doi:10.11887/j.cn.201901020 http://journal.nudt.edu.cn

面向大规模容器集群的网络控制技术*

王宝生,张维琦,邓文平
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:针对容器技术在网络层面缺乏控制的问题,设计一套面向大规模容器集群的网络控制架构,分别从容器集群网络的灵活组网、智能适配以及安全隔离三个方向进行研究,主要解决大规模容器集群部署中的网络适配和隔离控制等关键问题。实验结果表明,设计的网络控制架构可以根据网络特点有针对性地实现虚拟局域网的快速划分、网络节点的稳定迁移和节点通信的精确隔离控制。

关键词:大数据;容器网络控制;灵活组网;智能适配;安全隔离
中图分类号:TP393 **文献标志码**:A **文章编号**:1001-2486(2019)01-142-10

Network control technology for large-scale container clusters

WANG Baosheng, *ZHANG Weiqi*, *DENG Wenping*
(College of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: Aimed at solving the lack of control of container technology at the network level, a network control architecture for large-scale container clusters was designed. The architecture was researched respectively from three aspects (flexible networking, intelligent adaptation and security isolation) in container cluster network, mainly to solve the key issues of large-scale container clusters deployment in the network adapter and isolation control, etc.. The experimental results show that the designed network control architecture can achieve intelligent network adaptation, network node migration, and accurate isolation control of node-to-node communication in virtual local area network.

Keywords: big data; container network control; flexible networking; intelligent control; security isolation

数据中心通过虚拟化技术为云计算构建虚拟资源池。云计算通过数据中心的虚拟资源管理技术即可实现资源自动部署、动态扩展、按需分配等功能,满足用户按需和即付即用的资源获取需求。虚拟化是构建云基础架构不可或缺的关键技术之一。然而,当前虚拟资源管理仍然存在着云计算资源利用率低、应用与平台无法解耦、应用运行环境局限性强、运维人员控制力下降等问题。数据中心的资源虚拟化初始主要由虚拟机实现,随着容器技术的逐渐成熟,虚拟机体量大、部署困难的问题逐渐暴露[1-2]。

随着 Docker[3-5] 等容器化技术的发展和推广,基于容器的资源虚拟化方式正在逐步代替虚拟机,容器技术越来越多地被应用于数据中心以实现大规模集群部署。容器是一种内核虚拟化技术,可以提供轻量级的虚拟化,以便隔离进程和资源,而且不需要提供指令解释机制以及全局虚拟化的复杂操作。容器的轻量级资源管理和秒级启动等特性可以简化管理难度,解决数据中心当前所面临的虚拟资源管理等问题,使用容器代替虚拟机在数据中心实现大规模集群部署已经成为必然趋势[5-9]。

容器技术仍处于发展阶段,在集成部署等方面都有很大的发展空间,尤其是网络层面缺乏系统的管理架构和基本的控制手段[10]。容器的简易部署特性使容器可以快速实现大规模的集群部署,但也相应地增加了容器网络的离散程度,导致网络管理控制困难。容器在集群中不仅需要承担复杂的计算任务,还需要承担不同节点间的网络通信和数据交互,复杂的网络交互和快速变化的网络重构也对网络控制提出了更高的要求[11-13]。实现面向大规模容器集群的网络控制成为数据中心进行容器化变革时亟待解决的问题[14]。

容器网络模型是网络控制面临的首要难题。容器技术发展至今,网络方面发展滞后,网络层面缺乏标准接口,两种后续的网络方案——容器网络模型(Container Network Model, CNM)和容器网络接口(Container Network Interface, CNI)[15-16]

* 收稿日期:2018-01-16
基金项目:国家自然科学基金资助项目(61472438,61602503)
作者简介:王宝生(1970—),男,河北沧州人,教授,博士,博士生导师,E-mail:wangbaosheng@126.com

也难以定义标准的网络接入模型[17-18]。因此,大量的创业公司和开源组织开发了各式各样的网络实现,产生了许多复杂的解决方案,从而增加了网络管理的难度[19-22]。

容器网络的安全与隔离策略是当前面临的另一关键问题。由于容器网络离散程度的提高,传统网络安全策略已经不再适用,容器在网络层面缺乏有效的隔离与控制手段。容器依赖于操作系统存在,虽然可以借用 Linux 系统原有的方案实现安全与隔离,但仍旧缺乏系统完整的安全控制方案。容器网络急需一套完整的安全隔离方式,从而实现对网络的控制,保证容器网络的安全[23-25]。

本文采用基于虚拟可扩展局域网[26-27](Virtual extensible Local Area Network, VxLAN)的 Overlay 组网和基于 macvlan[28-31]的网络模型为基础,分别发扬两者扩展性强和传输性能高的特点,根据不同网络需求进行适配;结合两种网络基本模型,采用机器学习中的逻辑回归模型用于网络模型适配,不仅实现了网络模型的快速匹配,还在一定程度上实现了智能化与自动化,保证了网络模型适配的高效性;针对容器网络系统管理控制方案的缺乏问题,结合现有安全手段,归纳出系统化实现网络安全隔离的方法与手段,在保证基本网络功能的前提下以最小的代价实现网络的隔离与控制。

1　容器网络控制架构

在整个架构中,以网络模型选择为基础,通过服务端实现网络部署的全部过程。其中,Web 界面负责对用户需求进行抽象,并将当前网络状态进行展示;服务端则掌握所有主机以及容器节点的互联互通状态,通过节点反馈的信息绘制网络拓扑;数据库用于记录网络服务历史,并提供镜像管理等服务[32-33]。整体架构如图 1 所示。

整体架构为上中下结构,通过设置独立的管理层,便于实现容器网络的灵活部署。最顶层为 Web 层,用户在此填写个人需求,查看日志,并向服务器发送指令。服务端采用应用程序编程接口(Application Programming Interface, API)调用的方式对网络进行控制,根据 Agent 返回的信息形成整体网络结构,并将相关信息通过 Web 界面向外界展示。最底层为容器所在主机,采用 Agent 组件与上层服务进行信息交互,通过 API 调用实现网络配置、网络状态信息上传等功能。底层容器网络的所有配置信息都来自服务端,服务端可

