基于 CNI 的容器网络插件性能对比研究

李 雯[1] 戴琳琳[2] 李贝贝[2] 潘浪涛[2]
LI Wen DAI Linlin LI Beibei PAN Langtao

摘 要 作为现代软件研发和部署的一种方法，云原生架构旨在发挥云计算的技术优势，通过采用容器技术，提供良好的跨平台、服务弹性伸缩等能力，已成为众多企业数字化转型进程中的选择。面向容器的网络通信类型较为多样，容器为其提供了标准化的 CNI 接口，支持多种网络插件集成。通过设计并分类容器网络通信过程中的不同场景，开展针对 Flannel、Calico 和 Cilium 网络插件的网络耗时、时延、资源消耗率、吞吐量等多个指标的性能测试，朴素化表征并对比测试结果，为企业应用中网络插件的合理选型提供参考。

关键词 容器；CNI；Flannel；Calico；Cilium；容器网络插件

doi：10.3969/j.issn.1672-9528.2025.10.027

0 引言

云原生已发展为近年来计算机领域的热门话题之一，因其具有良好的弹性能力、服务自治故障自愈能力以及大规模可复制能力等特点，为不同产业释放了极大的红利，加速了异构资源标准化进程。通过数字基础设施升级解放了生产力，云原生进一步提升了业务应用的迭代速度，驱动云计算发展，赋能业务创新，推进企业数字化变革 [1]。其中，以 Kubernetes 为代表的容器编排引擎，已广泛应用于现代云原生应用的部署和管理 [2]，网络是其核心组件之一，为同一主机内和不同主机间的不同 Pod 提供相互连接，CNI（container network interface）为容器运行时之间的交互提供了统一的接口和规范。容器中的每个 Pod 均有唯一的 IP 地址用于内部通信，针对不同的联网需要，目前已发展出了多种 CNI 插件，这些插件执行 Kubernetes 集群中的 Pod 联网任务，负责维护和编排 Pod 网络。当容器集群频繁地创建或停止 Pod 时，CNI 插件将与容器运行时协同，并将容器网络命名空间与主机网络命名空间进行连接，不限于通过创建虚拟网卡、对新的 Pod 分配唯一的 IP 地址、应用网络策略、分发路由信息到集群的其他节点等一系列操作，完成整个容器集群的组网。

前期已经有大量学者对容器网络 CNI 的性能进行了研究。Park 等人 [3] 利用 OpenStack 云平台和 Kubernetes 容器管理环境，设计了基于 Kuryr[4] 和 Flannel[5] 的容器网络架构，主要研究了基于 Overylay 和 Kuryr 原生 VLAN 的数据吞吐量的测试；刘渊等人 [6] 提出了一种结合自适应 Overlay Network 与 Directrouting 的容器网络模式，引入 Canal 到 Flannel 架构中，测试容器网络的传输速率。Qi 等人 [7-8] 围绕功能、性能和扩展性开展研究，指出 Pod 数量的增加和 Http 工作负载的变化，CNI 插件的表现存在差异。Novianti 等人 [9] 通过设计不同的测试场景，对 Flannel、Calico[10]、Cilium[11]、Antrea、Kube-router 等 CNI 插件进行了基准测试，完成吞吐量、CPU 和内存资源消耗等性能指标的测试，得出 Kube-router 的吞吐量较高，其次是 Flannel；尚佳友等人 [12] 针对 Pod 网络深度定制，提出了新的 CNI 插件 N-Net，在吞吐量和网络延迟方面相比 Flannel 表现优异；肖文杨等人 [13] 构建了一个评价模型，通过评分表达网络 CNI 插件的功能和性能；为提升 Flannel 在虚拟可扩展局域网（virtual extensible lan, VxLan）模式下的性能损耗，严振星 [14] 整合了 Vx 模式和直接路由模式对 Flannel 进行优化，同时提升 Flannel 的高级网络策略管理和加密通信能力。

不同的容器网络 CNI 插件具有面向不同应用环境的网络特性。大部分的企业数字化进程中，一般更加聚焦业务改造和应用价值提升，兼顾考虑开源的协同治理和团队研发效率等因素，针对常见且基础的 CNI 插件，普遍局限于选型和使用，鲜有定制化研发新的网络 CNI 插件的情况。为加深对这些典型 CNI 插件的定性认识，本文基于较新版本的多种 CNI 插件，包括 Flannel、Calico 和 Cilium 等技术，针对多种应用场景下的时延、资源消耗、吞吐量等指标开展进一步的研究，力求朴素化表征不同 CNI 插件的整体性能，为企业容器应用

