计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(5):57−66 [doi: 10.15888/j.cnki.csa.009070]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向 Kubernetes 的容器立体化监控系统①

曾　缘, 李　轲, 窦　亮

(华东师范大学 计算机科学与技术学院, 上海 200062)
通信作者: 窦　亮, E-mail: ldou@cs.ecnu.edu.cn

摘　要: 容器监控是保证容器基础设施正常运行的核心要素之一, 然而当前容器监控角度较为单一, 尚缺少直观有效的方法协助运维人员快速定位业务容器异常原因. 本文设计并实现了一个面向 Kubernetes 的容器立体化监控系统, 通过监控指标相关性分析, 把相关性较强的指标作为立体化监控的核心指标提供给运维人员, 更好地实现对容器的全局统筹监控.

关键词: Kubernetes; 容器监控; 立体化监控; 相关性分析

引用格式: 曾缘,李轲,窦亮.面向 Kubernetes 的容器立体化监控系统.计算机系统应用,2023,32(5):57–66. http://www.c-s-a.org.cn/1003-3254/9070.html

Multi-dimensional Container Monitoring System for Kubernetes

ZENG Yuan, LI Ke, DOU Liang

(School of Computer Science and Technology, East China Normal University, Shanghai 200062, China)

Abstract: Container monitoring is one of the most important factors in ensuring the proper functioning of container infrastructure. However, the dimensions of current container monitoring are single, and there is a lack of intuitive and effective methods to assist operation engineers in locating the cause of anomalies in the business container. Therefore, this study proposes a multi-dimensional container monitoring system for Kubernetes. Through the correlation analysis of monitoring metrics, it provides highly relevant metrics as the core metrics of multi-dimensional monitoring for operation engineers and thus better implements comprehensive container monitoring.

Key words: Kubernetes; container monitoring; multi-dimensional monitoring; correlation analysis

近年来, 微服务架构和容器技术广泛应用于各大云计算平台中, 成为云原生技术的重要组成部分. 以 Docker[1] 为代表的容器虚拟化技术日益盛行, 促成了容器编排工具的发展. Kubernetes (K8s)[2] 凭借其强大的容器编排能力和轻量开源的特点成为当前云原生时代的主流容器集群操作系统. K8s 是谷歌内部大规模资源调度系统 Borg 的开源改进版本, 提供了容器应用部署、规划、更新、维护的机制. 随着 K8s 与容器的应用越来越广泛, 如何有效监控 K8s 容器逐渐成为关注的热点.

相对于传统的主机监控, K8s 容器监控面临着许多新的挑战. 容器短暂的生命周期提高了对监控实时性的要求, 由于容器本身的环境是与外界隔离的, 一旦容器消亡, 内部所有数据消失, 用户便无法查看当前已消亡容器的监控数据, 而容器的生命周期往往是短暂的, 有报告显示 20% 的容器存活时间少于 10 s, 49% 的存活时间少于 5 min[3], 这对容器监控的采集频率提出了更高的要求. 微服务架构的复杂性也大大增加了监控难度, 一方面是监控数据量大幅度增加, 原本提供一个服务的应用程序转变为由多个微服务共同提供, 大

① 基金项目: 上海市 2022 年度科技创新行动计划 (20511102502)
收稿时间: 2022-10-04; 修改时间: 2022-11-18; 采用时间: 2022-11-30; csa 在线出版时间: 2023-02-17
CNKI 网络首发时间: 2023-02-17

大提高了实例的数量与实例间网络通讯的流量; 另一方面是监控目标的频繁变化, 由于弹性伸缩的加持, 服务可以按照需求实现快速扩缩容, 并且这种情况会时刻发生, 这对监控的高可用性而言是一个巨大的挑战. 此外, 微服务架构也使得定位故障变得更加困难. 微服务架构的松耦合度使得每个组件可以进行独立的开发维护, 但是对于故障定位的问题并没有实质性的改善. 当故障出现的时候, 用户仍然需要从众多的服务中找出导致该故障的服务, 然后再进一步确认故障产生的原因.

除了应对以上挑战, 业界对 K8s 容器监控工具还有着全局统筹监控的迫切需求. 在实际应用中, K8s 业务容器的监控指标需要涉及资源消耗、网络流量访问、容器健康状况等多个维度, 当业务容器提供服务时, 往往会因为生产环境或是访问压力的变化引起多项指标的浮动, 说明了某些指标间存在较为密切的关联, 然而目前对这种关联性的认识往往仍依赖于专业人员的运维经验, 缺乏有效的工具协助. 当业务容器出现服务异常时, 有经验的运维人员能够根据异常现象推断出异常出现的原因, 并查看对应的监控指标的变化情况, 及时采取措施解决问题, 反之则可能束手无策. 如果能够将存在密切关系的数据进行关联, 在监控时给出相关性较强的指标, 可以使得指标监控更加立体化, 从而降低运维人员发现容器异常原因的困难程度, 协助运维人员完成立体化的全局统筹监控.

针对上述挑战和需求, 本文设计并实现了一个面向 K8s 的容器立体化监控系统, 一方面对 K8s 容器集群的 CPU、内存、网络和存储资源进行监控, 另一方面对容器网络流量进行实时全量采集, 得到反映容器对外提供服务性能情况的网络流量计算指标; 通过数据相关性分析, 给出两个角度之间相关性较强的指标, 使得指标监控更加立体化, 能够帮助运维人员更加快速准确地发现导致服务异常的原因, 实现立体化的全局统筹监控.

1 相关工作

目前, 业界出现了许多适配容器的监控工具, 典型的有 Heapster[4]、cAdvisor[5]、Prometheus[6] 等. Heapster 是 K8s 的原生集群监控方案, 但在 K8s 1.12 版本中 Heapster 被弃用, 官方鼓励用户改用 metrics-server 或是第三方监控解决方案取而代之. cAdvisor 是 Google 开源的一款用于监控容器运行状态的可视化工具, 可以自动收集容器 CPU、内存、网络、存储等指标的使用情况, 并且对外提供原生的 API 接口. cAdvisor 支持本地 Docker 容器, 但并不提供长期存储与分析的功能. 目前, cAdvisor 已作为 K8s 的监控组件内嵌到 K8s 中. Prometheus 是一款开源的监控告警框架, 属于新一代云原生监控系统, 提供了监控数据收集、存储、处理可视化和告警的一套监控解决方案. 据文献 [3] 统计, 作为当前主流的开源容器监控工具, 在所有的用户自定义监控指标中, 平均有 62% 由 Prometheus 提供. 对于监控对象, Prometheus 支持静态文件配置与动态发现两种方式发现监控对象, 并且能够自动完成监控数据的采集. 然而, 目前 Prometheus 提供的监控方式, 对于多集群的情况适配性不佳. 除此之外, 一些传统的服务器监控工具, 如 Zabbix[7] 和 Nagios[8]也开始提供 K8s 相关的监控插件[9,10], 通过容器化方式部署于集群内, 能够获取集群的核心组件列表与部署状态.