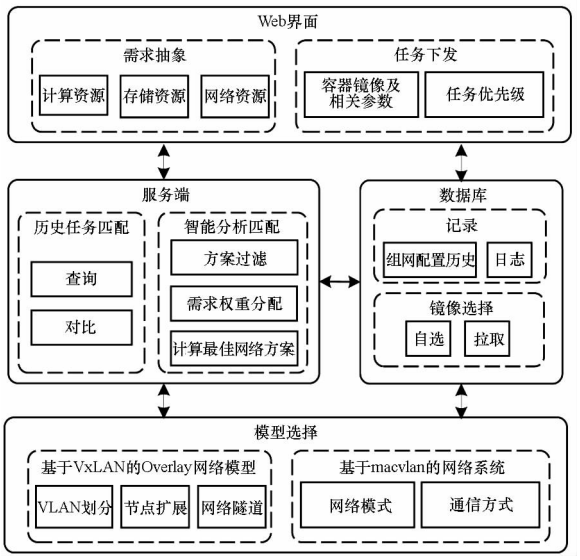


图 1　大规模容器集群网络控制架构

Fig. 1　Large-scale container cluster network control architecture

以根据组网需求,统一进行规划配置,从而最大限度地实现容器组网的灵活性。

网络控制架构采用统一管理的方式,为容器网络智能适配提供模式基础。由 Web 界面获取用户网络需求,根据网络特点生成 json 文件,递交智能适配模块计算最佳适配网络,实现以特定网络需求为牵引、智能适配容器网络模型的目的[17-18]。

隔离和控制是保证网络安全性的重要方式[19]。良好的网络隔离可以确保不同隔离区域的容器或主机不能相互通信,从不同方面进行隔离,减少节点被攻击和干扰的可能性。本架构设计的隔离方式主要涉及以下几个方面:

1)主机间的物理隔离:容器部署在多台主机上,不同主机间不能直接进行通信,实现容器网络间的第一层隔离。

2)容器自身的安全隔离机制:Docker 容器利用 Namespace 和 cgroups 来实现容器隔离技术。其中,Namespace 为隔离技术的第一层,确保 Docker 容器内的进程不可见,并且与外部进程相互隔离,互不影响。cgroups 为 LXC(Linux containers)技术的关键组件,用于实现运行时的资源限制。

3)容器网络虚拟局域网(Virtual Local Area Network, VLAN)划分:处于同一主机的容器可以划分到不同 VLAN 中,不同 VLAN 的容器间不能进行通信。

4)iptables 访问控制:通过 iptables 过滤表对容器和容器间、容器和主机间的通信进行限制,细

致的访问控制规则可以实现网络的精细化隔离控制。

2 基于大规模容器集群的灵活组网技术

2.1 基于 VxLAN 的 Overlay 组网

根据容器技术的特点,计划采用基于隧道技术的大二层网络部署方式。隧道技术属于数据平面的虚拟化,通过对二层帧进行再封装,把底层网络归结在一个平面,底层网络可达之处便是二层网络可及之处,在组网的物理位置上提供了几乎无限的可扩展性,可以充分发扬容器组网扩展性强的优势特点。在现存的多种隧道技术中,本文计划采用以 VxLAN 技术为代表的第四代隧道技术进行网络部署。

VxLAN 技术将隧道设备与虚拟机直连,在物理接入交换机上甚至服务器内部的管理程序中进行部署,形成二层端到端的隧道。通过在接入设备上打通隧道,将 IP 的灵活性用于传输,从而实现了网络的进一步扩展。在容器网络集群中应用 VxLAN 技术,基于已有的服务或 IP 网络,为分散的物理站点提供二层互联功能。不同物理站点可以涵盖多个网段,同一站点中的容器分属不同 VLAN,不同 VLAN 的容器间不能进行通信,示意图如图 2 所示。

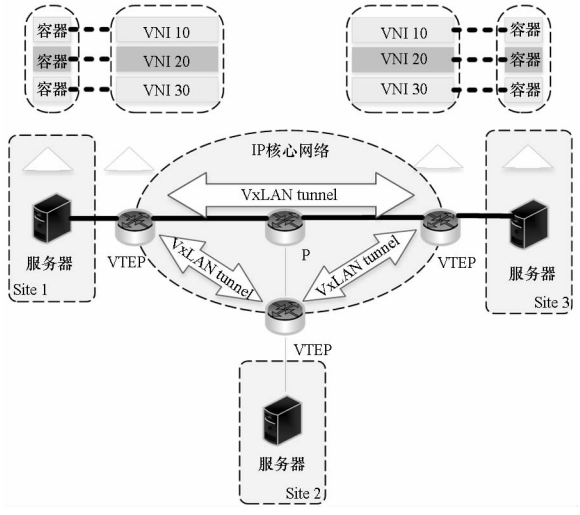


图 2 基于 VxLAN 的容器网络部署示意图

Fig. 2 VxLAN-based container network deployment diagram

在容器网络环境中,将每个服务器分属不同集群网络的容器分配到不同的子网网段,当网络中的信息经过 VxLAN tunnel 传输到达边缘设备 VTEP(VxLAN tunnel end point)后,对数据帧进行识别、转发、封装/解封装等操作,实现网络信息的定向传输和隔离。根据待启动容器当前的网络状态,在容器启动命令中直接添加网络信息,包括所属网段、连接方式以及访问权限等信息,实现网络信息预配置。在确保网络性能的条件下,通过简化容器所占资源,实现容器节点的快速加入和删除。

2.2 基于 macvlan 的容器网络模型

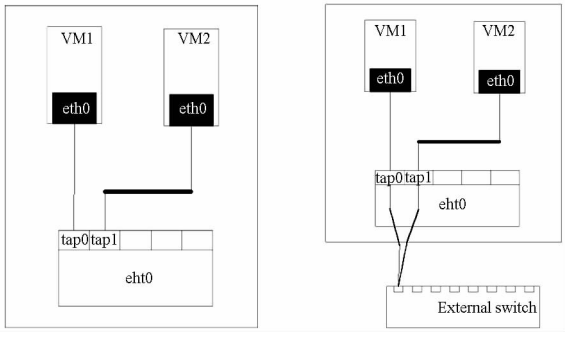
为了充分发挥容器性能,提高网络适配性,本文设计了基于 macvlan 的容器网络模型,在保证网络基本功能的前提下尽可能地提高网络传输效率,保证网络通信质量。macvlan 是 Linux 内核实现的一个虚拟网络设备,可以使一块物理网卡虚拟出多块虚拟网卡,每一块虚拟网卡都拥有不同的媒体访问控制地址(Media Access Control address, MAC)。当物理网卡接收到数据时,根据虚拟网卡 MAC 地址信息将数据转发给相应的网卡。macvlan 设备共有四种模式,其功能和作用介绍如下:

