 DOI: https://doi.org/10.12349/iser.v6i8.7007

Research and Implementation of PXE Network Booting Technology Based on OpenEuler Operating System

Ye Zhu

Yunnan Post and Telecommunications Engineering Co., Ltd., Kunming, Yunnan, 650233, China

Abstract

This study utilizes Docker containerization technology to develop an intelligent monitoring system integrating Zabbix and Grafana, exploring the construction path of modern operation and maintenance monitoring platforms. The research validates the value of containerization technology in enhancing system deployment efficiency, improving visualization capabilities, and optimizing resource utilization, achieving standardization and automation transformation of O&M processes. By deeply integrating traditional monitoring tools with containerization technology, a complete “data collection-analysis-processing-visualization” technical chain is established, providing a reference implementation framework for enterprises’ digital transformation in O&M.

Keywords

Docker containerization; Zabbix; Grafana; Intelligent O&M; Automated O&M; Monitoring Visualization;

基于容器化监控系统构建实践

朱烨

云南邮电工程有限公司，中国 · 云南 昆明 650233

摘　要

本研究基于Docker容器化技术，构建了Zabbix与Grafana集成的智能监控系统，探索了现代化运维监控平台的建设路径。研究验证了容器化技术在提升系统部署效率、增强可视化能力以及优化资源利用等方面的价值，实现了运维流程的标准化和自动化转型。通过将传统监控工具与容器化技术深度融合，构建了完整的"数据采集-分析处理-可视化展示"技术链条，为企业的运维数字化转型提供了可参考的实施框架。

关键词

Docker容器化；Zabbix；Grafana；智能运维；自动化运维；监控可视化

1 研究背景

在数字化转型浪潮下，云计算、大数据、微服务等新兴技术得到了广泛应用，与之相关的设备数量也在迅速增加，其中存在大量的不同厂家的服务器设备、安全设备、网络设备，不同设备对应的巡检命令也存在差异。传统依赖人工的对系统进行巡检的监控体系面临前所未有的挑战。随着信息系统复杂度飙升，需要进行监控的指标从单一物理设备，扩展到应用进程、虚拟化资源、服务组件等多维度层次。这对监控系统可监控类型、扩展性提出了更要的要求。在这情况下，Zabbix作为目前较为主流的开源监控系统解决方案，在监控领域得到了较为广泛的应用。在此背景下，研究如何将传统监控工具与容器化技术深度融合，实现监控系统的现代化转型，具有重要的理论意义和实践价值。

【作者简介】朱烨（1997–），男，中国云南曲靖人，本科，从事信息通信工程技术研究。

2 研究现状

2.1 人力资源配置与运维效率问题

传统运维模式高度依赖人工操作，运维人员需要远程登录管理界面对数以千计的设备，执行指令检查当前设备状态，来完成日常巡检和故障排查。这种人工密集型运维方式不仅需要投入大量专业技术人员，还存在操作效率低下、人工疏漏风险等问题，难以保证运维质量的稳定性和可靠性。

2.2 系统健康状态的可观测性缺陷

现有运维体系缺乏有效的预警机制和主动监测手段，对系统健康状况的掌握往往具有滞后性。运维人员只能在故障发生后进行被动处置，无法实现事前预警和事中快速响应，这种 " 事后救火 " 式的运维模式严重制约了系统可靠性的提升。

2.3 异构环境下的管理碎片化

数据中心基础设施由多厂商、多类型设备构成，各厂商设备采用不同的管理接口和工具，形成了 " 烟囱式 " 的管理架构。运维人员需要同时操作多个独立的管理系统，导致运维工作碎片化，增加了管理复杂度和操作负担。