在学术领域, 国内外关于容器监控也有一定研究. 张松等人[11] 提出了 3 种数据的收集方式: 通过 Docker 的 API 接口、在容器中安装监控代理和通过虚拟文件系统获取容器的运行数据. 徐鑫辰[12] 设计研究了边缘计算系统下的 Docker 监控交互平台, 实现了 Docker 容器的实时监控、增删容器、控制容器资源等功能. Hong 等人[13] 针对提供 5G 网络服务的 M-CODE 平台设计实现了一款监控网络功能虚拟化容器的监控系统, 可以获得容器内的计算资源使用情况和数据流量负载. Chang 等人[14] 基于 Heapster、InfluxDB、Grafana 设计了一款面向 K8s 的监控平台, 用于实现集群的动态资源调配.

虽然业界已在 K8s 容器监控方面做了不少工作, 然而现有工作的监控角度较为单一, 一般仅从资源角度进行监控, 缺乏对各个指标的相关性分析, 不利于运维人员在业务应用出现问题 (尤其是性能问题) 时迅速排查问题原因[15]. 因此, 本文提出容器指标立体化监控的理念, 对容器网络流量进行实时全量采集, 得到反映容器对外提供服务性能情况的网络流量计算指标, 结合容器集群资源监控指标, 分析不同维度指标间的相关性, 为反映不同状态的指标建立联系, 实现容器级别的立体化监控. 需要说明的是, 尽管从 K8s 容器集群资源监控指标中也可以获取网络指标, 然而那些指标不足以直接反映容器对外提供业务服务的性能情况.

2 监控指标数据相关性分析实验

本节选取容器资源指标与网络流量计算指标作为实验对象, 通过计算业务某一段时间内的容器资源指标与网络流量计算指标的相关系数, 分析业务容器的网络流量计算指标与资源指标之间的关系, 提取相关性较强的指标, 为后续协助用户快速定位异常原因和实现立体化监控提供依据.

2.1 实验流程

本实验使用网络流量监控工具 NetSensor[16] 采集网络流量并实时计算指标, 并使用之前提出的多集群资源监控方案[17] 采集容器资源指标. NetSensor 由控制器与采集探针两部分组成, 采集探针采集目标的网络接口, 进行抓包分析并生成指标数据; 控制器用于收集探针采集到的指标数据. 采集探针通过二进制软件的形式打包在 Docker 镜像中, 以 DaemonSet 形式部署在 K8s 集群中, 在集群的每一个节点上部署一个探针 Pod; 控制器可以运行在能与 K8s 集群进行网络通讯的虚拟机或物理机上, 也可以直接运行在 K8s 集群所处的虚拟机或物理机上. 本实验采用一个电子商务销售系统“月光茶人”作为测试业务, 以容器的方式部署在 K8s 集群中, 并通过 Apache JMeter 来模拟用户访问业务容器产生的网络流量.

NetSensor 采集到的网络流量计算指标数量多达 70 多种, 本实验选取了部分关键指标用于相关性分析, 可以较为全面地反映容器对外提供服务的性能情况, 见表 1. CPU、内存以及存储相关的资源指标见表 2. 在计算时, 将同一组件的资源数据与网络流量数据基于采集时间进行关联形成数据矩阵, 计算相关系数得到相关系数矩阵, 最后将相关系数矩阵绘制成相关系数图示.

在相关性分析实验过程中, 严格控制测试业务容器的网络流量访问, 除了集群内部必要的网络通讯以及 JMeter 模拟用户产生的网络流量 (使用梯度加压的方式) 之外, 不会对业务容器产生额外的流量. 本文使用了斯皮尔曼相关系数作为指标相关性的评判依据. 由于监控系统采集到的指标通常带有时序特征, 在现实生活中, 时序数列通常具有趋势性、季节性、周期性以及随机性[18], 因此用于表示线性关系的皮尔逊系数并不适用. 另外, 鉴于相关系数只有相对意义, 除了给出相关系数之外, 相关性分析实验还给出了用于相关系数的测试样本数量以及对应的临界相关系数, 用于标识相关性强弱的边界, 当相关系数的绝对值大于临界相关系数, 则表明相关关系显著; 否则表示相关关系不显著. 分析中的相关系数保留两位小数, 则显著性因子 α=0.01.

表 1 用于相关性研究的容器网络流量计算指标详情

中文名称	指标含义
主机IP	集群组件 (节点、Pod) 的IP地址
总流量	组件发送与接收流量之和
总速率	组件发送与接收速率之和
总数据包	组件发送与接收数据包之和
总包率	组件发送与接收包率之和
有效载荷	携带TCP载荷的总数据量
负载数据包	携带TCP载荷的总包量
字节收发比	接收总流量/发送总流量
数据包收发比	接收总包数/发送总包数
丢包总数	组件发送与接收丢包数之和
请求量	组件收到的请求总数
重传数量	组件数据包重传总数
重传率	组件数据包重传率
总重置数	组件网络连接总重置数
网络延迟	TCP三次握手的时间
响应时间	组件收到第一个数据包应答的时间

表 2 用于相关性分析的容器资源指标详情

资源类型	中文名称	英文名称
CPU	CPU使用核数	usage_cores
	CPU使用率	usage_cores_percent
	CPU使用时间	usage_cores_seconds_period
内存	RSS内存使用率	rss_percent
	内存使用率	usage_percent
	工作集内存使用率	working_set_percent
存储	存储容量使用率	used_amount_percent
	INODE使用率	used_inodes_percent

2.2 实验内容与结果分析

由于月光茶人系统的业务容器有多个接口, 为了控制变量, 实验仅通过访问“/httpclient4/listProduct.app-trace”接口获取系统的商品列表.

为了研究业务容器网络流量计算指标与资源消耗指标的相关性, 需要分别研究集群内部网络流量和外部网络流量对业务容器的资源消耗的影响.

