文章编号：2095-2163(2023)01-0178-05 中图分类号：TP316.8 文献标志码：A

基于 SR-IOV 的 Docker 网络架构研究

梁克会，董 龙，洪 华
（中国银联股份有限公司，上海 201201）

摘 要：Docker 作为一个轻量级资源虚拟化解决方案，有易扩展、秒级启动、灵活的弹性伸缩等特点，受到越来越多企业的青睐，是云计算领域构建私有云的主流方案之一。Docker 容器在网络架构方面存在短板，虽然存在多种网络解决方案，但是无法同时解决网络资源隔离和性能损耗的问题，尤其是对于网络性能和可用性敏感的应用。本文提出基于单根 I/O 虚拟化技术 SR-IOV（Single Root I/O Virtualization）的 Docker 网络架构。实验结果表明，网卡虚拟化方案无需打开网卡的混杂模式（Promiscuous Mode）就能够解决网络资源隔离的问题，同时还具有与物理网卡基本同等的性能和稳定性，适合在生产环境大规模使用。

关键词：云计算；容器；网卡虚拟化；SR-IOV

Docker container network architecture research based on SR-IOV

LIANG Kehui, DONG Long, HONG Hua
（China UnionPay Co.,Ltd., Shanghai 201201, China）

【Abstract】 As a lightweight resource virtualization solution, Docker has the characteristics of easy expansion, second-level startup, and flexible elastic scaling. It is favored by more and more enterprises and is one of the mainstream solutions for building private clouds in the field of cloud computing. Docker containers have shortcomings in network architecture. Although there are various network solutions, they cannot solve the problems of network resource isolation and performance loss at the same time, especially for applications that are sensitive to network performance and availability. This paper proposes a Docker network architecture based on the single root I/O virtualization technology SR-IOV (Single Root I/O Virtualization). The experimental results show that this network card virtualization solution can solve the problem of network resource isolation without turning on the promiscuous mode of the network card. At the same time, it also has basically the same performance and stability as the physical network card, which is suitable for large-scale use in production environments.

【Key words】 cloud computing; Docker; network interface card virtualization; SR-IOV

0 引 言

Docker 是一个基于 LXC（Linux Container）技术的开源容器引擎，具有跨平台、移植性强和快速启动等优点[1]。开发人员可以使用 Docker 打包应用程序并将所需的依赖项运行到便携式容器中，使得每个 Docker 应用都拥有独立的 CPU、存储设备、内存以及网络空间。2014 年 12 月，Docker 公司发布了基于 Go 语言开发的原生态容器集群管理工具 Swarm，用来对集群中的 Docker 镜像和容器进行统一的管理[2]。与 Kubernetes 工具相比较，Swarm 可以满足容器的在线扩缩容，尤其适合有状态的容器管理[3]。Swarm 工具的发布促进了 Docker 在集群中的运用，越来越多的公司利用 Docker 和 Swarm 工具部署私有的 PaaS（平台即服务），提供大规模基于容器的服务[4]。随着越来越多的应用部署在 Docker 容器中，对于 Docker 网络通信的需求也越发复杂[5]。目前 Docker 支持的网络架构，在一个节点上部署多个容器时无法同时满足网络隔离、端口不冲突以及较好的性能和可管理性等目标，需要研究更加有效的架构。

1 相关技术

1.1 Docker 网络框架

Linux 内核中提供 6 种资源隔离技术，包括分时系统（UNIX Time-sharing System）命名空间，进程

作者简介：梁克会（1984-），男，硕士，工程师，主要研究方向：云计算、数据库云和容器；董 龙（1980-），男，学士，工程师，主要研究方向：云计算、数据库云和容器；洪 华（1980-），男，学士，工程师，主要研究方向：云计算、大数据技术。