2.4 运维数据治理能力不足

传统监控系统缺乏完善的日志收集和分析机制，关键设备的运行状态、报错信息等历史数据未能有效留存，不仅影响故障诊断效率，也不利于进行系统性的事后分析和持续优化。

现有对于现有监控系统讨论较多的探讨的角度，在系统指标方面，主要考虑系统运行指标、日志记录两方面来进行监控体系的理论框架构建；在可扩展性方面，主要涉及兼容设备类型、是否支持二次开发和自定义模块；在设备监控难易程度，主要衡量是否能够使用模块化、批量化对设备进行管控；在数据可视化方面，监控数据的是否能够多维呈现也是关键因素，此外，出于维护成本考虑，还需要对监控系统是否开源进行筛选。目前市场上比较主流的监控软件，Zabbix 和 Prometheus 能够满足部分的条件，但是部署过程繁琐，存在一定技术壁垒，急需一个可移植、便携式、具备出色的数据可视化能力的监控手段。

3 Docker 容器化技术

3.1 容器化技术概述

在现代软件开发与自动化测试领域中，Docker 容器化技术通过提供一种轻量级、可移植的解决方案，使得软件应用可以在几乎无需修改的情况下在不同的环境中运行，从而优化了开发流程，极大增强了测试工作的效率与一致性。[1] 与传统的虚拟机技术相比，容器化技术具有更轻量、更快速、更便携的特点。Docker 利用 Linux 内核的 cgroups 和 namespace 等特性，实现了进程级的资源隔离，使得应用程序及其依赖可以被打包成一个标准化的单元，即容器镜像。

3.2 Docker 的核心优势

在应用交付领域，在保障 Docker 容器保障了环境的一致性的前提下，还具有实现快速启动的功能，通常可以在秒级完成部署和扩展，这对于需要弹性伸缩的监控系统尤为重要；容器镜像的版本控制机制使得系统升级和回滚变得更加可控和可靠。此外，容器化还支持微服务架构，可以将复杂的监控系统拆分为多个独立的服务单元，提高系统的可维护性。

3.3 容器化在监控平台中的使用

本方案采用 Docker 容器化技术对 Zabbix 和 Grafana 等核心组件进行封装部署。具体实现包括：为每个服务组件构建专用的 Docker 镜像，通过 Dockerfile 定义构建过程和环境配置；使用 Docker Compose 定义和管理多容器应用的服务编排；利用 Docker Volume 实现配置文件和监控数据的持久化存储；通过 Docker 网络实现容器间的安全通信。这种容器化的部署方式不仅简化了环境配置的复杂度，还提高了系统的可移植性和可扩展性。

容器化技术的引入为监控平台部署带来了显著变革：部署效率提升方面，新节点的部署时间从小时级缩短至分钟级；资源利用率方面，容器共享主机操作系统内核，较虚拟机节省了大量系统资源；系统可靠性方面，容器提供了进程隔离机制，避免了应用间的相互干扰；运维标准化方面，容器镜像成为交付的标准单元，实现了开发、测试、生产环境的一致性。这些优势使得容器化技术成为构建现代化智能运维平台的理想选择。

4 Zabbix 容器化部署实践

运维监控系统的实现过程是，将基础平台和业务系统中所涉及的硬件资源信息、基础组件信息、应用软件信息等统一纳入运维监控平台，并进行指标的规范、收集及统一集中存储。以可用性指标为基础，逐步增加服务质量相关指标。实现系统运维监控的规范化和故障告警处理的智能化。[2]

4.1 Zabbix 监控

4.1.1 系统架构概述

Zabbix 作为开源监控领域的代表性解决方案，采用分布式架构设计，具有良好的扩展性和灵活性。该系统基于客户端 - 服务器模型构建，核心组件包括管理端的 Zabbix Server 和被监控端的 Zabbix Agent。Zabbix Server 可以通过 SNMP、Zabbix agent、ping、IPMI 等方法对远程服务器和网络设备进行状态监控和数据采集。通过 B/S 模式在 Web 端进行信息呈现和系统配置。[3] 这种模块化架构设计使得系统可以根据监控规模进行横向扩展，同时支持通过 API 接口与第三方系统集成，为二次开发提供了便利条件。