(1) 集群内部网络流量对业务容器资源消耗的影响: 首先, 在 K8s 集群中部署月光茶人系统, 不进行任何网络访问, 持续采集其消耗的网络流量计算指标和资源消耗指标. 研究闲置状态下月光茶人业务容器的指标在长期和短期两种情况下的相关性, 持续采集时间分别为 6 h、0.5 h, 选取数据量分别为 360 条、30 条, 显著性因子 α=0.01, 相关系数临界值分别为0.135 602 和 0.462 892.

长期情况下的相关系数热力图见图 1, 图中所有值均未超过临界相关系数, 说明从长期来看, K8s 集群内部网络流量对业务容器的资源消耗无显著影响. 短期情况下相关系数变化见图 2, 图中虚线表示相关系数临界值, 实线表示短期指标相关系数, 在两条虚线中间表示相关性不显著, 在两边则为相关性显著. 由此看出, 在绝大部分时间内, 网络流量计算指标与资源指标的相关系数均在临界值之内, 说明了短期 K8s 集群内部网络流量对业务容器的资源消耗无显著影响. 综上所述, K8s 集群内部网络流量对业务容器的资源消耗无显著影响.

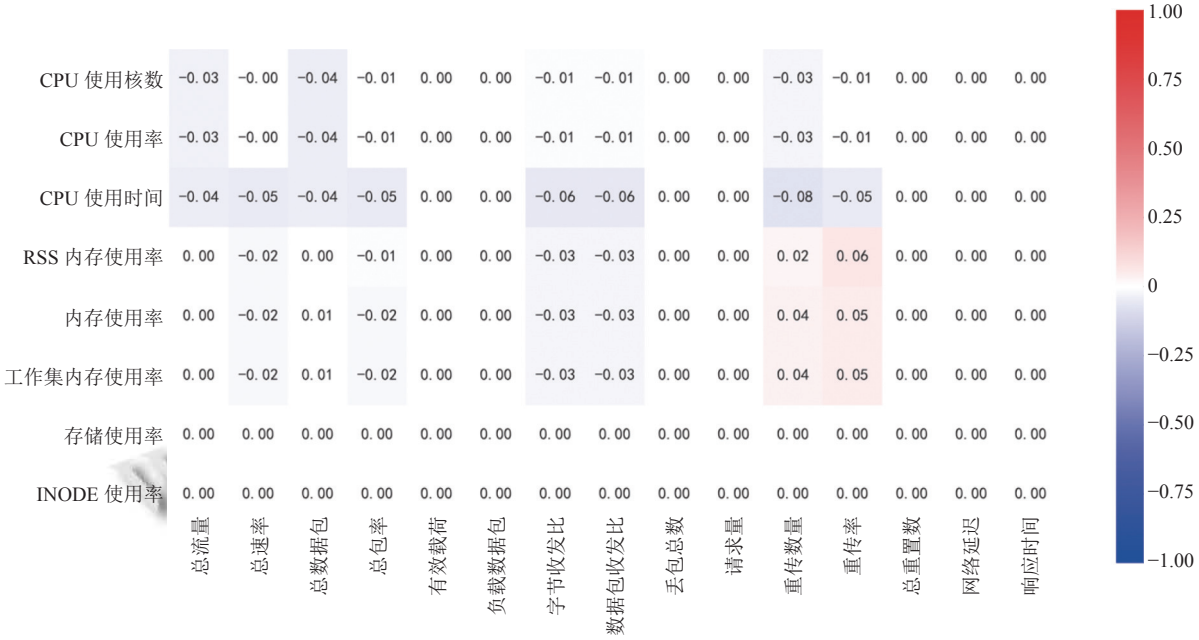

图 1 月光茶人系统闲置状态长期指标相关系数热力图

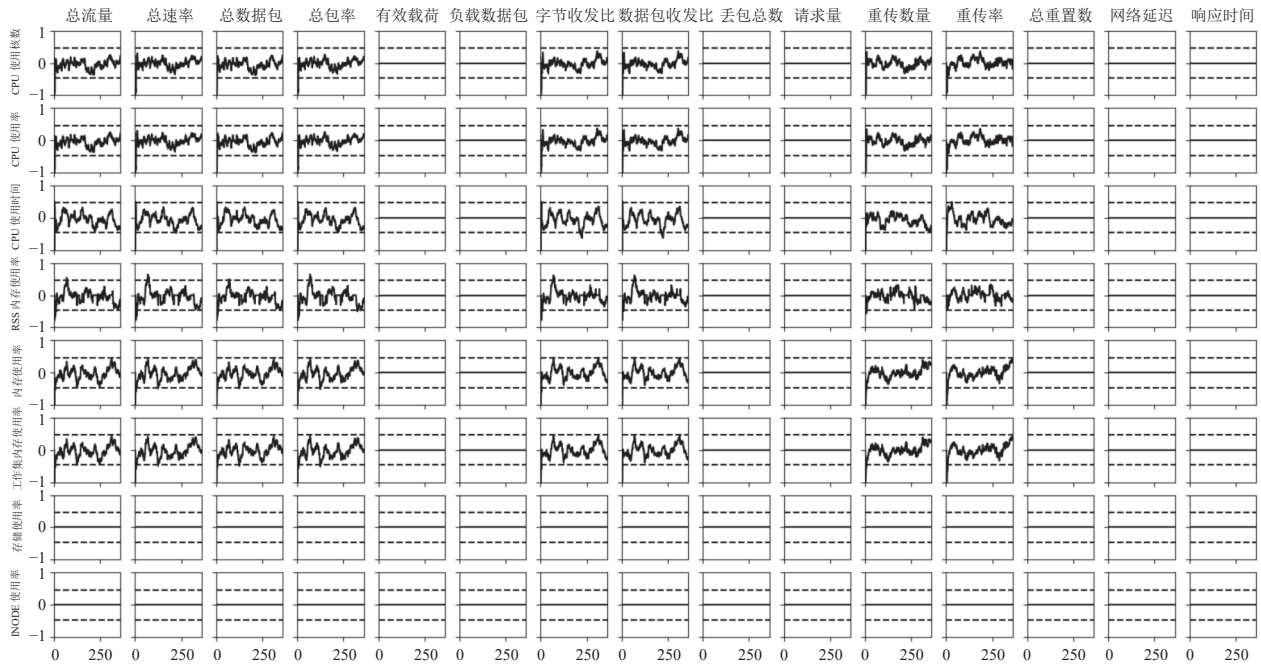

图 2 月光茶人系统闲置状态短期指标相关系数

(2) 外部访问产生的网络流量对业务容器资源消耗的影响: 首先, 使用 JMeter 进行阶梯式线性增压, 模拟用户访问月光茶人的商品列表页面, 起始并发数为 1, 终止并发数为 50, 持续 1440 min, 增压阶段数量为 100.

