

本文引用格式: 曾佳,王航,杨晨,等.面向分布式系统的无侵入式链路追踪方法研究[J].自动化与信息工程,2026,47(3):42-50.

ZENG Jia, WANG Hang, YANG Chen, et al. Research on non-intrusive link tracing method for distributed systems[J]. Automation & Information Engineering, 2026,47(3):42-50.

面向分布式系统的无侵入式链路追踪方法研究

曾佳 王航 杨晨 卢晓燕 张倍嘉 李豪

(中国民航信息网络股份有限公司重庆研发中心, 重庆 401122)

摘要: 微服务与云原生架构的普及使分布式系统日趋复杂, 可观测性面临严峻挑战, 而传统链路追踪方法存在的代码侵入性强、链路还原不完整及异构环境兼容性差等问题, 为此本文提出一种基于多源数据融合的无侵入式分布式链路追踪方法。该方法利用扩展伯克利包过滤器内核级探针捕获网络调用事件, 结合标准化应用日志进行上下文关联, 在无需修改应用代码的前提下实现端到端的链路数据采集。基于 OpenTracing 标准构建统一的数据模型, 采用包含关系型数据库、时序数据库、图数据库及全文检索数据库的多模态存储架构, 实现异构数据的高效管理。通过实时流式计算与离线批处理的协同工作, 自动识别违规调用、循环调用等异常模式。实验结果表明, 该方法在保证低资源开销的同时, 实现了调用链路的精准还原与可视化展示, 显著提升了故障定位效率与系统运维水平。

关键词: 链路追踪; eBPF; 依赖分析; 异常检测

中图分类号: TP393.07

文献标志码: A

文章编号: 1674-2605(2026)03-0006-009

DOI: 10.12475/aie.20260306

开放获取

Research on Non-intrusive Link Tracing Method for Distributed Systems

ZENG Jia WANG Hang YANG Chen LU Xiaoyan ZHANG Beijia LI Hao

(Chongqing R&D Center, TravelSky Technology Limited, Chongqing 401122, China)

Abstract: The proliferation of microservices and cloud-native architectures has led to increasingly complex distributed systems, posing significant challenges to observability. Traditional distributed tracing methods often suffer from strong code invasiveness, incomplete link reconstruction, and poor compatibility in heterogeneous environments. To address these issues, this paper proposes a non-invasive distributed tracing method based on multi-source data fusion. This method utilizes the extended Berkeley Packet Filter kernel-level probe to capture network call events and combines them with standardized application logs for context correlation, achieving end-to-end link data collection without modifying business code. Based on the OpenTracing standard, a unified data model is constructed, and a multi-modal storage architecture incorporating relational, time-series, graph, and full-text search databases is adopted to efficiently manage heterogeneous data. Through the collaboration of real-time stream processing and offline batch processing, abnormal patterns such as prohibited calls and cyclic calls are automatically identified. Experimental results show that this method achieves precise reconstruction and visualization of call links while maintaining low resource overhead, significantly improving fault localization efficiency and system operation and maintenance.

Keywords: link tracing; eBPF; dependency analysis; anomaly detection

0 引言

随着微服务与云原生架构的普及, 分布式系统呈现出服务繁多、调用复杂、部署动态性强等特点。同时, 跨节点交互频繁且技术栈高度异构, 导致系统行

为难以被感知, 传统观测手段无法实现请求全链路的精准还原与故障分析。链路追踪^[1]作为提升系统可观测性的关键方法^[2], 虽已有 Jaeger^[3]、Zipkin^[4]、SkyWalking^[5]等主流方案, 但这些方案普遍依赖于在

应用层代码中植入埋点^[6]，存在侵入性强、链路覆盖不全、跨系统还原能力弱及智能分析不足等问题，不仅难以适应复杂多变的生产环境，也制约了分布式系统的运维效率与稳定性。