1)bridge 模式:这种模式类似于 Linux bridge。一块物理网卡虚拟出来的所有 macvlan 设备之间可以直接交换数据,如同连在同一个交换机上,结构如图 3(a)所示。

2)虚拟以太端口汇聚器(Virtual Ethernet Port Aggregator, VEPA)模式:当父网卡从子网卡接收到数据时,即使目的地址为同一主机上到虚拟设备,数据仍旧会通过处于外界的上游交换机进行转发,结构如图 3(b)所示。

3)Private 模式:这种模式是一种隔离模式,处在这种模式下的子设备不能与其他子设备之间通信。

4)Passthru 模式:在这种模式下,一块物理网卡只允许拥有一个 macvlan 子设备。



(a) bridge 模式 (b) VEPA 网络模式

(a) bridge mode (b) VEPA network mode

图 3 macvlan 网络模式

Fig. 3 macvlan network mode

在容器网络技术中,macvlan 是与 Docker Overlay 类似的网络方案,是一种支持跨主机方式

的容器网络驱动。macvlan 在进行网络通信时不需要创建网桥,直接通过以太网接口连接到物理网络,在众多容器网络方案中同环境下网络性能最好。另外,macvlan 根据终端设备的 MAC 地址来划分 VLAN,即使用户改变了接入端口,容器也仍然处在原 VLAN 中,便于对容器节点进行灵活的迁移控制。

2.3　节点灵活迁移

通过对容器组网过程进行细化,充分发扬容器精巧、灵活的特点,实现容器在不同网络中快速、稳定的迁移。本文利用虚拟交换机(Open VSwitch, OVS)搭建主机间的网桥,通过 Docker 容器 API 调用的方式实现容器虚拟网卡的灵活添加和删除,并分配相应的 IP 地址,从而实现容器在不同网络中的灵活部署。

在网络中对容器节点进行加入或者退出操作时,不仅需要考虑新网络中网络节点的连通关系,还需要保证原有网络的稳定性。网络中容器主要分为两种,一种为 master 节点,一种为 node 节点。其中,master 节点主要用于整体网络架构的管理,node 节点实现基本的网络功能,所有 node 节点定时向 master 节点发送状态信息,实现网络整体可控。当 node 节点需要退出时,需要先向 master 节点注销;同理,当节点需要加入网络时,也需要在 master 节点进行注册,实现节点的有序加入。

由于容器网络通信模式的不同特点,网络节点的迁移只能针对部分容器。当前,容器网络通信模式主要有 host、bridge、container、none、自定义五种。其中,host 与 container 模式采用共享主机和容器网络的配置方式,限制节点向本机或者容器之外的网络进行迁移,不利于网络的灵活性。所以本文在进行容器网络部署时,主要采用 bridge、none 以及自定义模式,保证容器节点的可迁移性。

3　网络智能适配技术

容器快速启动和易于迁移的特性对网络模型的快速选择提出了更高的要求。

3.1　适配流程

在对历史任务进行检测的基础上进行智能分析匹配,具体步骤为:①提取任务对网络的必要需求,将数据进行分类,整理数据格式;②通过分类器实现网络模型的快速适配;③添加数据进入样本集,重新生成分类器。建立历史信息库,当接收到相同请求时,直接参考历史任务所使用的网络方案,减少计算开销,流程如图 4 所示。

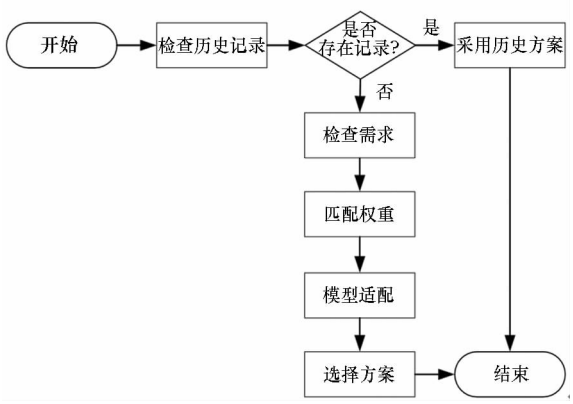


图 4　网络智能选择流程图

Fig. 4　Network intelligent selection flow chart

网络需求信息采集主要针对容器网络中的关键特点信息进行收集,在综合考虑容器网络安全性、连通性和辨识性的前提下设定需求描述,主要包含网络、主句、安全和标签等几个方面。

3.2　Logistic 回归模型

Logistic 回归模型是一种多元统计分析模型,可以完成一组自变量与一个因变量之间的回归统计分析工作,这些自变量可以是连续型数据也可以是离散型数据。Logistic 回归模型需要训练样本和测试样本两种样本,本文初始随机生成 2000 条网络需求特点,并对网络特点进行专家适配,得到训练样本,在训练样本中随机选择 300 条数据作为测试样本,样本示例如表 1 所示。其中,安全性、网络规模、传输效率、扩展性四栏中的数据表示特征权重,网络模型中“0”表示基于 VxLAN 的 Overlay 网络模型,“1”表示基于 macvlan 的网络模型。

表 1　样本数据格式

Tab. 1　Sample data format

安全性	网络规模	传输效率	扩展性	网络模型
6	0	7	2	0
2	8	9	5	1
1	3	4	7	0
5	4	10	7	1
6	0	7	2	0
1	8	5	4	1
4	4	6	10	1
5	4	10	7	0
⋮	⋮	⋮	⋮	⋮