实验分别研究长期和短期两种不同情况下的数据相关性, 整个实验过程可以根据网络情况分为通畅阶段和拥塞阶段. 通畅阶段持续 247 min, 拥塞阶段持续 1 193 min.

对于长期情况的研究, 实验设置如下: 网络通畅状态下, 整体数据 247 条, 显著性因子 α=0.01, 相关系数临界值为 0.163 62; 网络拥塞状态下, 整体数据 1 193 条, 显著性因子 α=0.01, 相关系数临界值为 0.745188. 本文绘制了网络通畅状态下和网络拥塞状态下的相关系数热力图, 见图 3 与图 4. 由图可以看出, 网络通畅状态和网络拥塞状态下月光茶人网络流量与资源指标相关性出现了非常明显的区别. 我们分析认为: ① 当网络通畅时, CPU、内存各项指标与大部分网络流量计算指标呈现非常显著的正相关性; ② 当网络拥塞时, CPU、内存的相关指标大部分指标相关性大幅下降, 部分内存相关的指标甚至呈现负相关. ③ 实验中, 存储指标表现为与网络流量计算指标无关, 这只能说明实验中存储指标并没有直接因为网络流量的变化而产生变化, 导致这个现象的原因, 一是可能与业务容器本身的功能有关, 二是可能与 API 接口刷新数据有关, 有可能是因为接口刷新频率较低或者集群出现了相关故障导致这种情况的发生. 由此, 后面的研究只考虑网络流量与 CPU、内存指标之间的指标关联.

对于短期情况的研究, 实验设置如下: 截取长期研究实验中的一小时内实验数据, 数据量为 60 条, 显著性因子 α=0.01, 相关系数临界值为 0.330 104, 在网络通畅状态下和网络拥塞状态下的相关系数变化见图 5 和图 6.

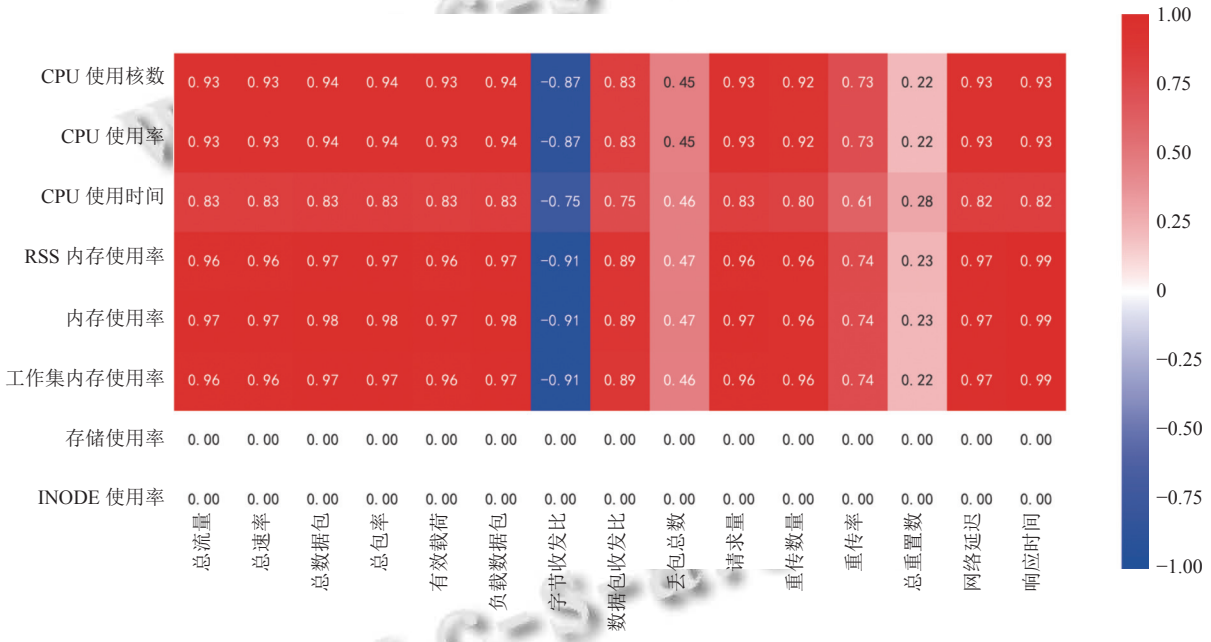

图 3 月光茶人系统网络通畅状态长期指标相关系数热力图

从短期指标中, 我们分析认为: ① 当网络拥堵状况加剧时, 大部分网络流量与资源指标的相关性的大体趋势是显著程度在不断地降低, 但是这种相关性会随着时间不断变化; ② 对于 CPU, 无论网络状况如何, 访问容器的网络流量和请求量与 CPU 资源的消耗始终呈现出一定的正向相关性, 这说明了 CPU 资源的消耗量可以作为访问容器网络流量情况的判断依据; ③ 对于内存, 无论网络状况如何, 访问容器的响应时间与内存资源的消耗始终呈现出非常强的正相关性, 因此当网络状况良好时内存消耗的变化情况可以作为判断是否能够正常提供服务的参考依据.

通过长期指标与短期指标之间的对比, 可以看出一些相关系数的出入. 当容器网络通畅时, 长期指标较短期指标能够提供更显著的相关性结论. 对于实际的应用而言, 短期指标计算量小, 能够及时提供指标间的相关性变化情况, 较长期指标在实际应用中更具有参考价值; 长期指标在理想条件下能够给出更明显的指标关系性质, 较短期指标而言更具有理论意义相关性变化情况, 较长期指标在实际应用中更具有参考价值; 长期指标在理想条件下能够给出更明显的指标关系性质, 较短期指标而言更具有理论意义.

