面向边缘计算的轻量级容器化网络功能部署架构

王善程[1] 陈玉林[2]
WANG Shancheng CHEN Yulin

摘 要 为满足边缘计算场景中对网络功能高效、灵活部署的需求，文章提出了一种面向边缘计算的轻量级容器化网络功能部署架构，以容器技术为基础，结合网络功能虚拟化理念，构建适配边缘节点资源特性的部署模型，在设计过程中选取典型容器平台进行性能对比，并构建测试环境，对架构的资源占用率、启动时延与网络吞吐能力进行验证。实验结果表明：该架构能明显降低资源开销，提高网络功能的启动效率，对推动边缘计算环境下网络功能的轻量化演进具有一定的现实意义与应用价值。

关键词 边缘计算；容器化；网络功能虚拟化；轻量部署；系统架构

doi：10.3969/j.issn.1672-9528.2026.02.021

0 引言

边缘计算（Edge Computing）指在靠近物或数据源头的网络边缘侧，融合网络、计算、存储、应用核心能力的开放平台，就近提供边缘智能服务，以满足行业数字化在敏捷连接、实时业务、数据优化、应用智能、安全与隐私保护等方面的关键需求，边缘计算是一种分布式计算框架，使企业应用程序更接近数据源，如物联网设备或本地边缘服务器，这种接近数据源的方式可以带来强大的业务优势，包括更快的洞察力、更快的响应时间和更好的带宽可用性。边缘计算作为推动新一代网络基础设施发展的关键技术，能有效缓解云端压力，提升数据处理效率，在复杂多变的网络环境中，传统网络功能部署方式存在资源占用高、灵活性差等问题，难以满足边缘节点对轻量化、快速响应的实际需求。容器化技术因其启动速度快、资源开销小、部署灵活等特点，逐渐成为网络功能虚拟化的新趋势。针对边缘计算场景下的实际挑战，构建一种基于容器的轻量级网络功能部署架构，探索适用于资源受限环境的高效方案，对于提升边缘计算系统的整体性能和应用扩展能力具有重要研究价值。

1 轻量化容器技术

1.1 容器技术演进

容器技术作为操作系统级虚拟化的一种实现方式，其演进过程体现了对计算资源利用率与服务部署效率不断优化的技术趋势，早期的虚拟化依赖于传统虚拟机架构，采用完整操作系统镜像实现隔离，尽管具备较强的安全性与兼容性但资源消耗大、启动时间长等问题逐渐显现，难以适应弹性计算和微服务架构的快速发展需求，容器技术在 Linux 内核支持下逐步成熟采用进程级隔离机制，借助命名空间与控制组实现资源控制与环境隔离，具备轻量、灵活的优势。容器镜像构建基于分层文件系统，让部署版本可快速切换且具备良好的可移植性。近年来，随着云原生理念的兴起容器生态逐步完善，涵盖编排、网络、安全、监控等多个维度，推动了容器技术在边缘、云计算、物联网等多个领域的广泛应用。

1.2 主流容器平台对比

当前主流的容器平台中 Docker 作为最早普及的容器引擎之一，具备丰富的社区资源和工具链支持，其镜像管理与容器生命周期控制高度集成，但在资源消耗和守护进程设计上存在一定冗余，不利于极端资源受限场景，containerd 作为 Docker 的核心组件之一，在功能划分上更为精简专注于镜像管理与运行时管理，符合现代微服务架构对可插拔与分层组件的需求，适用于系统底层集成与边缘侧的定制化部署，Podman 在无守护进程的运行模式下提供与 Docker 兼容的命令接口，具备更高的安全性和系统控制权限便于与系统服务进行集成，适用于边缘节点中对安全隔离要求较高的场景。不同容器平台在镜像拉取效率、容器启动时延、资源占用比例以及日志管理方式等方面表现各异，需根据实际部署环境进行权衡与选择。边缘计算场景下对于容器平台的选型要考虑功能完整性，还需充分评估平台在网络环境不稳定、计算资源有限、启动响应时效等方面的综合适应能力。