本文的网络适配主要包含基于 VxLAN 的

Overlay 网络模型和基于 macvlan 的网络模型两种,因此采用 Binary Logistic 回归模型。二分类问题的概率与自变量之间的关系图形往往是一个 S 形曲线,采用的 Sigmoid 函数实现逻辑函数的形式为:

$$y = f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

根据离散型随机变量期望值的定义,用 P 代表自变量为 x 时 $y = 1$ 的概率,并采用线性模型进行分析,可得 $P(y = 1) = p$。采用线性模型进行分析,其公式变换如下:

$$P(y = 1 \mid x) = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \tag{2}$$

由于概率 P 和因变量非线性度较高,因此引入 *logit* 变换,使得 $logit(P)$ 与自变量之间存在线性相关的关系,逻辑回归模型定义如下:

$$logit(P) = \ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \tag{3}$$

此时,可计算概率:

$$P = \frac{e^{(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n)}}{1 + e^{(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n)}} \tag{4}$$

用 α 和 β 表示概率,可得:

$$\alpha = \frac{e^{p}}{1 + e^{p}} = \frac{e^{(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n)}}{1 + e^{(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n)}} \tag{5}$$

$$\beta = \frac{1}{1 + e^{p}} = \frac{1}{1 + e^{(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n)}} \tag{6}$$

其中:α 为选择基于 VxLAN 的 Overlay 网络模型的概率;β 为选择基于 macvlan 的网络模型的概率;$b_i(i = 0,1,2,\cdots,n)$ 为基于训练样本得到的回归系数;n 为参与回归分析的自变量的数量;$x_i(i = 0,1,2,\cdots,n)$ 为自变量。

通过随机逻辑回归模型筛选特征的结果如表 2所示。

表 2 随机逻辑回归模型特征筛选

Tab. 2 Stochastic logistic regression model feature screening

安全性	网络规模	传输效率	扩展性
1	0.46	1	0.49

由特征筛选结果可得,安全性与传输效率为网络模型选择的主要影响因子,网络规模与扩展性的影响程度也接近 50%,可以采用这四种特征对最终网络模型进行预测。经过测试,本文通过逻辑回归模型实现的网络模型适配的正确率为 94.6%。

4 基于策略的容器网络隔离控制技术

结合容器自身的资源隔离特点,融入 OVS 虚拟网桥技术,用以搭建容器通信虚拟网桥,在 OVS 网桥基础上使用 VLAN 隔离技术,并基于不同策略利用 iptables 完善容器服务的访问控制机制。通过在网络配置中有针对性地设计网络隔离策略,从而灵活控制容器网络间、容器与主机间、主机间的通断,保证容器网络的安全性。

4.1 基于 iptables 的访问控制机制

在超大规模容器集群网络下,将不同功能、不同应用及不同用户的容器进行逻辑隔离分层的主要方法是对其进行访问控制配置,本节主要说明 iptables 在容器网络隔离分层中的应用。iptables 可以完成封包过滤、封包重定向和网络地址转换(Network Address Translation, NAT)等功能,其架构如图 5 所示。

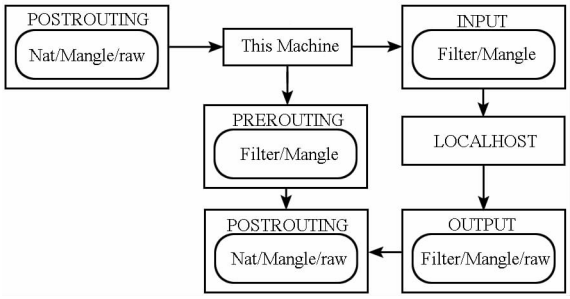


图 5 iptables 架构

Fig. 5 iptables architecture

iptables 传输数据包的过程如下:

1)当一个数据包进入网卡时,首先传入 PREROUTING 链,内核根据数据包目的 IP 判断是否需要转送出去。

2)如果数据包进入本机,则到达 INPUT 链,此次任何进程都会收到它。本机上运行的程序可以发送数据包,这些数据包会经过 OUTPUT 链,然后到达 POSTROUTING 链输出。

3)如果数据包转发出去,并且满足 Filter/Mangle 规则,数据包会经过 PREROUTING 链后到达 POSTROUTING 链输出。

在容器网络中需要有针对性地应用 iptables 过滤策略对网络进行配置,从而实现针对容器的网络隔离方式。本文主要应用 iptables 完成了以下几个方面的网络隔离策略:同一主机内容器间的连通与隔离、容器间跨主机互联互通、容器间跨主机单项通信、容器与外网单项通信等。

4.2 基于 VLAN 的网络隔离技术

基于 VLAN 隔离的访问控制方法在现今的虚拟网络中有广泛的应用。VLAN 是对二层交换机端口上的网络节点的逻辑分段,不受网络用户物

理位置的限制。VLAN 的分组方式较为灵活,可以参考靠容器的作用、服务、结构以及使用的应用程序等特点进行部署。VLAN 在单个交换机甚至多个交换机之间进行灵活的划分,从而解决网络中的域冲突、带宽占用等问题。在容器平台中使用 VLAN 技术,可以增加容器集群的网络连接灵活性,将不同地点、不同网络、不同用户组合在一起,形成一个虚拟的网络环境,构成单个广播域,增加网络隔离性。

VLAN 隔离在容器中的应用不同于传统网络,不同主机内容器所属 VLAN 网络划分成为关键。单个容器在集群网络中的启动与删除十分灵活,可以实现按需部署,从而保证了资源的按需分配,但这也导致容器网络动态性和离散性的增加,容器在 VLAN 中加入和删除的效率也需要得到保证。另外,同一主机内的容器可以分数多个 VLAN 网络,多个主机间的容器也可以分属同一 VLAN,这对 VLAN 隔离控制提出了更高的要求。

如图 6 所示,通过 OVS 网桥可以为容器设置 VLAN,将容器 1 和容器 3 所在 VLAN 的 VLAN ID 设置为 1,容器 2 和容器 4 所在 VLAN 的 VLAN ID 设置为 2。容器 1、3 和容器 2、4 分别属于同一个广播域,建立起完整的虚拟通信环境。此时,容器 1 与容器 2、4 处于隔离状态,但能与容器 3 进行通信。使用 VLAN 可以保证容器之间的隔离性,提高网络利用率,增强容器网络的安全性和保密性。