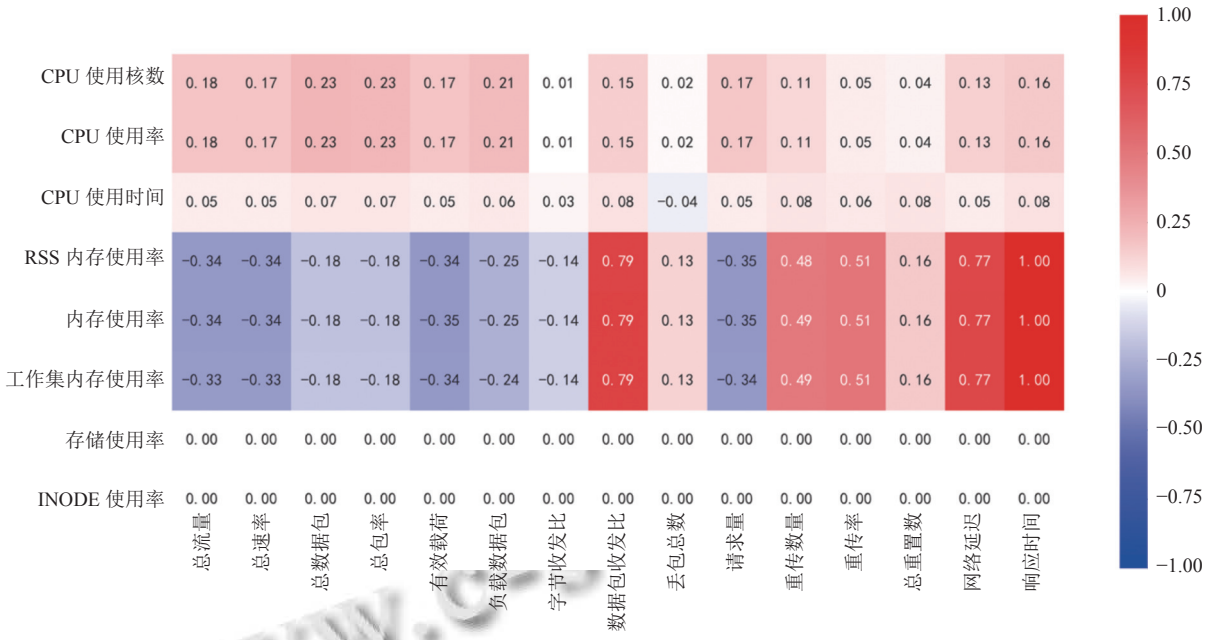

图 4 月光茶人系统网络拥塞状态长期指标相关系数热力图

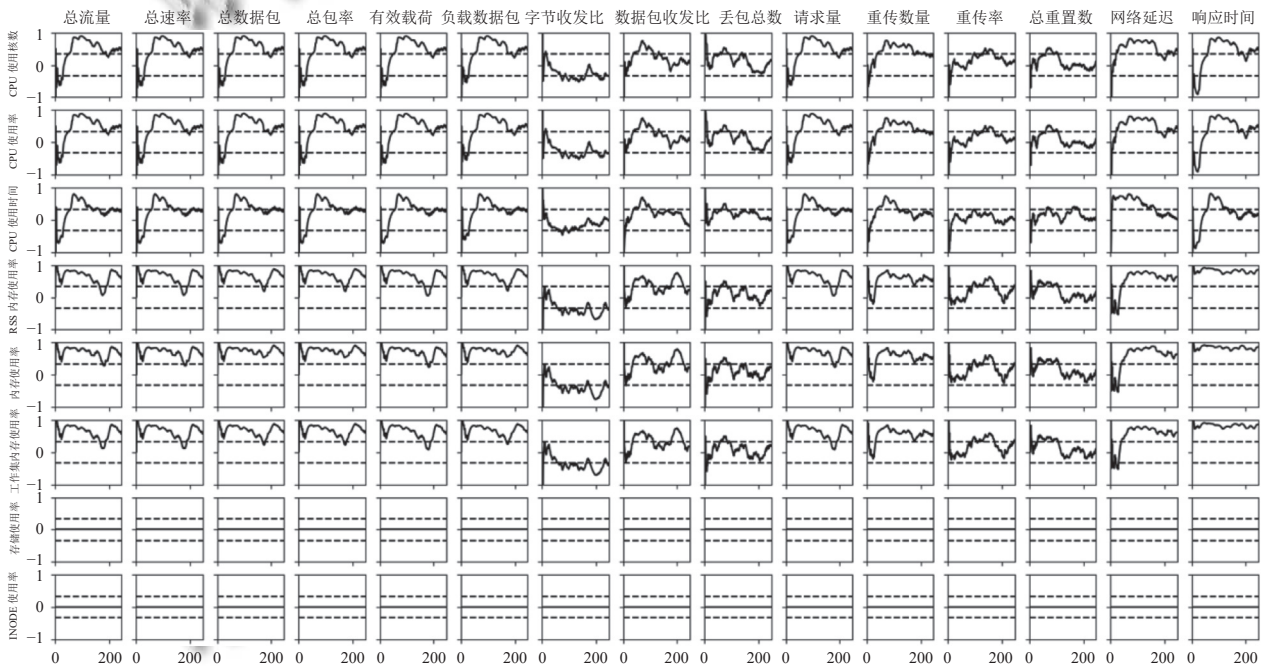

图 5 月光茶人系统网络通畅状态短期指标相关系数

本节的监控指标数据相关性分析实验验证了业务容器在提供服务时, 网络流量计算指标与资源指标存在一定的联系, 不但说明通过计算指标相关系数实现立体化监控这一思路的合理性, 还提取出“CPU 消耗-网络流量”“内存消耗-响应时间”两对相关性较强的指标. 通过相关系数长期与短期数据的分析, 也说明长期数据的结论并不能够完全代表业务容器实际工作时实时网络流量与资源指标之间的关系. 经过业界专家的确认, 以上的两对指标确实在相应的业务环境中有强相关性, 能够直观地反映容器的业务压力以及资源的消耗情况, 在一定程度上起到预警作用. 需要说明的是, 由于业务容器功能各不相同, 本节相关性分析结论无法代表所有业务容器, 在实际的业务环境中, 可以选择但不限于以上两对指标协助运维和监控.