1. 中铁程科技有限责任公司 北京 100081
2. 中国铁道科学研究院集团有限公司电子计算技术研究所 北京 100081

[基金项目] 中国国家铁路集团有限公司科技研究开发计划系统性重大项目支持（P2024S002）

过程中的网络插件选型提供参考。

1 典型容器 CNI 网络插件简述

1.1 Flannel

Flannel 由 CoreOS 出品，主要通过部署于 Kubernetes 集群的每个节点的 flanneld 代理实现容器节点的网络配置，当容器需要分配 IP 地址时，Flannel 会从节点的子网（该子网通常是一个较小的 CIDR 块）中选择一个可用的私有 IP 地址，并将其分配给容器，每个容器均会被分配一个唯一的 IP 地址，以确保 Pod 网络的唯一性，同时通过设置虚拟网络、分配子网段、每个节点创建路由规则等，保障容器之间甚至跨主机的容器之间能够正常通信。每个 Flannel 集群都会维护一个可用的私有 Pod IP 地址池，通常采用 Etcd 或类似的分布式键值存储进行管理，每个 Flannel 节点将其网络配置信息写入 Etcd 中，并从 Etcd 中获取其他节点的信息。当 Flannel 初始化或运行时，通过 API Server 间接地与 Etcd 进行通信，获取网络配置信息。Flannel 也通常与 Kubelet 和 Kube Proxy 等 Kubernetes 组件集成，除实现 Pod 之间的网络通信外也用于服务发现。

Flannel 的 Overlay 网络（又称为覆盖网络）的实现主要包括 UDP、VxLan 和 Host-Gw 三种模式，其总结对比如表 1 所示。

表 1 Flannel 的 3 种 Overlay 网络模式对比

模式	构建方式	优点	缺点	适用场景
UDP	用户态封装 TUN/TAP 设备	兼容性强，配置简单	性能差，可靠性低，安全性弱	小规模集群、测试环境
VxLan	内核态 VxLan 封装 VTEP 隧道	性能较好，支持跨子网，安全性较高	配置稍复杂，有一定封装开销	中大规模集群、跨子网环境
Host-Gw	节点路由表直接转发	性能最佳，配置简单	要求同一二层网络，不支持跨子网	高性能需求、同一二层网络环境

表 1 中 UDP 网络模式是一种较早的实现方式；VxLan 模式因操作系统内核原生支持且性能较强，一般应用较广；Host-Gw 模式直接通过节点的路由表进行通信，性能最高，但要求各节点必须在同一网段。

1.2 Calico

Calico 是由 Tigera 公司开发的开源网络和网络安全解决方案，专为容器、虚拟机和基于主机的本地工作负载设计。Calico 以其高性能、灵活性和强大的网络策略功能而闻名，广泛应用于 Kubernetes 集群中。其核心组件包括 Felix、BIRD、Confd、Etcd（或 Kubernetes API）和 Calico CNI 插件。Felix 是 Calico 的守护进程，运行在每个节点上，负责配置路由、ACLs（访问控制列表）和其他网络相关的设置；BIRD 是一个轻量级的动态路由守护进程，用于在节点之间分发路由信息；Condfd 检测 Calico 的数据变化以便及时修改 BGP 配置，如 AS 编号、日志级别等。Etcd 或 Kubernetes API 用于存储 Calico 的配置数据和状态信息；Calico CNI 插件则负责与 Kubernetes 集成，管理 Pod 的网络配置。

Calico 的网络模型基于纯三层网络设计，不使用 Overlay 网络（如 VxLan 或 IPIP），而是通过 BGP（边界网关协议）在节点之间直接交换路由信息，使得 Calico 具有极高的性能和低延迟，特别适合大规模和高性能要求的集群环境。

Calico 支持基于标签的选择器、端口和协议的网络策略，并且可以与 Kubernetes 的网络策略 API 无缝集成，突显了其强大的网络策略功能，用户可以定义精细的访问控制规则，实现 Pod 之间的微隔离。Calico 还兼具了高级的安全功能，如网络加密、威胁检测和响应等。