1.3 边缘场景的适配性分析

容器化方案在边缘计算环境中的适配性成为评估部署架构可行性的核心要素，轻量级容器平台因其低资源消耗特性

1. 武警河南总队信阳支队 河南信阳 464000
2. 信阳涉外职业技术学院 河南信阳 465550

能有效降低节点负载，提高系统整体可用性与稳定性，在镜像体积控制方面使用精简化操作系统或构建最小功能镜像可以减少启动延迟与带宽消耗，有助于应对网络波动与远程分发难题，调度策略方面容器编排工具如 K3s、KubeEdge 针对边缘场景进行了适配优化，支持异步通信、边缘离线处理与节点自治运维，提升容器管理的灵活性与鲁棒性。对于边缘节点所需的快速响应能力，容器具备秒级甚至毫秒级的启动能力，相较传统虚拟机具备明显优势，可用于应对突发任务与服务弹性扩展需求 [1]。系统安全方面容器平台根据运行时隔离、权限最小化与安全策略控制减少潜在攻击面，增强了系统对外部风险的防御能力。

2 网络功能部署架构

2.1 NFV 架构简述

NFV 根据将物理网络设备中的专用功能抽象为软件模块，在标准计算平台上运行，实现网络服务从硬件依赖向灵活部署的转变，其核心架构由三部分组成，分别是虚拟网络功能（VNF）、NFV 基础设施（NFVI）与 NFV 管理与编排系统（MANO），VNF 承担实际网络处理任务，如防火墙、负载均衡、NAT 等功能的实现，具备可插拔、可移植的特性。NFVI 提供计算、存储和网络资源的支持，抽象了底层物理硬件支持在多个异构平台上统一部署虚拟化资源。MANO 作为协调和管理 VNF 生命周期的核心组件，负责资源调度、服务编排、状态监控以及故障恢复策略的实施 [2]。传统 NFV 主要基于虚拟机进行功能部署，存在资源利用效率不高、启动延迟较长、弹性伸缩不足等问题。在边缘计算环境下，上述问题更为突出，制约了网络功能虚拟化的普及与高效运行。容器技术的引入为 NFV 架构带来了新的活力，能缓解传统 NFV 在边缘部署中面临的性能瓶颈和资源开销限制，让网络功能具备更高的可部署性与动态响应能力，如图 1 所示。

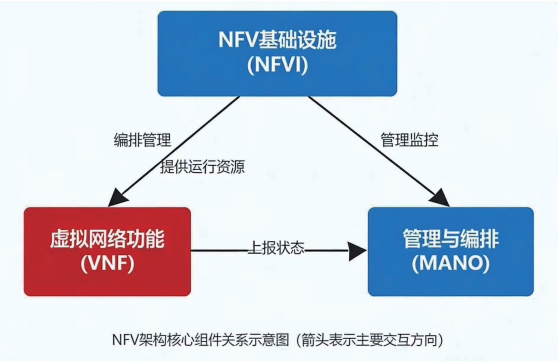

图 1 NFV 架构与功能模块关系示意图

2.2 容器化 NF 部署方案

容器化网络功能（CNF）构建基于统一镜像规范，采用构建 - 部署 - 运行的流水线方式，在持续集成与持续交付流程中实现服务的自动发布与迭代，部署方案通常集成 Kubernetes 等编排系统，以实现资源动态分配和状态监测与故障恢复，与虚拟机部署方式相比，容器部署在资源隔离方式上更加轻便，启动与销毁过程不涉及完整操作系统加载提升启动效率与部署密度。网络功能在容器内运行涉及网络命名空间与虚拟网络接口的配置管理，适配方式需综合考虑服务链编排、负载分发与数据转发效率，采用 SR-IOV、DPDK 或 eBPF 等高性能技术以保障转发性能与低时延特性 [3]。容器编排系统可借助服务网格构建跨容器通信机制，在不改动业务逻辑的前提下实现链路加密、负载均衡与可观测性提升。资源调度方面容器平台需配合边缘侧资源监测机制，基于节点负载预测与服务需求波动进行动态迁移与部署优化。为量化容器化部署中资源调度效率与容器性能损耗可构建性能模型，用公式表示为：