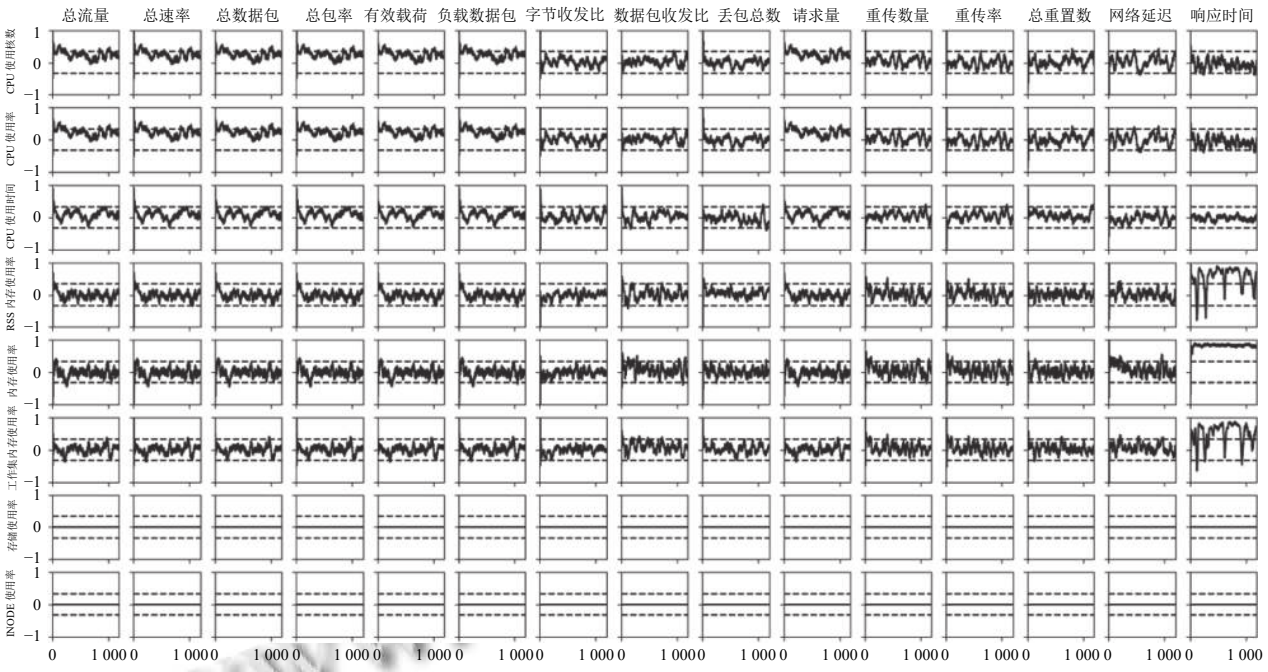

图 6 月光茶人系统网络拥塞状态短期指标相关系数

3 面向 K8s 的容器立体化监控系统

基于上述监控指标数据相关性分析, 本文设计实现了一个面向 K8s 的容器立体化监控系统, 从容器对外提供业务的视角对 K8s 集群的节点和 Pod 进行监控. 图 7 是系统总体架构图.

面向 K8s 的容器立体化监控系统分为数据采集层, 数据处理层和数据呈现层. 数据采集层负责采集 K8s 容器集群的网络流量以及容器节点和 Pod 的资源信息. 容器集群网络流量数据分为内部和外部两种, 通过在交换机做镜像的方式采集容器外部发送到容器内部的流量, 通过部署容器软探针的方式采集容器内部的交互流量. 使用已提出的多集群资源监控方案[17], 即基于 K8s 原生的 API 接口对 K8s 集群中所有节点和 Pod 的 CPU、内存、网络、存储资源指标进行采集. 数据处理层主要针对采集层上报的网络指标和资源指标进行加工处理和相关性计算分析. 这两部分数据的结构差异较大, 所以将容器网络流量数据保存到非关系型数据库 MongoDB 中, 而将容器资源数据保存到关系型数据库 PostgresSQL 中. 数据引擎对这两部分数据进行相关性分析, 预备好缝合的数据. 数据呈现层通过各种 API 接口为业务运维和 IT 运维等提供可视化工具支持, 建立立体化监控. 图 8 是立体化监控系统的部署原理图.

4 系统展示与性能测试

4.1 系统部署

以针对月光茶人的立体化监控系统部署为例进行说明, 共部署 3 套 K8s 集群, 每套 K8s 集群中均部署月光茶人业务容器. 监控系统与集群部署在 172.16 以及 10.168 两个网段内, 网段间可以相互通信.

4.2 系统展示

在立体化监控系统中, 运维人员可以通过可视化界面查询到节点和 Pod 的 CPU、内存、网络等指标信息以及资源指标与网络流量计算指标相关系数的热力图, 并且可以根据自己的需求设置资源指标类型以及日期范围. 基于相关性较强的“CPU 消耗-网络流量”“内存消耗-响应时间”两对指标, 系统给出了容器 CPU 使用率与总流量的指标展示以及内存使用率与相应时间的指标展示, 见图 9 和图 10, 图中为使用月光茶人系统的业务容器进行在线性增压访问下的真实数据.

图 9 和图 10 中的数据变化印证了相关性分析的结论, 这两对指标的相关性在图中有明显的体现: 当访问月光茶人的网络流量增加时, CPU 的使用量随着网络流量的增加而逐步增加; 当月光茶人无法正常提供服务时 (实际为服务延迟增大), 响应时间指标暴增, 内存指标也随之骤增. 由此可见, 网络流量计算指标对于容器资源使用的变化情况给出一个直观的解释, 用户

可以通过这种立体化监控的方式直观地了解容器的业务压力以及资源的消耗情况, 之后根据具体情况进行合理的配置, 这便是容器立体化监控的意义所在.

图 9 和图 10 也揭示了: 尽管资源问题使容器在短时间内变为无法正常提供服务的状态, 但是资源消耗的趋势以及网络指标的变化可以一定程度上预示了容器无法正常提供服务的可能. 通过立体化监控, 运维人员可以多维度判断容器出现异常的可能性, 便于预测容器可能出现的问题. 由于部署在节点上的容器共享节点资源, 随着节点部署容器数量的增加, 单从容器的资源使用率上往往不能够直接体现出资源问题. 依靠立体化监控提供的网络流量计算指标, 可以根据发生服务异常时业务容器的资源使用情况自定义告警机制, 使得资源告警针对性更强、效果更好.