收稿日期：2022-08-22

(process ID)命名空间,进程间通信(Inter-Process Communication)命名空间,挂载(mount)命名空间,网络(network)命名空间和用户(USER)命名空间。Docker 使用 Linux 网络命名空间作为网络实现基础,网络命名空间主要提供了网络资源的隔离,包括网络设备、IPv4 和 IPv6 协议栈、IP 路由表、防火墙等网络资源。Docker 实现了 4 种网络模式满足不同的场景,供用户选择[6]。

host 模式:容器和宿主机共用一个网络命名空间,容器没有独立的网络资源。这种方式可以很好的解决容器与外界的通信问题,但是没有隔离性,同时还会引发网络资源冲突。

container 模式:container 模式与 host 模式类似,不同的是这种模式第二个容器共享另外一个容器的网络命名空间,而不是宿主机的。这种模式同样是降低了容器间网络的隔离性,引发网络资源冲突。

bridge 模式:容器默认的网络模式,Docker 在启动的时候会创建名称为 docker0 的网桥,然后使用虚拟设备接口(veth pair)连接到网桥上,这种模式下容器拥有独立的网络命名空间。容器会通过 iptables NAT 将容器 IP 地址映射出去,网桥负责转发数据帧,使得容器通过宿主机与外界通信。Linux 网桥使用灵活,是目前业界主流,但这种用软件封装的模式导致网络性能下降,架构复杂度高,出现问题排错困难。

none 模式:容器拥有独立的网络命名空间,需要用户为容器添加网卡、配置 IP 地址、路由等信息。这种模式给了用户最大的自由度,但是需要额外的网卡以及配置容器才能与外界通信。

1.2　SR-IOV 技术

SR-IOV 是 PCI-SIG 定义的 IO 设备硬件虚拟规范,是一种基于硬件的辅助虚拟化 I/O 技术[7]。SR-IOV 标准允许在虚拟机或者容器之间高效共享 PCIe(Peripheral Component Interconnect Express,快速外设组件互连)设备,并且是在硬件中实现的。SR-IOV 具有两种功能:物理功能(Physical Function,PF)和虚拟功能(Virtual Function,VF),PF 包含 SR-IOV 功能结构,用于管理 SR-IOV 功能;PF 是全功能的 PCIe 功能,可以像其他任何 PCIe 设备一样进行发现、管理和处理。PF 拥有完全配置资源,可以用于配置或控制 PCIe 设备。VF 是一种轻量级的 PCIe 功能,拥有独立的配置空间,并具有数据通信的关键设备资源,如数据缓冲区、DMA 通道等,非关键的设备资源则与其他 VF 共享。

SR-IOV 是一种基于硬件的网卡虚拟化解决方案,VF 设备可以直接访问寄存器,具有较好的性能和稳定性[8]。同时 SR-IOV 还能够减少使用交换机数量、简化布线工作,节省数据中心的成本和能耗[9]。SR-IOV 通过硬件辅助实现不同对象间数据隔离,提升数据安全保护级别。目前很多主流的中高端网卡都支持 SR-IOV 技术,如 Intel 82576 网卡、I350 网卡、82599 网卡、X540 网卡等。

2　基于 SR-IOV 的 Docker 网络

Docker 实现的网络模式都存在一定的问题,在数据库等对隔离性、性能以及网络延时等都敏感的场景无法大规模使用。文献[10]提出将 Macvlan 应用在 Docker 中,弥补 Docker 网络本身功能缺陷,同时提升数据传输的效率,是一种较好的 Docker 网络解决方案。Macvlan 是利用 Linux 内核实现的虚拟网络设备,将物理网卡虚拟出多个虚拟网卡,并保证虚拟网卡都具有独立的 MAC 地址。当物理网卡接收到数据后,会根据 MAC 地址,将其发送到相应的虚拟网卡上。Macvlan 优点是性能优异,因为无须端口映射或者额外桥接,可以直接通过主机接口(或者子接口)访问容器接口[11]。但是 Macvlan 的缺点是需要将网卡设置为混杂模式,混杂模式下采用以太网广播信道争用的方式,网卡将接收所有经过的数据包,而不管是不是发送给自己的,并传递给上层应用,这很难被允许,尤其是在公有云的场景下。为解决上述问题,本文提出在 Docker 网络中使用 SR-IOV 技术,为容器分配独立的 VF,并由用户为容器配置地址,配置过程无需将网卡设置为混杂模式。