$$\eta = \frac{\sum_{i=1}^{n}\left(w_i \cdot \frac{C_i^{\text{used}}}{C_i^{\text{total}}}\right)}{\sum_{i=1}^{n} w_i} \tag{1}$$

式中：η 表示整体资源利用率；C_i^{used} 和 C_i^{total} 分别表示第 i 个资源类型的使用量与总量；w_i 表示该资源的权重因子，反映其对系统性能的影响程度，该模型可用于评估不同容器调度策略在边缘节点上的资源分配效果与系统负载平衡性。

2.3 边缘侧部署策略

边缘计算环境中部署策略要兼顾服务质量与性能优化，还需适配设备异构性与部署异步性，多节点异步部署场景中网络功能需具备分布式启动与局部容错能力，调度系统应基于网络拓扑结构与地理分布特征进行服务映射与节点绑定，形成区域自治与中心协同的多层部署格局，系统需引入边缘感知机制，根据流量行为预测与服务热度检测调整部署优先级与副本数量，以实现资源与需求之间的动态匹配。服务部署应基于负载动态迁移策略与微服务粒度控制，在资源紧张情况下可采用分阶段拉起或功能裁剪方式提高节点承载能力。高可用性保障方面，可构建冗余部署模型与动态恢复机制，结合状态同步方案与副本一致性协议，在节点故障时实现服务的快速接管与最小服务中断 [4]。针对不同边缘场景的异构硬件基础和部署成本控制要求，调度系统应具备资源拓扑感知与多维度策略适应能力，边缘节点间的数据流路径优化问题可表示为一类资源约束型最短路径问题，其目标是在延迟与资源成本之间取得最优平衡，路径选择需结合网络负载状态与带宽变化趋势动态调整。路径最优化函数为：

$$\min_{P\in\mathcal{P}}\left(\sum_{(i,j)\in P}\alpha_{ij}\cdot d_{ij}+\beta_{ij}\cdot c_{ij}\right) \tag{2}$$

式中：P 为所有可能路径集合；d_{ij} 为边 (i, j) 的传输延迟；c_{ij} 为资源消耗成本；α_{ij} 和 β_{ij} 分别为延迟权重与资源成本权重系数，副本冗余度最优化模型用公式表示为：

$$R^* = \arg\max_{R\in\mathbb{N}} \left(\frac{A(R) - L(R)}{C(R)} \right) \tag{3}$$

式中：R 表示部署副本数量；$A(R)$ 表示系统可用性函数；$L(R)$ 表示因冗余增加引发的资源负载函数；$C(R)$ 表示与副本部署相关的资源成本函数，最大化目标是提高系统整体效益与冗余利用效率。

3 应用验证与分析

3.1 实验环境构建

实验设计围绕边缘计算环境下容器化网络功能部署的可行性与性能表现进行展开，主要验证轻量级容器技术在边缘节点中运行网络功能时的启动延迟、资源占用、网络吞吐与稳定性表现，实验目标为对比不同容器平台在网络功能部署过程中的关键性能指标，并分析其对边缘计算系统整体运行效率的影响，实验对象包括 3 种主流容器平台：Docker、containerd 与 Podman，以及两个典型网络功能服务：轻量级防火墙模块与负载均衡模块，均为开源项目构建，具备较高移植性与部署可控性。实验硬件平台为 ARM 架构与 x86 架构混合边缘节点，其中 ARM 节点使用树莓派 4B，内存为 4 GB，四核处理器主频为 1.5 GHz，x86 节点使用 Intel NUC 平台，配备 8 GB 内存与四核 i5 处理器，主频 2.4 GHz。网络连接采用本地交换机构建内网，测试工具选用 iperf3 进行吞吐量测试，top 与 pidstat 用于资源监控，wrk 用于模拟 HTTP 请求压力，实验采用脚本自动部署服务与采集性能数据，每项测试进行五轮，取平均值处理 [5]。部署采用 K3s 轻量级 Kubernetes 集群，配置每个节点单独调度一个容器网络功能模块，以便独立监控性能数据，避免资源争用干扰实验结果。

