层网络，是因为它可以实现以下两点：

* 它给每个node上的docker容器分配相互不想冲突的IP地址；
* 它能给这些IP地址之间建立一个覆盖网络，同过覆盖网络，将数据包原封不动的传递到目标容器内。

**Flannel介绍**

* Flannel是CoreOS团队针对Kubernetes设计的一个网络规划服务，简单来说，它的功能是让集群中的不同节点主机创建的Docker容器都具有全集群唯一的虚拟IP地址。
* 在默认的Docker配置中，每个节点上的Docker服务会分别负责所在节点容器的IP分配。这样导致的一个问题是，不同节点上容器可能获得相同的内外IP地址。并使这些容器之间能够之间通过IP地址相互找到，也就是相互ping通。
* Flannel的设计目的就是为集群中的所有节点重新规划IP地址的使用规则，从而使得不同节点上的容器能够获得“同属一个内网”且”不重复的”IP地址，并让属于不同节点上的容器能够直接通过内网IP通信。
* Flannel实质上是一种“覆盖网络(overlaynetwork)”，也就是将TCP数据包装在另一种网络包里面进行路由转发和通信，目前已经支持udp、vxlan、host-gw、aws-vpc、gce和alloc路由等数据转发方式，默认的节点间数据通信方式是UDP转发。