4.1.2 数据采集机制

Zabbix 提供两种基本的数据采集模式：在主动模式下，Agent 端定期向 Server 推送监控数据，这种模式适用于需要实时性较高的监控场景；被动模式则由 Server 端主动发起数据请求，更适合网络环境受限的情况。本方案选择主动模式，主要考虑到其具有更好的实时性和网络适应性。除 Agent 采集方式外，系统还支持通过 SNMP、IPMI 等标准协议获取设备数据，实现对网络设备、服务器硬件等基础设施的全面监控。

4.1.3 数据处理流程

Zabbix 的数据处理流程遵循典型的监控系统架构：首先由各采集端获取原始监控数据，然后传输至 Server 端进行预处理和阈值分析。系统内置的告警引擎会对异常数据进行实时判断，触发相应的告警规则。所有监控数据最终持久化存储到后端数据库，为历史数据查询和分析提供支持。这种分层处理架构既保证了实时监控的响应速度，又确保了历史数据的完整性和可靠性。

4.1.4 跨平台支持能力

Zabbix 具有出色的跨平台兼容性，可以无缝监控 Linux、Windows 等主流操作系统环境。通过标准协议支持和丰富的插件机制，系统能够对接各类网络设备、安全设备以及中间件、数据库等应用系统。这种全方位的监控能力使其成为构建统一运维平台的理想选择，有效解决了异构环境下的监控碎片化问题。

4.2 Zabbix 容器编排架构设计

4.2.1 容器服务定义

本研究采用 Docker Compose 3.5 版本规范定义 Zabbix Server 服务。容器基于官方 zabbix/zabbix-server-mysql:centos-6.0-latest 镜像构建，通过 container_name 参数明确指定服务实例名称。为确保服务的高可用性，设置 restart 策略为

always，使容器在异常退出时能够自动重启。服务暴露 10051 端口作为主要通信接口，实现与 Zabbix Agent 等组件的监控数据传输。

4.2.2 存储卷配置

通过 volumes 配置实现了多层次的数据持久化方案：首先，将主机时区文件挂载至容器，确保日志时间戳的准确性；其次，对关键目录进行细粒度挂载，包括 alertscripts（告警脚本目录）、externalscripts（外部检查脚本目录）等运维核心目录设置为只读模式，保障配置安全性；最后，为 snmptraps 等需要频繁写入的目录配置读写 (rw) 权限，并使用命名卷 snmptraps 实现数据持久化。这种分层存储设计既保证了配置文件的版本控制，又满足了运行时的数据持久化需求。

4.2.3 资源限制与安全配置

在资源控制方面，通过 deploy.resources 设置 CPU 和内存的使用限额和预留值，确保服务的稳定运行。同时配置 ulimits 参数调整进程数和文件描述符限制，提升服务在高负载情况下的稳定性。安全方面采用 env_file 引入外部环境变量文件，并通过 secrets 机制管理 MySQL 等敏感信息的访问凭证，避免密码硬编码带来的安全风险。

4.2.4 服务依赖与网络架构

服务启动顺序通过 depends_on 参数控制，确保 MySQL 服务就绪后再启动 Zabbix Server。网络配置方面，将服务接入 zbx_net_backend 和 zbx_net_frontend 两个自定义网络，并设置多个网络别名（如 zabbix-server、zabbix-server-mysql 等），既实现了前后端流量的隔离，又提供了灵活的服务发现机制。这种网络设计支持后续服务扩展时的无缝集成，为构建分布式监控系统奠定了基础。

5 Grafana 容器化集成

5.1 Grafana 介绍

Grafana 是一种使用 Go 开源监控平台，可以实时监控信息系统使用情况，具有查看历史数据、数据分析等功能。，其核心功能是将时间序列数据库（TSDB）中的监控数据转化为直观的可视化图表。该平台采用模块化架构设计，主要由三大组件构成：数据源模块支持包括 Zabbix、Prometheus 等在内的多种数据存储系统的接入；仪表板模块提供灵活的面板布局管理功能；可视化面板模块则负责具体的数据呈现。这些组件的协同工作使得 Grafana 能够实现多维度的监控数据展示与分析。