3.2 性能测试结果

性能测试聚焦网络功能模块的启动时间、服务延迟、并发处理能力与数据吞吐量，实验过程中，每种容器平台在不同架构节点上部署相同配置的网络功能模块，并记录从镜像拉取到服务端口就绪的时间，测量在高并发请求条件下系统响应能力与丢包情况。数据记录如表 1 所示。由表 1 可知，containerd 与 Podman 在启动时间上显著优于 Docker，其中 Podman 的启动时间为 790 ms，containerd 为 870 ms，而 Docker 为 1 580 ms，差距接近一倍。服务延迟方面，containerd 控制在 35.2 ms，Podman 略高为 38.9 ms，Docker 达到 41.7 ms，说明轻量级容器平台能有效减少处理延时。在并发能力方面，containerd 支持 1 120 个同时连接而无明显性能衰减，Podman 保持在 1 060，Docker 受限于资源调度机制，最大并发连接仅为 940。网络吞吐量测试中，containerd 达到 728 Mbit/s ，Podman 为 702 Mbit/s ，Docker 表现最差，仅为 613 Mbit/s 。

表 1 不同容器平台网络功能性能对比

容器平台	CPU 使用率 /%	内存占用 /MB	磁盘读写速率 /(kB·s^{-1})
Docker	38.4	152	783
containerd	27.1	126	592
Podman	25.6	118	548

3.3 资源开销评估

为评估容器平台在边缘节点上部署网络功能服务时的资源消耗情况，本部分测试容器在运行状态下的 CPU 使用率、内存占用与文件系统 I/O 活动强度，测试采用 pidstat 采集容器进程的平均 CPU 使用率，top 采集运行期间内存占用峰值，iostat 测量磁盘读写速率。各项数据统计如表 2 所示。

表 2 容器平台资源占用情况对比

容器平台	启动时间 /ms	平均服务延迟 /ms	最大并发连接数	网络吞吐量 /(Mbit·s^{-1})
Docker	1 580	41.7	940	613
containerd	870	35.2	1 120	728
Podman	790	38.9	1 060	702

由表 2 可知，Podman 在各项资源开销指标中表现最优，CPU 占用为 25.6%，内存峰值为 118 MB，磁盘读写速率为 548 kB/s，均为最低值，containerd 紧随其后，CPU 使用率 27.1%，内存占用 126 MB，磁盘 I/O 为 592 kB/s。Docker 在资源开销方面最为明显，CPU 使用率达到 38.4%，内存占用 152 MB，磁盘读写速率高达 783 kB/s。结合性能测试结果可以推断，Docker 在边缘场景中运行代价较高，其架构设计中对守护进程与容器引擎的紧耦合造成了额外资源消耗。Podman 采用无守护模式，容器进程直接由系统服务托管，运行开销更低适用于嵌入式设备与低功耗节点。

4 结论

实验结果表明，容器平台具备良好的启动效率、资源利用率与运行稳定性，能有效支撑网络功能在边缘环境中的快速部署与灵活扩展，研究结果为边缘计算系统的网络功能虚拟化提供了可行路径，具有一定的工程应用价值与研究拓展空间。