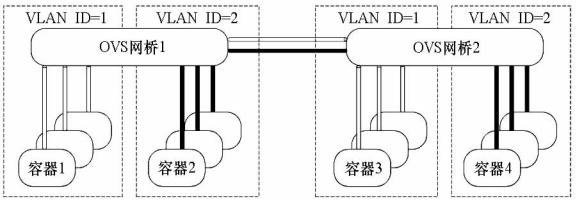


图 6　基于 OVS 网桥的 VLAN 隔离结构图

Fig. 6　VLAN-based isolation OVS bridge structure

5　实验与结果分析

实验主要在华为 FusionServer RH2288 服务器上进行,为保证网络测试的正确性和准确性,本文在 10 台服务器上进行实验验证,每台服务器之间通过光纤进行互联。另外,为保证容器节点基数,在每台服务器上都部署了超过 1000 个容器节点进行测试,基本满足大规模容器网络场景。为排除特殊因素的影响,每项实验数据都是进行超过 10 次测量并筛选掉明显错误项后整理所得。

5.1　容器灵活组网验证

容器灵活组网实验主要针对网络节点灵活可控和高效的组网效率进行。在基于 VxLAN 的 Overlay 网络中进行节点灵活迁移测试,在基于 macvlan 的网络模型中进行组网测试,从而分别突出两种不同网络模型扩展性好和组网效率高的特点。

5.1.1　节点灵活迁移

以基于 VxLAN 的 Overlay 网络为基础网络模型,分别测试 node 节点的灵活迁移和部署效率。在两台服务器上分别部署 OVS,并搭建虚拟网桥,其中两台服务器配置信息如图 7 所示。

```
服务器 1:
    网桥信息:Bridge vxbr: 10.1.2.1
    端口信息:Port "vxlan1"
                Interface "vxlan1"
                    Type: vxlan
            Options: {remote_ip = "202.197.18.35"}
    容器 1 虚拟网卡:10.1.2.3
服务器 2:
    网桥信息:Bridge vxbr: 10.1.2.2
    端口信息:Port "vxlan2"
                Interface "vxlan2"
                    Type: vxlan
            Options: {remote_ip = "202.197.18.34"}
    容器 2 虚拟网卡:10.1.2.4
```

图 7　服务器配置

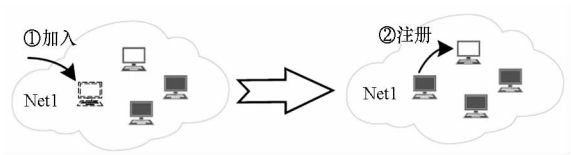
Fig. 7　Server configuration

每台主机都需要配置相互通信的虚拟网桥,并将挑选同处于一个网段的地址作为网关。VxLAN 方式定义在端口信息中,Port 中定义接口名称为 vxlan1 和 vxlan2,Type 设定为 vxlan 方式。在进行数据传送时,通过 remote_ip 信息定义到外部主机,确保信息可以转发到外部物理主机的 IP 地址。在容器中配置虚拟网卡,地址设定需要与虚拟网桥的网关地址相匹配。

容器间跨主机交互数据转发路径如下:当从服务器 1 上的容器 1 向服务器 2 上的容器 2 发送数据时,数据首先从 10.1.2.3 转到主机 1 的 202.197.18.34 地址,而后通过 vxbr 虚拟网桥跨主机传送到 202.197.18.35 地址,最后再转到 10.1.2.4,实现容器间数据的跨主机传输。

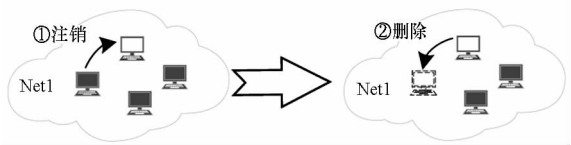
在基础网络平台搭建完成的前提下对网络灵活性进行验证。当容器需要加入到新网络中时,首先需要在 master 节点中进行注册,将任务和地

址等信息进行上传,通过 master 节点对 node 节点信息进行管理,如图 8(a)所示。同理,当容器退出所在网络时,也需要在 master 节点中删除信息,如图 8(b)所示。当 node 节点在两个网络中进行迁移时,需要先在原有网络中进行注销,然后再添加进入新网络,node 节点迁移情况如图 8(c)所示。同一 node 节点允许同时处于不同的网络中,即容器可以在已经加入一个网络的情况下再次加入另一个网络中,如图 8(d)所示。