在功能特性方面，Grafana 具备以下核心能力：快速和灵活的客户端图形具有多种选项。面板插件为许多不同的方式可视化指标和日志 [4]；智能告警系统能够在监控指标异常时触发多通道通知；动态仪表板功能通过模板变量实现监控视图的灵活配置；此外，平台还支持多数据源的混合展示以及基于注释的事件标记功能。这些特性共同构成了 Grafana 作为业界领先监控可视化解决方案的技术基础。

本研究中的 Grafana 通过容器化集成构建，在便携安装和移植的同时与 Zabbix 数据源的深度集成，平台实现了从底层硬件到上层应用的立体化监控体系，为运维决策提供了全面的数据支撑。这种实现方式既保留了 Zabbix 在数据采集方面的优势，又充分发挥了 Grafana 在数据可视化领域的特长。

5.2 Zabbix 可视化功能局限性分析

Zabbix 原生可视化模块存在三个主要技术瓶颈：首先，在数据渲染效率方面，其图表生成机制采用传统的服务端渲染方式，导致响应延迟显著；其次，数据源适配能力有限，仅支持单一监控系统的数据接入，难以实现多源数据的关联分析；最后，告警管理功能相对基础，缺乏灵活的可视化配置选项。这些架构层面的限制在混合云环境下的综合监控场景中表现得尤为明显。

5.3 Zabbix+Grafana 运维平台容器化部署方案设计与特点分析

本研究采用声明式部署方法实现 Grafana 的容器化集成。通过预置配置目录和标准化环境变量，构建了可复用的部署模板。关键技术实现包括：配置文件的持久化存储方案确保系统参数可维护性；插件预装机制实现功能扩展；基于容器网络的内部通信架构保障数据传输安全性。这种部署方式有效解决了传统安装模式的环境依赖问题，大幅提升了部署效率。

Grafana 平台通过三大技术特性显著提升了监控可视化能力：其多数据源融合架构支持同时接入 Zabbix 指标数据和日志系统的业务数据；智能告警引擎实现了基于可视化阈值的动态告警规则生成；丰富的模板库为不同业务场景提供了开箱即用的监控方案。性能测试表明，该方案使数据查询效率得到显著提升。

6 结语

本研究基于 Docker 容器化技术构建了 Zabbix 与 Grafana 集成的智能监控系统，探索了现代化运维监控平台的建设路径。在理论层面，验证了容器化技术在提升系统部署效率、增强可视化能力以及优化资源利用等方面的价值。通过将传统监控工具与容器化技术深度融合，实现了运维流程的标准化和自动化转型。在实践层面，构建了完整的 " 数据采集 - 分析处理 - 可视化展示 " 技术链条，为企业的运维数字化转型提供了可参考的实施框架。

7 未来展望

面向智能运维的发展趋势，未来研究可从结合 Ansible 等自动化工具构建完整的 " 监控 - 诊断 - 修复 " 闭环体系，实现从故障检测到自动修复的智能化运维流程；对于告警内容、时间、频率等数据，引入 AI 手段，对信息系统中的故障进行预测和预防工作。

参考文献

[1] 钟盈炯.基于Prometheus+Grafana实现新华全媒新闻服务平台统一运维监控[J].中国传媒科技,2023,(01):154-158.

[2] 王曦鋆.基于Docker容器化技术的软件工程自动化测试研究[J].网络安全和信息化,2025,(04):110-112.

[3] 谷胜元.铁路信号模拟试验中采用计算机软件+采驱板替代模拟盘的研制[J].科技创新与应用,2020,(18):86-87.

[4] 黄静,陈秋燕.基于Prometheus + Grafana实现企业园区信息化PaaS平台监控[J].数字通信世界,2020,(09):70-72.