为满足关键业务高可用和高性能的要求,通常服务器上通过两块或者多块网卡进行绑定。绑定是一种 linux 系统下的网卡绑定技术,可以把服务器上 n 个物理网卡在系统内部抽象(绑定)成一个逻辑上的网卡,能够提升网络吞吐量、实现网络冗余、负载等功能。

本文提出的基于 SR-IOV 的 Docker 网络架构如图 1 所示。宿主机两块物理网卡,分别连接 AB 两路交换机,启用 SR-IOV 技术后虚拟出多个虚拟网卡。虚拟网卡命名以 VF 开头,数字中第一位代表物理网卡序号,后面数字代表 VF 序号,如物理网卡 1 的第一个 VF 命名为 VF11,物理网卡 2 的第 63 个 VF 命名为 VF263。使用物理网卡 1 和 2 中对应序号的 VF 进行绑定,通过抽象形成若干个逻辑

bond 网卡,命名为 cbond 和序号,如 VF11 和 VF21 通过绑定后的逻辑网卡命名为 cbond1。在容器调度阶段为容器分配未使用的逻辑网卡,并且按照需求设置 IP 地址以及 VLAN ID。

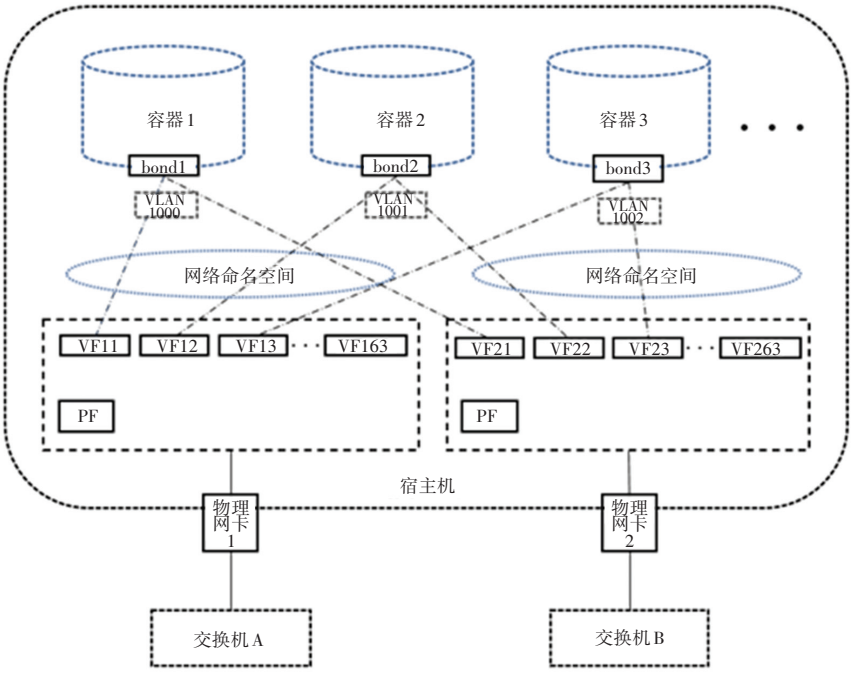

图 1 基于 SR-IOV 的容器网络架构

Fig. 1 Container network architecture base on SR-IOV

基于 SR-IOV 的容器网络配置包括宿主机 SR-IOV 功能启用、网卡虚拟化、VF 双网卡绑定以及容器配置虚拟化网卡过程,具体流程如下:

Step 1 进入 BIOS 设置,找到 Advance -> PCI Subsystem -> SR-IOV Support 选项,选择 Enabled,启用 SR-IOV 功能。同时找到 Advanced-secure Virtual Machine Mode,选择为 Enabled,启动 VT-d 技术;

Step 2 进入 ESXi 硬件配置页面,找到需要进行配置的 SR-IOV 网卡,设置 Virtual functions 数量为 64,保存配置之后重新启动;