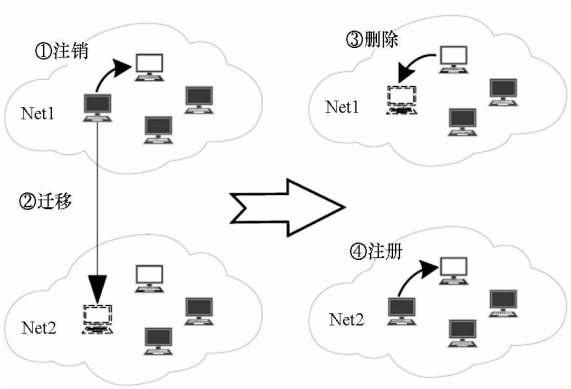
(a) 节点加入

(a) Node join



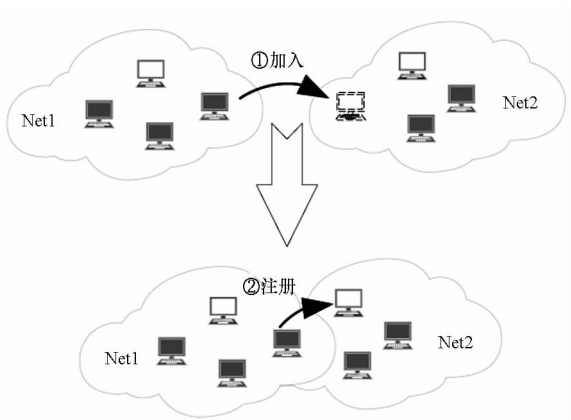
(b) 节点删除

(b) Node delete



(c) 节点迁移

(c) Node migration



(d) 节点重复加入网络

(d) The node repeatedly joins the network

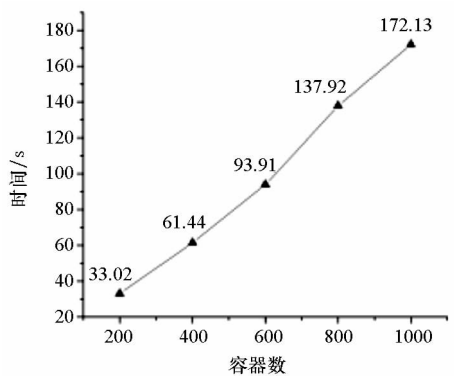
图 8 节点灵活迁移

Fig. 8 Nodes flexible migration

通过细化网络节点的迁移过程,逐步进行加入、删除、迁移、重复加入等动作,从而实现网络节点灵活、有序的迁移。利用 master 节点对网络进行基本控制,明确了 node 节点的加入、删除步骤,增强了对网络整体架构的控制。但是,当网络规模到达一定程度的时候,单一控制节点难以对网络进行合理控制,可以采用多节点管理的方式解决这一问题。

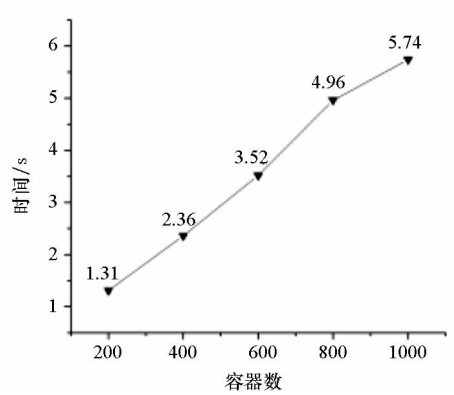
5.1.2 组网效率测试

网络部署效率是灵活组网的基础,通过在基于 macvlan 网络模型中测试不同规模网络的部署时间,获取网络部署的基础数据,为大规模容器网络部署提供参考。由于 busybox 镜像在保证较小的存储占用的情况下集成了 Linux 中几百个常用的命令及工具,便于实现网络基本功能,所以本试验选用 docker. io/busybox:latest 为基础容器镜像进行组网性能测试。在进行测试时,分别在 macvlan 网络模型中创建 200,400,600,800,1000 个容器进行组网,测试不同规模容器的组网时间。以容器规模数目为主要参考绘制散点图,如图 9 所示。



(a) 创建时间

(a) Create time



(b) 删除时间

(b) Delete time

图 9 不同规模容器组网效率

Fig. 9 Container under different sizes

由测试结果可以看出,当没有超过主机处理极限的前提下,容器启动和删除所用时间与容器数目成正相关,容器创建数目的多少不影响后续容器的启动与删除时间。在当前环境下,容器平均创建时间均在 200 ms 以内。

5.2　网络智能适配测试

在服务器中配置网络基本环境,对网络智能适配技术进行测试。在主机中设计好两种网络模型的适配方式,根据用户需求,采用逻辑回归模型计算出最佳匹配后直接应用网络模型,实现网络的快速智能部署。

通过 2000 条训练样本由逻辑回归模型生成网络模型的分类型后,利用 300 条测试样本对分类器进行性能测试,测试结果表现为准确率、精确率、召回率以及综合评价指标。其中,准确率是分类器预测正确性的比例,但是并不能分辨出假阳性错误和假阴性错误;精确率是指分类器预测出网络模型与实际相匹配的比例;召回率也叫作灵敏度,在本例中表示网络模型被分类器正确找出来的比例;综合评价指标能反映分类器整体的性能特点。测试数据如表 3 所示。

表 3　性能测试指标

Tab. 3　Performance test indicators

项目	数值
准确率	0.947 01
精确率	0.945 07
召回率	0.928 90
综合评价指标	0.936 57

由测试指标可得,分类器对网络模型的选择正确率达到 90% 以上,可以完成基本的网络选型。通过受试者工作特征(Receiver Operating Characteristic, ROC)曲线以可视化的方式进一步反应分类器效果。ROC 曲线对分类比例不平衡的数据集不敏感,显示的是对超过限定阈值的所有预测结果的分类器效果,如图 10 所示。

图 10 中,横坐标误警率表示所有阴性样本中分类器识别为阳性的样本所占比例,*AUC* 是 ROC 曲线下方的面积,它把 ROC 曲线变成一个值,表示分类器随机预测的效果,由图 10 可见分类器预测效果可以达到 99%。由此可见,采用逻辑回归方式生成的分类器可以有效地实现网络模型的快速智能适配,节省网络适配时间,减少适配随机性,提高网络适配效率。

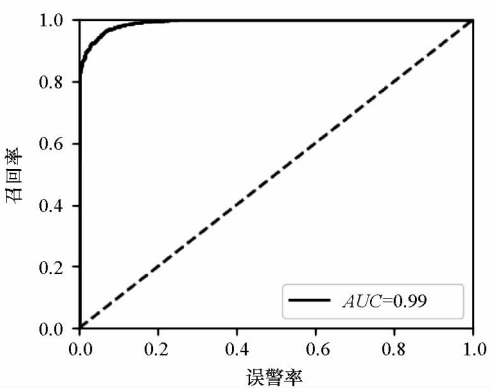


图 10　分类器 ROC 曲线图

Fig. 10　Classifier ROC curve

5.3　容器网络隔离测试

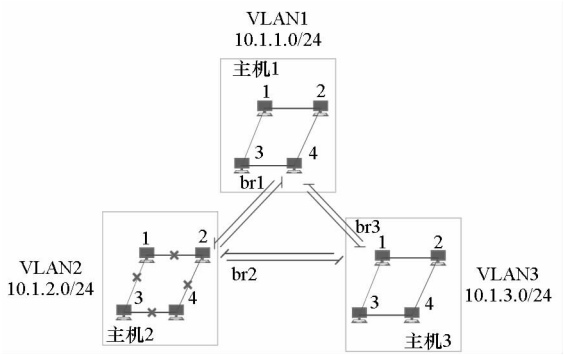
本实验以 OVS 网桥为主要连通手段,应用 VxLAN 网络隧道协议,在此基础上测试 VLAN 隔离和 iptables 网络策略隔离功能。其中 VLAN 划分主要由 OVS 进行配置,通过分配不同网段,实现网络间的地址隔离。使用 Linux iptables 建立过滤表,有针对性地对网络中的数据流进行截断,实现网络通信链路的精确阻断。简单情况下,通过 docker network create 命令即可创建子网,直接实现 VLAN 的划分,在全局网络规则中也可通过 iptables 的 filter 和 nat 表,限制不同网络间的通信,部分隔离策略如表 4 所示。