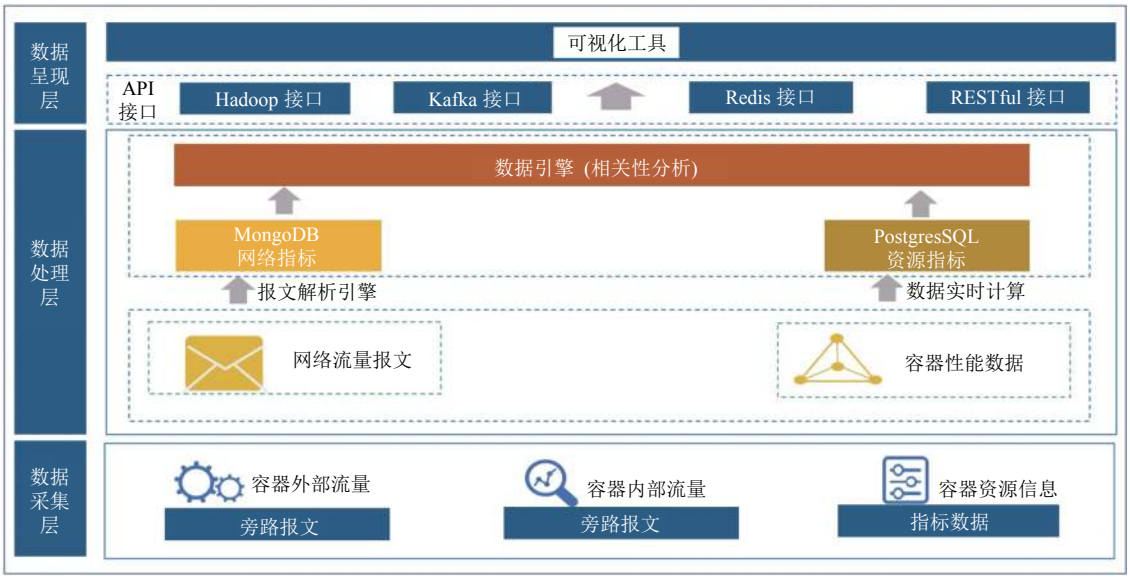

图 7 容器立体化监控系统总体架构图

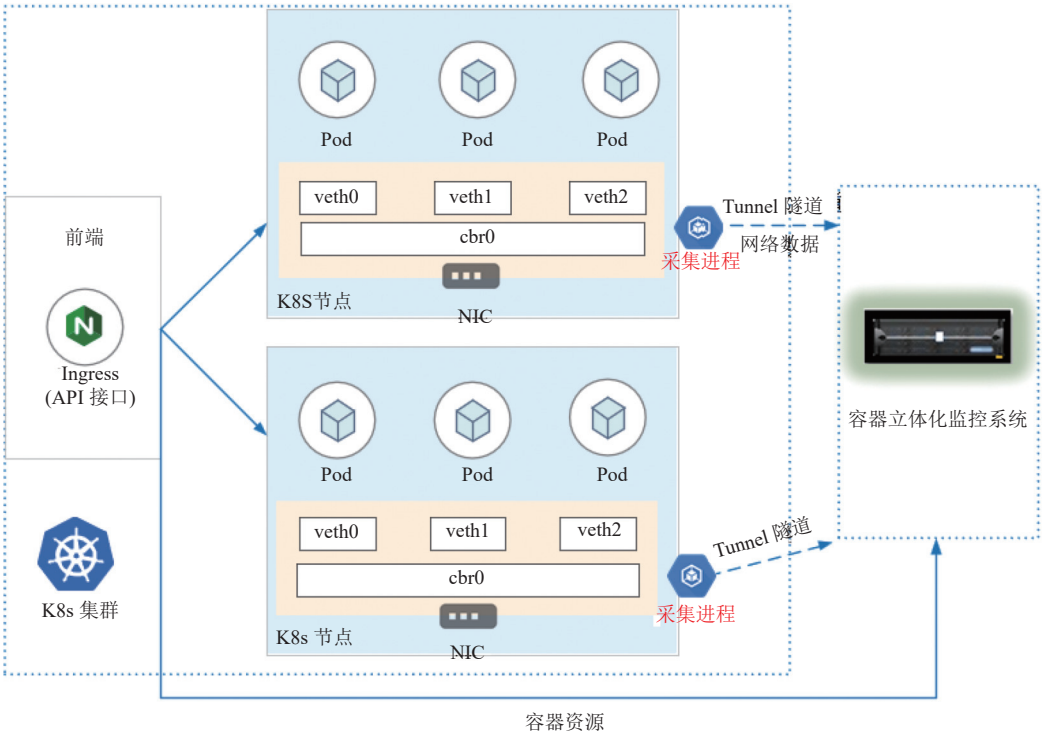

图 8 容器立体化监控系统部署原理图

相对于 Prometheus, 本文提出的立体化监控系统从资源指标与网络流量计算指标的不同角度更加全面地监控容器在集群中的运行状态, 由于系统并不需要在 K8s 集群中配置代理, 对 K8s 集群资源消耗也更小.

图 9 月光茶人系统线性增压访问下 CPU 使用率与网络流量的指标展示

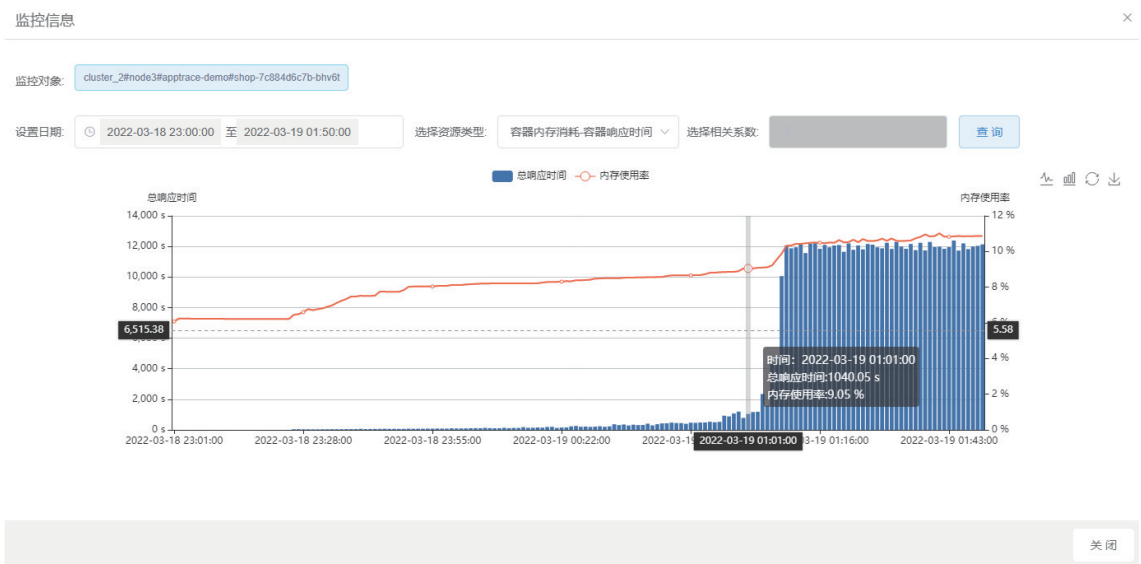

图 10 月光茶人系统线性增压访问下内存使用率与响应时间的指标展示

4.3 性能测试

性能测试从资源消耗、定时采集的耗时情况两个方面考察监控系统的可用性.