Step 3 找到需要虚拟化的网卡 eth0,执行:echo 63 > /sys/class/net/eth0/device/sriov_numvfs,虚拟出 63 个 VF,并按照命名规则对 VF 进行命名。如果有多块物理网卡,依次执行网卡虚拟化;

Step 4 执行双网卡绑定,将不同物理网卡虚拟化的 VF 进行 bond,采用负载均衡模式;

Step 5 容器创建时设置网络模式为 none。如创建一个 mysql 数据库容器,并且容器网络模式设置为 node,则执行:docker run --name mysql -d mysql:5.7.19 --net=none;

Step 6 根据上一步骤创建的容器 ID 号,为容器配置一个逻辑网卡,命令如下:ip link set cbond1 netns $container_id。同时将这个逻辑设备在容器内进行重命名,执行命令如下:ip netns exec $container_id ip link set cbond1 name eth0;

Step 7 对于容器内的网络设备配置 ip 地址,执行命令如下:ip netns exec $container_id ip addr add ipv4/ipv4_prefix dev eth0,如果需要配置 IPv6 地址方法类似,此处不在赘述。

3 实验与结论

3.1 实验环境

本文实验采用 Intel Xeon Gold 5118,CPU 主频 2.3 GHz,内存 256 GB,内存频率 1333 MHz,配置 2 块 Intel 82599 10-Gigabit 网卡。使用 3 台服务器搭建一个 Swarm 集群,验证本文提出的容器网络方案可行性以及性能;操作系统版本为 CentOS 7.5,Swarm 版本为 1.2.9,Docker 版本为 1.13.1,3 台服务器中一台为 Swarm 管理节点,其余两台为工作节点。

3.2 实验结果与分析

通过实验环境验证,使用本方案可以为容器配置不同的逻辑网卡,实现为容器配置独立的网络。同时在关闭网卡混杂模式的情况下,实现容器间网络隔离,方案可行。

为模拟真实的业务场景,验证容器跨服务器访问的网络性能,使用本方案在两个工作节点各创建

5 个容器，并为容器配置不同的 IP 地址和 vlan id。在容器中运行 iperf3 网络性能评测工具，针对 TCP 或 UDP 的传输，测试其网络性能。测试中将其中 1 个容器定义为 iperf3 服务端，其余的容器定义为 iperf3 客户端。通过客户端向服务端发送大小不同的数据包，测试其 TCP 吞吐量（Throughput）、UDP 网络延迟（Latency）和 UDP 丢包率（Packet loss rate），并以此为依据进行网络性能的测定并与物理网卡（NIC）以及 macvlan 方案进行对比。

3 种网络方案 TCP 吞吐量实验结果如图 2 所示，可以看到 SR-IOV 方案在不同消息窗口大小场景下与物理网卡和 macvlan 具有基本相同的吞吐量。随着消息窗口增大，吞吐量不断增加。当消息窗口为 80 K 时，基本到达网卡流量上限。

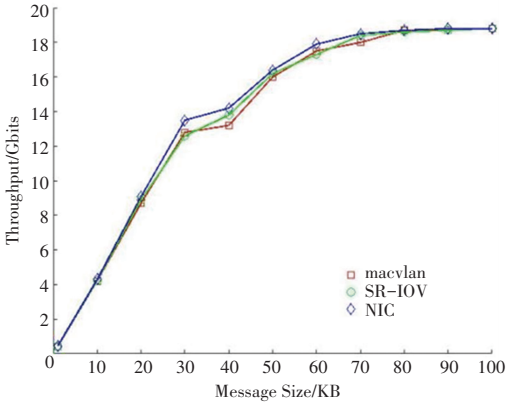

图 2　3 种网络方案 TCP 吞吐量实验结果

Fig. 2　Experimental Results of TCP Throughput of Three Network Schemes

3 种网络方案 UDP 网络时延实验结果如图 3 所示，SR-IOV 方案在不同的读写缓冲区大小的场景下与物理网卡和 macvlan 具有基本相同时延，并且随着读写缓冲区增加时延不断增加，当读写缓冲区大小为 32 K 时时延在 28 μs 左右。