表 4　网络隔离策略

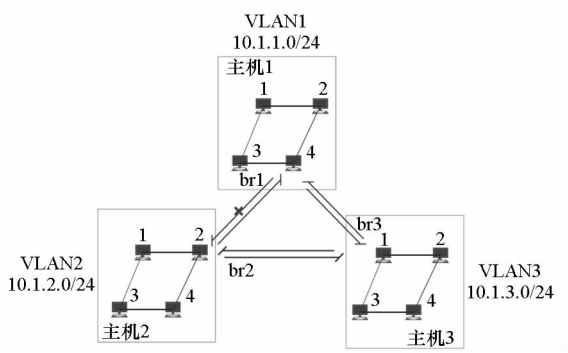
Tab. 4　Network isolation strategy

序号	策略	内容
1	同一 VLAN 中网络阻断	– – subnet 192.168.200.0/24 "com.docker.network.bridge.enable_icc" = "false
2	VLAN 间单向阻断	DROP all from br1 to br2
3	VLAN 间双向阻断	DROP all between br1 & br2
4	通过 4:全网阻断	DROP all
⋮	⋮	⋮

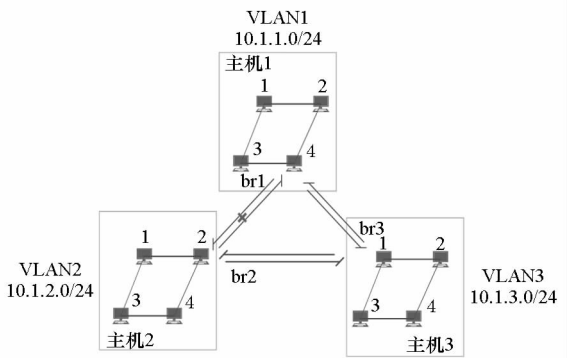
对表 4 中已经列举出来的几种基础策略进行测试,并以网络拓扑图的形式进行展示,如图 11 所示。通过定义过滤表的方式能实现对网络产生最小影响的情况控制网络的通断,从而对全局网络进行控制。单条过滤表策略,不仅可以控制单条链路的通信,也可以实现整个网段的隔离控制,如通过定义 enable_icc 为 false,即可将整个子网



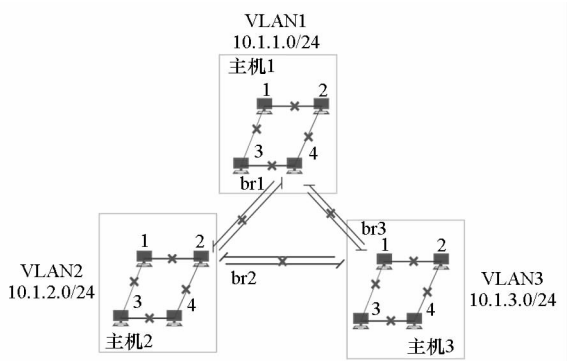
(a) 同一 VLAN 阻断
(a) Same VLAN blocking



(b) VLAN 间单向阻断
(b) Unidirectional blocking between VLANs



(c) VLAN 间双向阻断
(c) Bidirectional blocking between VLANs



(d) 全网阻断
(d) Blocking of the entire network

图 11 网络隔离测试
Fig. 11 Network isolation test

中的网络全部阻断,如图 11(a)所示;若需要单项阻断网桥,只需要将这一方向的数据包丢弃,双向阻断即丢弃网桥中的所有数据包,如图 11(b)、图 11(c)所示;全网阻断则只需阻断所有网络通信即可,如图 11(d)所示。

网络隔离策略灵活多变,不仅可以将单个网络进行隔离,还能实现全网的数据传输。所有的阻断策略均记录在数据表中,即易于后期查询和修改,也便于对新型网络进行配置。本文设计的网络隔离方案,可以在对网络通信影响最小的情况下实现网络的基本隔离控制,但由于配置策略较为细致,合理的配置网络规则需要对网络通信规则有一定的掌握。

6 结论

通过对大规模集群环境下容器网络技术的发展现状进行分析,探讨分析了容器网络控制技术在当前大规模集群中的重要作用。根据对容器网络和传统网络进行对比,分析出容器网络控制应该重点突破灵活组网、智能适配以及安全隔离三个方面。设计并测试网络节点迁移方案,通过细化迁移过程,逐步进行加入、删除、迁移、重复加入等动作,实现了网络节点的灵活、有序迁移。网络智能适配主要实现了网络模型的快速智能匹配,安全隔离模块则通过 VLAN 划分和添加过滤策略的方式实现了对容器网络的安全控制。未来计划逐步完成容器网络控制架构的搭建,开发专用的控制平台,为应用开发人员、系统管理员以及相关研究人员在大规模容器集群上更好地部署和控制容器网络提供指导。

参考文献(References)

[1] Dua R, Raja A R, Kakadia D. Virtualization vs containerization to support PaaS[C]//Proceedings of IEEE International Conference on Cloud Engineering, 2014: 610 - 614.

[2] Pahl C. Containerization and the PaaS cloud[J]. IEEE Cloud Computing, 2015, 2(3): 24 - 31.

[3] Merkel D. Docker: lightweight linux containers for consistent development and deployment[J]. Linux Journal, 2014, 239: 2.

[4] Fink J. Docker: a software as a service, operating system-level virtualization framework[J]. Code4Lib Journal, 2014.