基于 YOLOv8–MT 与 RGB–D 相机的番茄检测方法

江 健[1]
JIANG Jian

摘 要 针对番茄采摘机器人视觉系统在作业中面临的果实遮挡、光照变化以及在嵌入式平台上部署等具体挑战，文章开发了一种集成增强型 YOLOv8-MT 模型与 RGB-D 相机的轻量化视觉检测系统。YOLOv8-MT 模型通过引入双路径架构瓶颈注意力模块，以提升遮挡环境下的小目标番茄检测能力。采用共享卷积与自适应缩放层的轻量化检测头，在减少参数的同时增强多尺度特征表征能力。通过将 2D 检测中心与深度数据对齐，无需计算密集的点云即可实现番茄的实时三维定位。实验表明，该系统在嵌入式平台上实现 51 帧 /s 推理速度的同时，检测精度达到 93.9%（提升 2.5%），参数量减少 33.3%。最大三维定位误差分别控制在 11 ms、15 ms、14 ms 以内，满足了视觉检测系统对轻量化、实时性与准确性的综合要求，为自动化番茄采摘提供了技术支持。

关键词 番茄检测；实时检测；深度相机融合；自动化番茄采摘

doi：10.3969/j.issn.1672-9528.2026.02.022

0 引言

番茄作为全球广泛栽培的蔬菜作物[1]，在传统采收方式下面临劳动力短缺与效率低下的双重挑战。在此背景下，番茄采摘机器人应运而生，而视觉系统在其中起着关键作用。通过精准分析番茄的形态特征与三维空间信息，视觉系统为机械臂作业提供重要的决策依据[2]。针对番茄果实生长密集、相互遮挡、质地脆弱等特点，视觉系统需具备较高的检测精度，以实现精确的成熟度判别与空间定位[3]，最终支撑不同成熟度番茄的选择性采摘与分级管理。

视觉识别系统的设计主要基于计算机视觉技术。传统方法通常依赖轮廓、颜色和纹理等手工特征进行果实识别。Liu 等[4]开发了一种适用于常规彩色图像的番茄自动识别算法，实现了 94% 的检测准确率。Li 等[5]提出通过果实颜色和果实目标轮廓重建的三维模型检测方法进行果实识别。在自然环境中，光照角度的变化易导致颜色特征提取失效，检测目标的复杂背景和果实形态的多样性则使轮廓提取、几何拟合与实际目标的对应性较弱。此外，这些方法的设计过程通常较为繁琐，难以同时检测植物不同发育阶段生长特征差异显著的性状。

随着计算机视觉的快速发展，深度学习技术已成为果实识别的主流方法[6]。相较于传统算法，深度学习方法能够从数据中自主学习和提取更多样化的特征，展现出更优异的鲁

1. 武汉轻工大学 湖北武汉 430023

参考文献：

[1] 邵华，李军 . 基于移动边缘计算的无人机应急场景中绿色通信方案研究 [J]. 中国军转民 ,2025(10):53-54.

[2] 申立军，韦智珑 . 基于边缘计算存储的无线电监测数据“采存管用”标准化技术研究 [J]. 中国无线电 ,2025(5):75-78.

[3] 肖琛，牟云飞 .5G 网络下边缘计算任务卸载优化策略研究 [J]. 软件 ,2025,46(5):175-177.

[4] 黎志生，覃健诚，王磊 . 基于边缘计算和云融合构建智慧农用集群系统 [J]. 自动化与仪表 ,2025,40(5):1-6.

[5] 蔡洁聪，国旭涛，陈思艺，等 . 基于边缘计算和机器学习的光伏面板热斑检测系统设计 [J]. 工业控制计算机 , 2025, 38(5): 65-66.

【作者简介】

王善程（1993—），男，河南信阳人，本科，助理工程师，研究方向：计算机。

陈玉林（1993—），女，河南信阳人，本科，助教，研究方向：计算机。

（收稿日期：2025-07-11 修回日期：2026-01-27）