近年来，为克服传统埋点方法的局限，链路追踪研究逐步转向无侵入式追踪方法。例如利用可编程数据平面实现带内网络遥测^[7]，能够实时获取数据包的转发路径信息，具有高精度与低开销的特点，但其应用受限于特定硬件支持，通用性不足。基于内核级探针技术的扩展伯克利包过滤器（extended Berkeley Packet Filter, eBPF）^[8]，无需修改应用代码与内核源码，即可安全高效地捕获系统调用、网络 I/O 及进程上下文等细粒度运行信息，从而实现对主机内部执行流与跨进程调用的完整追踪。结合机器学习方法，可从日志、指标及稀疏采样数据中推断调用拓扑与依赖关系，缓解链路断裂问题，提升追踪覆盖范围。然而，现有方法在多源数据融合能力、语义关联精度及跨层级建模方面仍存在不足，难以兼顾低侵入性、高覆盖率与深度分析的需求。

为此，本文提出一种多源数据融合的非侵入式链路追踪方法。该方法通过整合多类异构数据，构建跨层级、跨系统的统一追踪语义模型，在最小化侵入的前提下实现调用链路的高度还原与上下文关联；结合智能化分析引擎，自动识别调用关系，评估依赖合规性，定位异常根源，并动态呈现全景拓扑与性能状态，从而提升复杂环境下链路可观测的完整性与分析深度，增强故障定位^[9]效率与系统韧性，为云原生架构下的运维决策^[10]、风险预警与服务治理提供高精度、高实时性的数据支撑。

1 相关方法

链路追踪是一种用于观测分布式系统中跨服务、跨进程调用行为的核心方法，旨在实现对复杂微服务架构下请求的全链路追踪与性能分析。其基本原理是在请求发起时生成唯一的追踪标识^[11]，并在调用链路的每个环节通过上下文传播机制传递该追踪标识，使分散在不同服务中的操作记录能够被关联到同一逻辑请求中。

1.1 链路数据模型

OpenTracing^[12]是一套跨平台、跨语言的分布式追踪标准，通过统一语义与应用程序编程接口（application programming interface, API）推动追踪技术标准化，其核心模型由 Trace 与 Span 构成。其中，Trace 为一次完整请求的调用链路，由多个因果关联的 Span 组成；Span 为调用链路中的逻辑单元，包含操作名称、时间戳、标签、日志及父子引用关系，据此可构建调用拓扑与时序视图。为细化性能分析，Span 还支持事件注解，可标记关键节点，以精准度量网络延迟与服务耗时。OpenTracing 标准还提供上下文传播机制，确保 Trace 信息跨服务透传，支撑链路重建，并兼容多种采样策略，可兼顾监控精度与系统开销，从而适配高并发生产环境。请求调用链路图如图 1 所示。

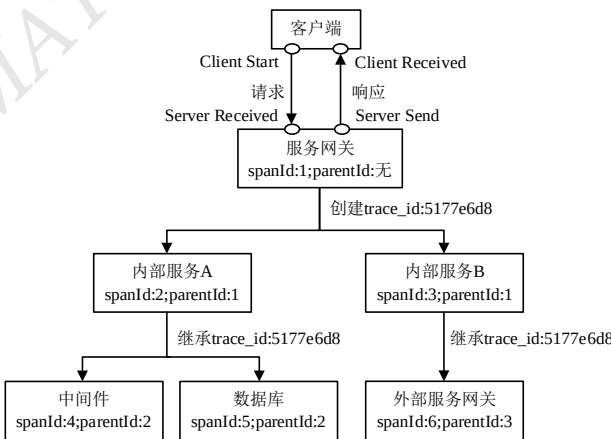


图 1 请求调用链路图

1.2 链路数据采集

依据对应用的影响程度，链路数据采集方法分为侵入式和非侵入式两类。

侵入式链路数据采集方法通过应用代码中插入探针或引入软件开发工具包的方式，直接获取方法调用栈、链路上下文信息与执行耗时等运行信息。该类方法能够提供高精度、结构化的追踪数据，语义完整且易于与现有监控系统集成，适用于对调用链路分析要求较高的场景。然而，其实现依赖于特定语言和