Calico 的配置和管理通常通过命令行工具 calicoctl 进行，用户可以通过该工具创建、更新和删除网络策略、查看集群状态和诊断网络问题，支持丰富的监控和日志组件集成，帮助用户实时了解网络状态和排查问题。

1.3 Cilium

Cilium 是一款专为 Kubernetes 设计的高性能网络插件，专注于安全性和网络策略实施。与传统的 CNI（container network interface）插件不同，Cilium 基于 eBPF（extended berkeley packet filter）技术，能够在 Linux 内核层面实现网络策略的执行、负载均衡、网络流量监控等功能，同时避免了传统网络插件在用户态和内核态之间频繁切换的性能开销。

Cilium 的核心功能包括网络连接、网络安全和可观测性。在网络连接方面，Cilium 通过 eBPF 技术实现了高效的容器间通信，支持多种网络模式，包括 Overlay 网络（如 VxLan）和直接路由模式（如 Direct Routing）。与 Flannel 类似，Cilium 也能够为每个 Pod 分配唯一的 IP 地址，并确保跨主机容器之间的通信。不同的是，Cilium 通过 eBPF 程序直接在内核中处理网络数据包，从而大幅提升了网络性能。

在网络安全方面，Cilium 提供了强大的网络策略功能，支持基于身份（Identity-Based）的安全策略。Cilium 能够为每个 Kubernetes Pod 或 Service 分配一个唯一的安全身份，并基于这些身份实施精细化的网络访问控制。与传统的基于 IP 地址的防火墙规则不同，Cilium 的安全策略更加灵活且易于管理，能够有效防止未经授权的容器间通信。

在可观测性方面，Cilium 通过 eBPF 技术实现了对网络流量的深度监控和分析，能够实时捕获容器之间的网络流量，

并提供详细的流量统计信息，包括源 IP、目标 IP、端口、协议等，易与 Prometheus、Grafana 等监控工具集成，帮助用户更好地理解和优化集群的网络性能。

Cilium 的架构设计使其能够与 Kubernetes 的其他组件无缝集成。例如，Cilium 与 Kube-Proxy 集成，能够替代传统的 iptables 或 IPVS，实现 Kubernetes Service 的负载均衡功能；通过 eBPF 技术，Cilium 能够在内核中直接处理 Service 的流量转发，大幅提升负载均衡的性能和可扩展性。

Cilium 的网络模式主要包括 VxLan 和 Direct Routing。VxLan 模式适用于跨子网的容器通信，通过封装数据包实现 Overlay 网络；而 Direct Routing 模式则适用于同一子网内的容器通信，直接通过节点的路由表进行数据包转发，性能更高。与 Flannel 的 Host-GW 模式类似，Cilium 的 Direct Routing 模式也要求节点在同一子网内。

1.4 技术特性对比

Flannel、Calico 和 Cilium 均是 Kubernetes 中常用的用于管理容器网络的插件，但在设计目标、功能实现和性能上存在差异，基于技术本身的主要相关特性对比如表 2 所示。

表 2 不同 CNI 网络插件的特性对比

特性	Flannel	Calico	Cilium
网络模型	Overlay（VxLan/UDP）	基于路由（BGP）	基于 eBPF
性能	较低（隧道技术增加开销）	高（支持原生路由）	极高（内核级加速，直接流通）
网络策略	不支持	L3/L4 层策略	L3/L4/L7 层策略
安全性	较弱	较强	极强（身份感知策略）
可观测性	无	基本	强大（基于 eBPF）
适用场景	小型集群，简单场景	中大型集群，高性能场景	高性能、高安全、复杂场景
部署复杂度	简单	中等	较高

2 CNI 网络插件的性能对比试验

2.1 试验设计

生产环境中针对 Kubernetes 集群 Pod 资源的访问，一般由抽象的 Service 资源对象进行定义和纳管。针对 Service 的访问，其发起方一般来源于 Node 节点、集群内 Pod 或集群外应用等，不同访问方式的网络性能指标均存在差异。本文分类设计试验并开展相关研究，通过不同的工具进行数据统计与图形展示，定量观测 Flannel、Calico 和 Cilium 等不同网络 CNI 插件的性能对比结果。