[5] 李珂, 潘峰, 王德奎, 等. 一种基于 Docker 事件自动化配置 Docker 容器同主机网络 IP 的方法: CN106161104A[P]. 2016 - 11 - 23.
LI Ke, PAN Feng, WANG Dekui, et al. A method for automating the configuration of Docker containers with host network IP based on Docker events: CN106161104A [P]. 2016 - 11 - 23. (in Chinese)

[6] 王建飞, 李岩, 刘金国, 等. 一种解决 docker 容器启动并发瓶颈的方法:CN105824688A[P]. 2016 - 08 - 03.
WANG Jianfei, LI Yan, LIU Jinguo, et al. A method to solve the concurrency bottleneck of the docker container startup: CN105824688A[P]. 2016 - 08 - 03. (in Chinese)

[7] 齐勇. 基于 docker 的网络服务质量控制器的设计与实现[D]. 济南: 山东大学, 2015.
QI Yong. Design and implementation of network service quality controller based on docker[D]. Jinan: Shandong University, 2015. (in Chinese)

[8] Anderson C. Docker [software engineering] [J]. IEEE Software, 2015, 32(3): 102 - c3.

[9] 董博, 王雪, 索菲, 等. 基于 Docker 的虚拟化技术研究[J]. 辽宁大学学报(自然科学版), 2016, 43(4): 327 - 330.
DONG Bo, WANG Xue, SUO Fei, et al. Research on virtualization technology based on Docker[J]. Journal of Liaoning University (Natural Sciences Edition), 2016, 43(4): 327 - 330. (in Chinese)

[10] Joy A M. Performance comparison between Linux containers and virtual machines [C]//Proceedings of International Conference on Advances in Computer Engineering and Application, 2015.

[11] Xavier M G, Neves M V, Rossi F D, et al. Performance evaluation of container-based virtualization for high performance computing environments [C]//Proceedings of 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2017: 233 - 240.

[12] Bhimani J, Yang J, Yang Z, et al. Understanding performance of I/O intensive containerized applications for NVMe SSDs [C]//Proceedings of 35th International Performance Computing and Communications Conference, 2017.

[13] Boettiger C. An introduction to Docker for reproducible research [J]. ACM SIGOPS Operating Systems Review, 2015, 49(1): 71 - 79.

[14] Felter W, Ferreira A, Rajamony R, et al. An updated performance comparison of virtual machines and Linux container[C]//Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software, 2015: 171 - 172.

[15] de Bruijn N. eBPF based container networking[Z]. Holland: University of Amsterdam, 2017.

[16] DockerInfo. Container network spotlight: CNM and CNI[EB/OL]. [2017 - 12 - 15](2016 - 11 - 24). http://www.dockerinfo.net/3772.html.

[17] Kratzke N. About microservices, containers and their underestimated impact on network performance[J]. arXiv e-print, 2017: 1710.04049.

[18] Bernstein D. Containers and cloud: from LXC to Docker to Kubernetes[J]. IEEE Cloud Computing, 2014, 1(3): 81 - 840.

[19] Morabito R. A performance evaluation of container technologies on internet of things devices[J]. arXiv, 2016: 1603.02955.

[20] 张青, 刘剑, 朱晓民. 面向 Docker 的覆盖网络搭建探究[J]. 电信工程技术与标准化, 2015(9): 74 - 77.
ZHANG Qing, LIU Jian, ZHU Xiaomin. Research of Docker overlay network [J]. Telecommunication Engineering & Technology, 2015(9): 74 - 77. (in Chinese)

[21] 肖俊. 基于 Docker 的跨主机容器集群自动伸缩设计与实现[D]. 西安: 西北大学, 2015.
XIAO Jun. Automatic docking design and implementation of cross-host container cluster based on Docker[D]. Xi′an: Northwest University, 2015. (in Chinese)

[22] 赵星月. 基于 Docker 容器技术的 Host 网络通信模式研究[J]. 科学与财富, 2017(6): 30 - 31.
ZHAO Xingyue. Host network communication mode based on Docker container technology [J]. Science & Wealth, 2017(6): 30 - 31. (in Chinese)

[23] Bui T. Analysis of Docker security [J]. arXiv, 2015: 1501.02967.

[24] Dusia A, Yang Y, Taufer M. Network quality of service in Docker containers [C]//Proceedings of IEEE International Conference on Cluster Computing, 2015: 527 - 528.

[25] 蔡志强. 基于 Docker 技术的容器隔离性分析[J]. 电子世界, 2017(17): 195.
CAI Zhiqiang. Analysis of container isolation based on Docker technology [J]. Electronic World, 2017 (17): 195. (in Chinese)

[26] Mahalingam M, Dutt D, Duda K, et al. Virtual extensible local area network (VxLAN): a framework for overlaying virtualized layer 2 networks over layer 3 networks: RFC 7348[R]. USA: Internet Engineering Task Force, 2014.

[27] Zhang H. System and method for VxLAN inter-domain communications: US 20140146817 A1[P]. 2014 - 05 - 29.

[28] Cizixs. Linux network virtualization: macvlan [EB/OL]. [2017 - 11 - 20](2017 - 02 - 14). http://cizixs.com/2017/02/14/network-virtualization-macvlan.

[29] 杨鑫, 吴之南, 钱松荣. 基于 Macvlan 的 docker 容器网络架构[J]. 微型电脑应用, 2016, 32(5): 58 - 60.
YANG Xin, WU Zhinan, QIAN Songrong. Docker container network architecture based on Macvlan[J]. Microcomputer Applications, 2016, 32(5): 58 - 60. (in Chinese)

[30] 冯明振. 基于 Macvlan 的 Docker 容器网络模型的设计与实现[D]. 杭州: 浙江大学, 2016.
FENG Mingzhen. Design and implementation of Docker container network model based on Macvlan[D]. Hangzhou: Zhejiang University, 2016. (in Chinese)

[31] 李巍, 赵永彬, 王鸥, 等. 基于 Macvlan 的 Docker 容器网络架构研究[J]. 机械设计与制造, 2017(5): 270 - 272.
LI Wei, ZHAO Yongbin, WANG Ou, et al. Research on Docker container network architecture based on Macvlan[J]. Mechanical Design & Manufacture, 2017(5): 270 - 272. (in Chinese)

[32] Pahl C, Lee B. Containers and clusters for edge cloud architectures—a technology review[C]//Proceedings of 3rd International Conference on Future Internet of Things and Cloud, 2015: 379 - 386.

[33] Anderson J, Hu H X, Agarwal U, et al. Performance considerations of network functions virtualization using containers[C]//Proceedings of International Conference on Computing, Networking and Communications, 2016.