4.3.1 监控系统对集群资源的消耗程度

由于本监控系统仅访问 K8s 的 API 服务器用于采集数据, 因此只需要测试访问 API 服务器的资源消耗程度即可. 数据采集默认频率为每分钟一次, 也可以根据容器部署的策略以及容器实际生命周期的平均时间来设置采集频率, 最高精度为 1 s (若采集频率高于 K8s 资源指标 API 的刷新频率, 可能导致资源指标的重复采集).

测试度量为 CPU 使用率和内存使用率, 给出测试时间段内的最大值、最小值以及平均值. 首先测试平时 API 服务器的资源消耗程度, 测试时间为 30 min. 然后使用 JMeter 对 3 个集群的 API 服务器以每秒 10 次的访问频率进行资源消耗测试, 测试持续 30 min.

JMeter 对每个集群的访问频率是远高于监控系统定时采集监控指标的频率的, 而根据测试实验结果, 对 API 服务器的 CPU 消耗不超过 4%, 内存消耗不超过 2%, 可见监控系统通过 K8s 集群 API 服务器采集监控指标对集群的资源影响很小.

4.3.2 定时采集的耗时情况

考察用于存放监控数据的 8 数据表, 提取每张表 2 万余条连续采集的数据进行性能考察. 定义每次采集开始到第一条数据获取的时间为计算耗时、定义每次采集开始到最后一条数据入库为总耗时. 表 3 给出每张表的总数据量、每次采集的平均数据量 (平均数据量保留小数点后两位).

表 3 表数据量详情

资源类型	表类型	总数据量 (条)	单次采集平均数据量 (条)
CPU	metadata	22 302	207.50
	data	26 363	207.49
内存	metadata	27 416	207.45
	data	27 416	207.45
网络	metadata	22 972	265
	data	22 972	265
存储	metadata	22 394	300.81
	data	22 063	300.86

实验结果显示, 监控系统如果采用分钟级别的数据采集方式, 每张数据表能够完成万余条的数据写入, 相当于可以获取万余个集群组件的资源指标.

5 结论与展望

为了解决当前容器监控角度单一, 无法为运维提供全局统筹监控的问题, 本文提出面向 K8s 的容器立体化监控理念, 并实现了一个容器级别的立体化监控系统. 目前, 本文以一个特定的业务容器为例子验证了通过计算相关系数来实现立体化监控的方法的可行性, 但是由于容器往往提供单一服务, 不同的服务间消耗的资源各有不同, 因此相关性的结论在不同的容器中会存在不同, 需要根据实际容器提供的业务进行具体分析. 后续的研究工作可以通过这种立体化的方式以其他的指标为关注点进行分析, 为服务告警或异常预测提供技术上的支持.

参考文献

1 Anderson C. Docker [Software engineering]. IEEE Software, 2015, 32(3): 102–c3. [doi: 10.1109/MS.2015.62]

2 Kubernetes. https://kubernetes.io/. [2022-11-26].

3 Sysdig. Sysdig 2021 container security and usage report. https://dig.sysdig.com/c/pf-2021-container-security-and-usage-report?x=u_WFRi&utm_source=gated-organic&utm_medium=website. [2022-11-26].

4 Kubernetes-Retired. Heapster: Compute resource usage analysis and monitoring of container clusters. https://github.com/kubernetes-retired/heapster. [2022-11-26].

5 Google. Cadvisor: Analyzes resource usage and performance characteristics of running containers. https://github.com/google/cadvisor. [2022-11-26].

6 Prometheus Authors. Prometheus: monitoring system & time series database. https://prometheus.io/. [2022-11-26].

7 吴兆松. Zabbix 企业级分布式监控系统. 第 2 版, 北京: 电子工业出版社, 2019.

8 Nagios Enterprises, LLC. Nagios—The industry standard in IT infrastructure monitoring. https://www.nagios.org/. [2022-11-26].

9 Vitaly Agapov. Check_kubernetes: Nagios/Icinga/Zabbix style plugin for checking Kubernetes. https://github.com/agapoff/check_kubernetes. [2022-11-26].

10 Justin Miller. Kubernetes-Nagios: Basic health checks for a Kubernetes cluster. https://github.com/colebrooke/kubernetes-nagios. [2022-11-26].

11 张松, 疏官胜, 李京. 容器微云监控系统的设计和实现. 中国科学技术大学学报, 2017, 47(8): 627–634. [doi: 10.3969/j.issn.0253-2778.2017.08.001]

12 徐鑫辰. 边缘计算下的 Docker 容器安全监控系统 [硕士学位论文]. 成都: 电子科技大学, 2021.

13 Hong J, Kim W, Yoo JH, *et al*. Design and implementation of container-based M-CORD monitoring system. Proceedings of the 20th Asia-Pacific Network Operations and Management Symposium. Matsue: IEEE, 2019. 1–4.

14 Chang CC, Yang SR, Yeh EH, *et al*. A Kubernetes-based monitoring platform for dynamic cloud resource provisioning. Proceedings of 2017 IEEE Global Communications Conference. Singapore: IEEE, 2017. 1–6.

15 李轲. 面向 Kubernetes 的容器立体化监控研究与实践 [硕士学位论文]. 上海: 华东师范大学, 2022.

16 方奕, 柳亚磊. NetSensor 网络全流量分析解决方案. 2021 年国家网络安全宣传周“网络安全产业发展论坛”论文集. 西安: 《信息安全研究》杂志社, 2021. 126–130.

17 李轲, 窦亮, 杨静. 面向 Kubernetes 的多集群资源监控方案. 计算机系统应用, 2022, 31(7): 77–84. [doi: 10.15888/j.cnki.csa.008565]

18 Kubernetes API reference docs. https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.21/. [2022-11-26].

(校对责编: 牛欣悦)