本文将以 Cluster IP 方式分别通过 Node 节点和 Pod 对象发起对 Kubernetes 集群 Service 的访问，基于 Traefik Ingress 方式由外部访问 Kubernetes 集群的 Service 资源，包括跨 Node 节点的网络访问，研究 CNI 网络插件的网络延迟、资源利用率和吞吐量等网络性能指标。主要通过 Shell 脚本统计时延和资源利用率等数据，采用网络性能测试工具 iPerf 测试吞吐量等网络性能，最后通过 Python 的 Matplotlib 库将试验数据进行可视化展示。

2.2 试验环境

为避免试验过程中潜在的带宽饱和对生产业务的影响，本文测试基于测试环境进行。通过虚拟机搭载 Redhat Linux 8.5 操作系统构建基础的 Kubernetes 集群环境，单个 master 节点和 node 节点部署于 2 台宿主机上，每个宿主机配置 2 个 CPU 核心和 4 GB 内存。在此基础上，试验中用到的相关组件、部署方式及相关版本如表 3 所示。

表 3 容器 CNI 网络插件测试关键组件描述

序号	组件名称	功能	版本	部署方式
1	Kubernetes	编排	1.30.2	Kubeadm 部署
2	Containerd	容器引擎	1.7.18	二进制方式部署
3	Etcd	分布式对象存储	3.5.9	二进制方式部署
4	Flannel	容器 CNI 网络插件	0.25.6	Yaml 部署
5	Calico	容器 CNI 网络插件	3.25.2	Yaml 部署
6	Cilium	容器 CNI 网络插件	1.16.5	Helm 部署

2.3 对比试验

在开展试验前，本文完成基础的 Kubernetes 分布式集群搭建，封装了无状态的 Java Web 接口应用程序作为服务提供者，通过资源控制 Deployment 将其稳定运行于 Pod 中，并由 Service 提供统一的访问入口。宿主机和容器内部均支持 curl 和 iPerf 工具。

（1）场景一：由节点发起服务请求，以 Cluster IP 内部方式访问 Service。

由 Kubernetes 集群的 master 节点宿主机发起请求，由 curl 命令发起访问服务 Service 的内部地址，开展针对 Flannel、Calico、Cilium 等指标的 10 组测试，结果如图 1 所示，其中实线为对应的采样连接耗时、网络时延及总耗时，虚线为相应的平均耗时。Flannel 的平均连接耗时、网络延迟及总耗时（平均 3.032 ms）最长，Cilium 次之且相对耗时较短，Calico 耗时及网络延迟最小。

（2）场景二：由 Pod 发起服务请求，以 Cluster IP 内部方式访问 Service。

由 Kubernetes 集群的 Pod 内部直接发起服务请求，通过容器内部的 curl 命令发起访问服务 Service 的内部地址，开

展针对 Flannel、Calico、Cilium 等指标的 10 组测试，结果如图 2 所示，其中实线为对应的采样连接耗时、网络时延及总耗时，虚线为相应的平均耗时。当通过 Pod 由内而外向 Service 发起请求时，与由 Node 发起服务请求的各 CNI 网络插件的耗时较为相似且接近，Calico 依然保持最优。

（3）场景三：通过 Traefik Ingress 方式由外部向内访问 Service。

为避免网络传输的潜在影响，本试验场景依然由 Kubernetes 集群的 master 节点宿主机发起请求，但此时需要将集群 Service 的访问方式由 Cluster IP 修改为 NodePort 方式，通过 curl 命令发起访问服务 Service 的外部地址，开展针对 Flannel、Calico、Cilium 等指标的 10 组测试，结果如图 3 所示，其中实线为对应的采样连接耗时、网络时延及总耗时，虚线为相应的平均耗时。连接耗时方面 Flannel 依然耗时相对较长，Cilium 次之；整体耗时方面，Cilium 与 Flannel 较为接近，Calico 以领先大约 0.5 ms 保持最优。在这种试验场景下，所有网络 CNI 插件的整体耗时与场景一的内部访问方式较为接近。

（4）场景四：多种场景的网络组件性能对比。

在接下来的试验中，将融合不同测试场景下 CNI 网络插件的吞吐量和资源占用等指标情况。试验共进行 5 轮，每轮试验中针对不同的 CNI 网络插件分别发送 16 GB 的基准测试数据，不限制各网络插件的传输速率，以获取其最佳的性能表现，同时观测在试验期间的 Pod 的平均 CPU 利用、内存占用等资源消耗情况。分别面向跨节点的 Node 与 Node 之间、跨节点的 Pod 与 Node 之间，以及跨节点的 Pod 与 Pod 之间等不同场景下的 Flannel、Calico 和 Cilium 的网络插件性能，试验结果分别如图 4~6 所示，其中实线表征每轮的均值，虚线代表 5 轮的均值。