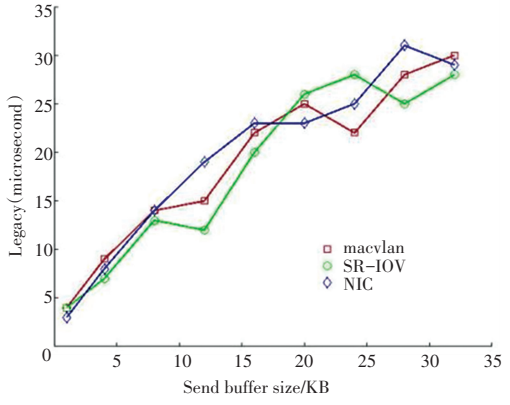

图 3　3 种网络方案 UDP 时延实验结果

Fig. 3　Experimental Results of UDP legacy of Three Network Schemes

3 种网络方案 UDP 网络丢包率实验结果如图 4 所示，SR-IOV 和 macvlan 两种方案具有基本相同的丢包率，都在 2‰左右，比物理网卡丢包率稍高。

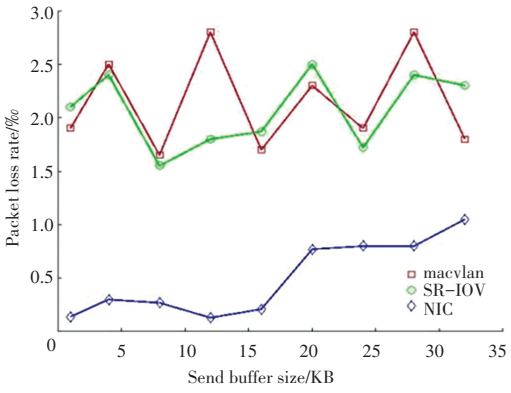

图 4　3 种网络方案 UDP 丢包率实验结果

Fig. 4　Experimental results of UDP packet loss rate of three network schemes

4　结束语

本文提出基于 SR-IOV 的 Docker 网络架构，能够解决容器网络面临无法通讯、无法隔离、性能差和管理复杂等难题，也能解决 Macvlan 方案中将网卡设置为混杂模式的问题，并且基于硬件虚拟化的 SR-IOV 技术也具备较好的稳定性，适合应用在大规模、对网络敏感的容器集群。当然 SR-IOV 技术也存在一定的不足，主要表现 SR-IOV 需要软硬件的支持，对于存量不支持 SR-IOV 的设备无法利用这项功能。另外，目前物理网卡能够虚拟出来 VF 数量有限，一般万兆网卡能够虚拟出 63 个 VF，但是在一些无状态应用场景下，单个服务器上部署大量容器，可能出现 VF 数量不足等情况。如何结合其它网络方案，比如 VXLAN（Virtual Extensible LAN）或者在服务器上配置更多支持 SR-IOV 网卡，满足单台服务器上部署大量容器场景，这是后续需要研究的课题。

参考文献

[1] 浙江大学 SEL 实验室. Docker 容器与容器云[M]. 人民邮电出版社，2015:2-7.
[2] (俄)Fabrizio Soppelsa(法布里齐奥·索贝尔). Swarm 容器编排与 Docker 原生集群[M]. 电子工业出版社，2017:3-10.
[3] 单朋荣，杨美红，赵志刚，等. 基于 Kubernetes 云平台的弹性伸缩方案设计与实现[J]. 计算机工程，2021，47(1)，312-320.
[4] 王亚玲，李春阳. 基于 Docker 的 PaaS 平台建设[J]. 计算机系统应用，2016，25(3)，72-77.
[5] 刘明达，马龙宇. 一种基于 SR-IOV 技术的虚拟环境安全隔离模型[J]. 信息网络安全，2016(9):84-89.
[6] 徐维波. 基于 SDN 和 Docker 容器的网络虚拟化研究[D]. 重庆：重庆大学，2017.

（下转第 185 页）