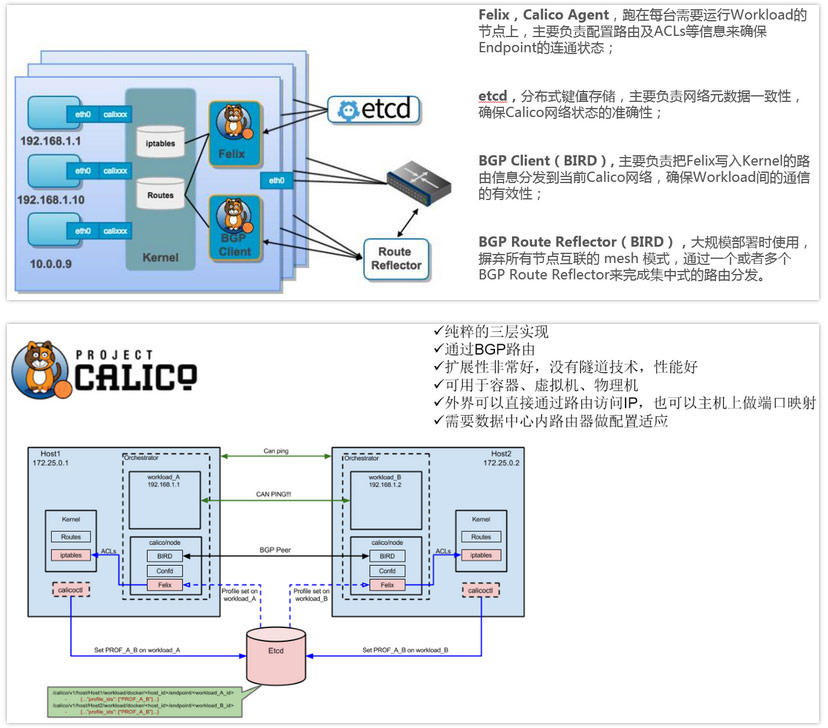


**5、Calico容器网络：**

**Calico介绍**

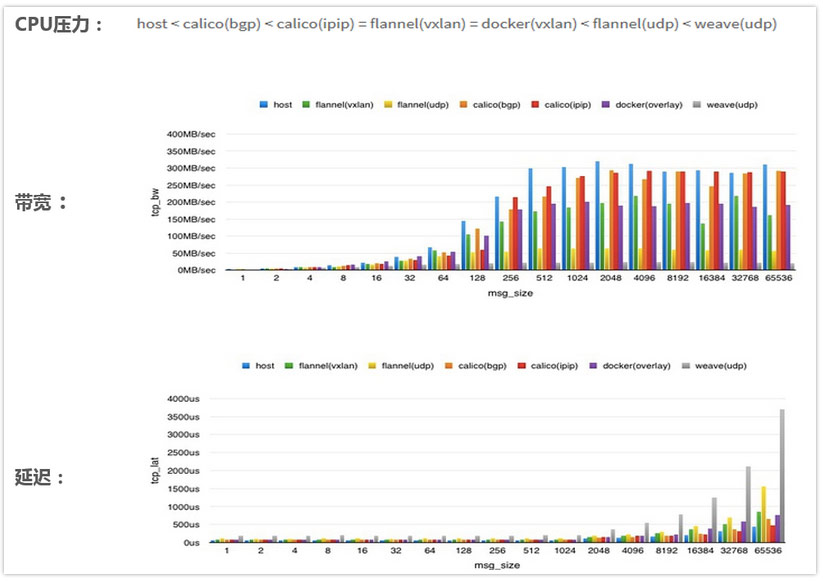
* Calico是一个纯3层的数据中心网络方案，而且无缝集成像OpenStack这种IaaS云架构，能够提供可控的VM、容器、裸机之间的IP通信。Calico不使用重叠网络比如flannel和libnetwork重叠网络驱动，它是一个纯三层的方法，使用虚拟路由代替虚拟交换，每一台虚拟路由通过BGP协议传播可达信息（路由）到剩余数据中心。
* Calico在每一个计算节点利用Linux Kernel实现了一个高效的vRouter来负责数据转发，而每个vRouter通过BGP协议负责把自己上运行的workload的路由信息像整个Calico网络内传播——小规模部署可以直接互联，大规模下可通过指定的BGP route reflector来完成。
* Calico节点组网可以直接利用数据中心的网络结构（无论是L2或者L3），不需要额外的NAT，隧道或者Overlay Network。
* Calico基于iptables还提供了丰富而灵活的网络Policy，保证通过各个节点上的ACLs来提供Workload的多租户隔离、安全组以及其他可达性限制等功能。

**Calico架构图**

****

**五、网络开源组件性能对比分析**

**性能对比分析：**

****

**性能对比总结：**

CalicoBGP 方案最好，不能用 BGP 也可以考虑 Calico ipip tunnel 方案；如果是 Coreos 系又能开 udp offload，flannel 是不错的选择；Docker 原生Overlay还有很多需要改进的地方

最后，再提下我们有容云5.22发布了基于Kubernetes的容器云平台产品UFleet，UFleet采用的是Flannel网络，后续我们将支持Calico网络，如需试用，欢迎联系有容云。

上一篇: [容器编排Kubernetes之kube-dns源码解读](https://www.kubernetes.org.cn/2050.html)下一篇: [魅族容器云平台基于Kubernetes自动化运维实践](https://www.kubernetes.org.cn/2081.html)

标签：[网络](https://www.kubernetes.org.cn/tags/%e7%bd%91%e7%bb%9c)

**相关推荐**

* [kubernetes容器网络接口(CNI) midonet网络插件的设计与实现](https://www.kubernetes.org.cn/2615.html)
* [基于Kubeadm的Flannel分析](https://www.kubernetes.org.cn/2270.html)
* [容器编排之Kubernetes网络隔离NetworkPolicy](https://www.kubernetes.org.cn/1909.html)
* [Kubernetes网络接口(CNI) midonet网络插件设计与实现](https://www.kubernetes.org.cn/1825.html)
* [Kubernetes集群中的网络](https://www.kubernetes.org.cn/1712.html)
* [借助 Calico，管窥 Kubernetes 网络策略](https://www.kubernetes.org.cn/1675.html)
* [使用基于策略的网络扩展Kubernetes Deployments](https://www.kubernetes.org.cn/1501.html)
* [正确的在Kubernetes集群中使用SDN技术方法](https://www.kubernetes.org.cn/1342.html)

登录