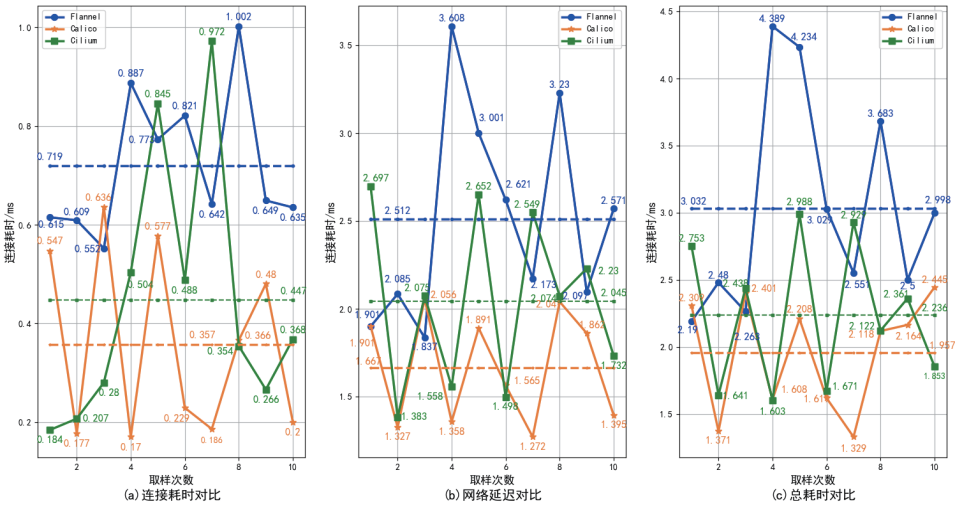

图 1 Node 节点发起访问 Service 资源的 CNI 插件耗时对比

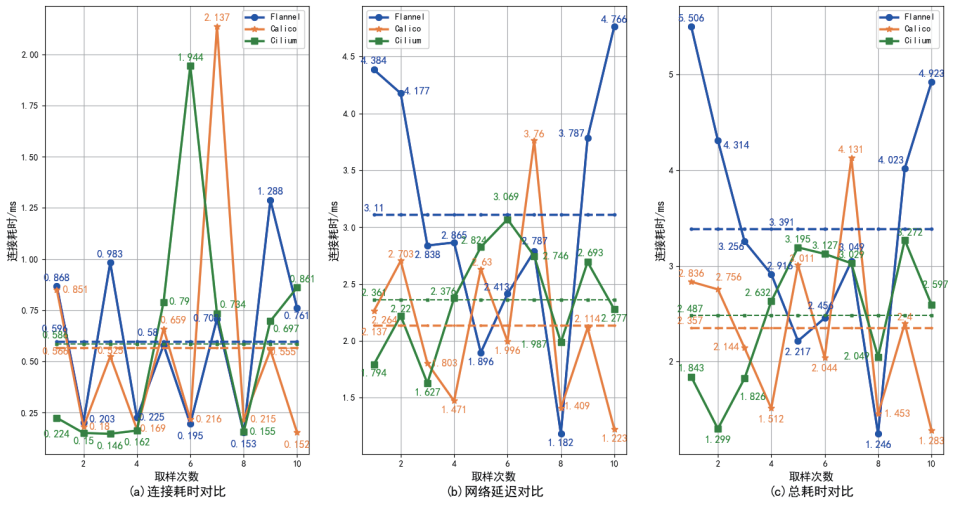

图 2 通过 Pod 发起访问 Service 资源的 CNI 插件耗时对比

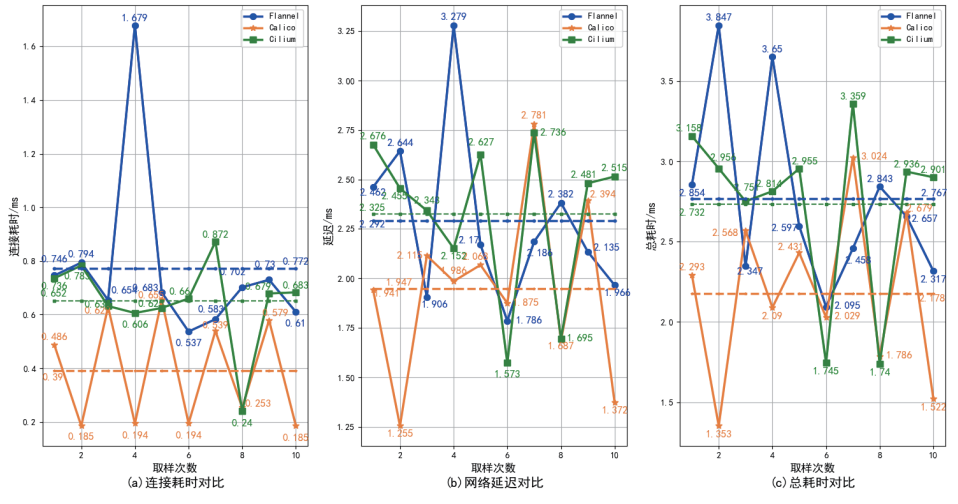

图 3 通过 Traefik Ingress 方式由外部发起访问集群 Service 的 CNI 插件耗时对比

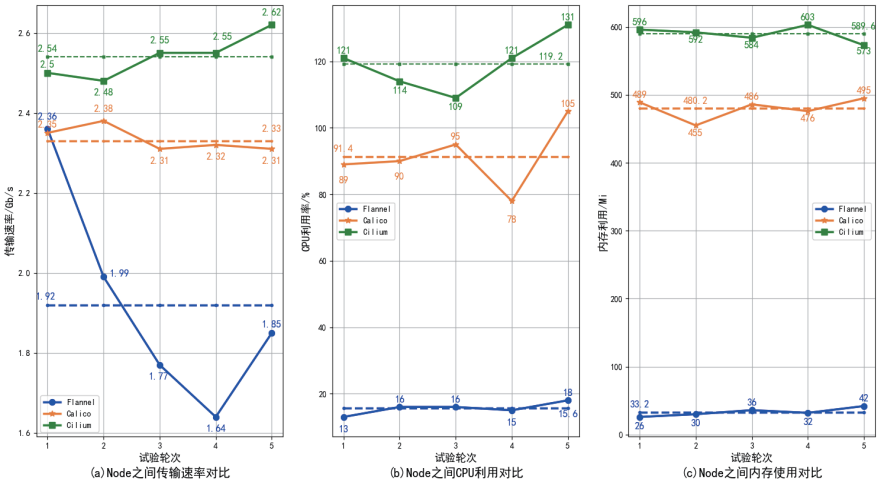

图 4 跨节点的 Node 之间 CNI 插件性能对比

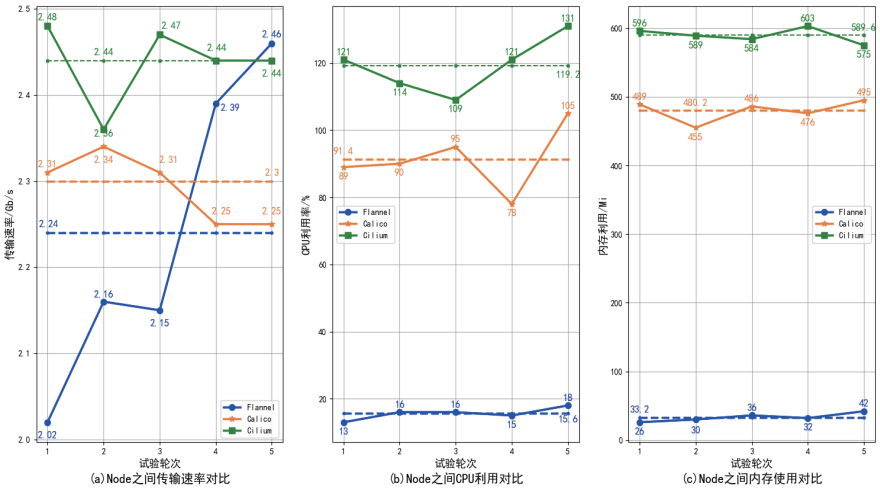

图 5 跨节点的 Pod 与 Node 之间的 CNI 插件性能对比

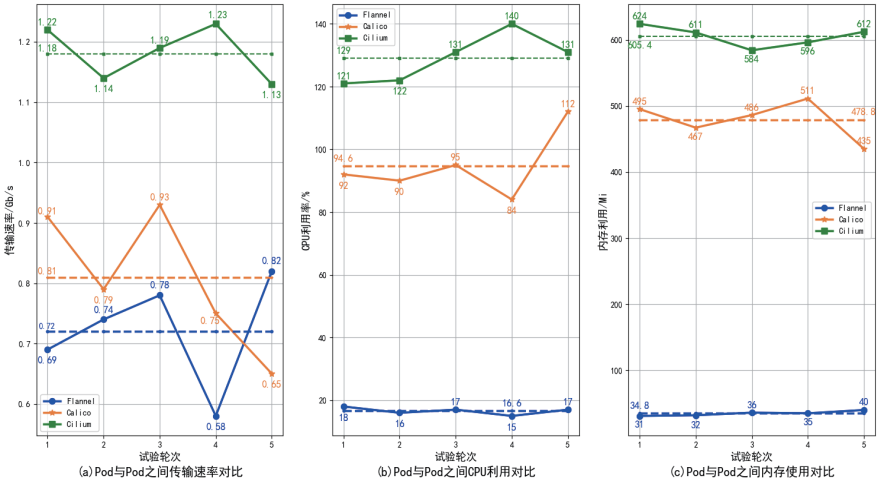

图 6 跨节点的 Pod 之间的 CNI 插件耗时对比

针对网络组件设计的多个应用场景的性能试验，5 轮测试的结果中，Cilium 的吞吐量最高，同时其 CPU 利用和内存使用也最高，Calico 次之，相比之下 Flannel 的网络吞吐量最小，资源利用也最低。

2.4 试验总结

针对不同的网络应用场景，Flannel、Calico 和 Cilium 在连接耗时、网络延迟及总耗时等指标方面相差不大，平均连接耗时均在 1 ms 以内，网络时延和总耗时方面，Flannel 表现弱于 Calico 和 Cilium。资源利用率方面，与 CNI 插件的网络吞吐量正相关，且相同数据发送量的情况下，Cilium 的吞吐量最高，Calico 次之，Flannel 性能相比最差。

3 结语

本文针对容器网络插件的所有应用场景进行分类，面向常用的 Flannel、Calico 和 Cilium 等不同容器 CNI 网络插件，设计并完成连接耗时、网络延迟、总耗时、吞吐量以及 CPU/ 内存资源使用率等多个性能指标试验，为企业容器网络插件的选型提供依据。随着企业业务量的增加，一般磁盘、内存、网络带宽等基础设施不再是企业应用中的主要考虑因素，网络插件的吞吐量成为大型企业的首要考虑因素，试验中也验证了在耗时、网络延迟等方面相差不足 1 ms 时，Cilium 成为高性能微服务架构的首选。本文并未涉及网络插件自身的安全性和可观测性等方面，是企业后续应用中需要持续关注的方向。

参考文献：

[1] 云原生产业联盟 . 云原生发展白皮书（2020 年）[R]. 北京：中国信息通信研究院 , 2020.

[2] 朱亚州，杜平川，柴志雷 . 基于 Kubernetes 的异构任务调度方法 [J/OL]. 计算机工程 ,1-10[2025-06-26].https://doi.org/10.19678/j.issn.1000-3428.0069437.

[3]PARK Y, YANG H, KIM Y. Performance analysis of CNI (Container Networking Interface) based container network[C]// 2018 International Conference on Information and Communication Technology Convergence (ICTC).Piscataway: IEEE, 2018: 248-250.

[4] Kuryr[EB/OL].[2025-06-15].https://github.com/openstack/kuryr.

[5]Flannel[EB/OL].[2025-06-19].https://github.com/flannel-io/flannel.

[6] 刘渊 , 乔巍 . 云环境下基于 Kubernetes 集群系统的容器网络研究与优化 [J]. 信息网络安全 ,2020,20(3):36-44.

[7]QI S X, KULKARNI S G, RAMAKRISHNAN K K. Assessing container network interface plugins: functionality, performance, and scalability[J]. IEEE transactions on network and service management, 2020, 18(1): 656-671.

[8] QI S X, KULKARNI S G, RAMAKRISHNAN K K. Understanding container network interface plugins: design considerations and performance[C/OL]//2020 IEEE International Symposium on Local and Metropolitan Area Networks, LANMAN.Piscataway:IEEE,2020[2025-02-12]. https://ieeexplore.ieee.org/document/9153266.DOI: 10.1109/LANMAN49260.2020.9153266.

[9]NOVIANTI S, BASUKI A. The performance analysis of container networking interface plugins in kubernetes[C]//Proceedings of the 6th International Conference on Sustainable Information Engineering and Technology.NewYork: ACM, 2021: 231-234.

[10] Calico[EB/OL].[2025-04-25]. https://github.com/projectcalico/calico.

[11]Cilium[EB/OL].[2025-06-09]. https://cilium.io/.

[12] 尚佳友 , 王晓锋 , 刘渊 . 面向 Kubernetes 的高性能网络方案研究 [J]. 现代计算机 , 2021 (21): 15-21.

[13] 肖文扬 , 刘伟鸿 , 朱宗卫 . 容器网络插件的测试与量化分析 [J]. 计算机应用与软件 , 2024, 41(8): 140-148.

[14] 严振星 . 云环境下的容器网络性能优化与实现 [D]. 成都 : 电子科技大学 , 2024.

【作者简介】

李雯（1987—），男，湖南邵阳人，硕士研究生，副研究员，研究方向：交通信息工程及控制。

戴琳琳（1983—），女，福建莆田人，硕士研究生，研究员，研究方向：铁路客运信息化应用。

李贝贝（1987—），男，安徽阜阳人，博士研究生，副研究员，研究方向：容器云原生、人工智能等。

潘浪涛（1985—），男，四川达州人，硕士研究生，副研究员，研究方向：铁路计算机应用技术等。

（收稿日期：2025-04-30 修回日期：2025-10-14）

（上接第 114 页）

由图 3 可以看出，与其他 3 种方法相比，本文所提方法的单管开路故障诊断误报率最低。这主要得益于本文方法综合考虑了电压、电流和温度等多种信号，通过多特征融合和模式识别，实现了故障器件的精准定位与诊断。

3 结语

本文针对电力电子变换器单管开路故障诊断，创新提出基于多传感器融合的方法。搭建多物理场耦合监测网络，实时捕获电压、电流、温度等故障敏感信号，采用滑动平均、低通滤波等技术对单传感器信号精准状态估计，运用关联分析等融合规则挖掘信号内在联系。在此基础上，量化电压突变幅度、电流对称性指标等关键特征参数，精准提取故障特征。构建故障特征向量和决策表，利用特征相似度匹配定位故障器件。实验验证表明，在负载突变工况下，该方法误报率大幅降低，有效提升了故障诊断可靠性。

参考文献：

[1] 荣飞 , 孙克强 . 基于最近电平逼近调制的 IGBT 开路故障诊断方法 [J]. 湖南电力 ,2022,42(4):7-12.

[2] 张磊 , 余茂全 , 夏远洋 . 基于 FastICA-LDA 的光伏并网逆变器故障诊断 [J]. 新余学院学报 ,2024,29(5):40-48.

[3] 宋保业 , 鲁朋 , 许琳 . 基于深度自编码器网络的逆变器开关管开路故障诊断 [J]. 山东科技大学学报 (自然科学版), 2023, 42(6):117-128.

[4] 张国宝 , 王朝廷 , 黄伟民 , 等 . 基于多源传感器数据融合的断路器故障诊断方法 [J]. 高电压技术 ,2025,51(2):660-668.

[5] 刘洋 , 赵鲁 , 马呈瑶 , 等 . 双向 CLLC 谐振变换器开关管开路故障特性分析 [J]. 太阳能学报 ,2025,46(1):531-539.

[6] 申永鹏 , 马梓洋 , 金楠 , 等 . 三相电压源型逆变器单传感器相电流重构故障诊断策略 [J]. 电机与控制学报 , 2024, 28(10): 135-146.

[7] 罗屿 , 陈春阳 , 石英春 , 等 . 基于电流值分析的三相逆变器开关管开路故障诊断方法 [J]. 铁道科学与工程学报 , 2024, 21(10): 4370-4382.

【作者简介】

黄影（1998—），女，重庆人，硕士，助教，研究方向：电力电子技术。

王振水（1991—），男，河南邓州人，硕士，助教，研究方向：电力电子技术、电机及驱动技术。

（收稿日期：2025-04-16 修回日期：2025-10-11）