运行环境,需对应用代码进行修改,易引发版本耦合、升级困难等问题;同时,采样逻辑通常运行在业务线程中,可能带来额外的性能开销甚至影响服务稳定性,导致整体维护成本较高。

无侵入式链路数据采集方法则无需修改应用代码,主要通过外部观测手段实现链路还原,具有部署灵活、兼容性强等特点,适合多语言、异构化架构的生产环境。常见的无侵入式链路数据采集方式包括日志解析、网络流量监听和系统调用捕获等。其中,日志解析实现简单,但依赖于应用输出的规范日志,存在延迟、丢失和格式不一致等问题,且难以准确重建跨服务调用时序;网络流量监听可在传输层捕获真实通信行为,但对加密流量解析能力有限,且缺乏上下文关联信息;系统调用捕获(如 eBPF)通过监听内核事件实现细粒度行为追踪,具有低开销、跨语言、高覆盖等特点,但对操作系统版本有一定要求,且在复杂协议解析时需配合用户态组件协同处理。不同的无侵入式链路数据采集方式在数据完整性、资源消耗与适用范围之间相互制约,需根据实际环境选择合适方案。

1.3 链路数据处理

链路数据处理主要包括数据传输、数据清洗与数据计算等阶段,依托分层架构实现高效协同与弹性扩展。数据传输阶段通过消息队列实现异步缓冲、削峰与解耦,有效应对高并发流量冲击,确保数据可靠流转。数据清洗阶段采用可配置的流式管道,完成结构化解析、字段映射与上下文增强,提升异构数据的一致性与可关联性。数据计算阶段一般采用流式与批处理一体化架构^[13]。其中,实时流式处理引擎基于事件时间与状态管理,实时完成调用链路拼接、指标聚合与异常检测,满足低延迟监控与即时告警需求,适用于高吞吐、高频率的在线场景;离线批处理引擎则面向长期存储的历史数据,周期性执行深度分析任务,如跨服务依赖建模、调用模式挖掘与数据质量校验,用于补充实时流式处理中难以覆盖的长周期行为。两种引擎分工明确:实时流式处理引擎保障实时性,离线批处理引擎拓展分析深度,共同提升链路追踪在复杂分布式环境下的可观测性与分析完备性。

1.4 链路数据存储

在分布式系统可观测性体系中,链路数据因类型多样、访问模式各异,需依托多模态存储架构实现高效管理。不同数据类型对应不同的存储范式,如结构化数据宜采用支持强一致性与事务保障的关系型数据库,适用于元数据与配置管理;图结构数据^[14]需采用基于节点与边关联模型的图数据库,支持高效遍历与深度关系查询,适用于服务依赖与调用拓扑分析;时序数据^[15]需采用适配高吞吐写入的时序数据库,通过优化时间索引、压缩策略与自动过期机制,适配性能指标与监控流存储;日志与半结构化数据则依赖全文检索数据库^[16]的高效检索与灵活聚合能力,支持快速定位与多维分析。

2 系统设计与实现

为应对传统链路追踪方法存在的链路还原完整性不足、侵入性高、实时性弱及异构环境兼容性差等问题,本文提出了一种多源数据融合的非侵入式链路追踪方法,并依据该方法设计并实现了一套分布式链路追踪系统,作为该方法的具体运行载体。该系统包括数据采集、分析处理、数据存储和系统应用 4 个模块,其架构如图 2 所示。

数据采集模块通过部署基于 eBPF 的 Agent 探针、Filebeat 日志采集器及 Exporter 指标采集器,覆盖网络、系统与应用层数据,在不修改应用代码的前提下实现全栈感知。

分析处理模块采用实时与离线双链路驱动架构。Prometheus 采集监控指标后直接存入数据存储模块;日志与探针数据以 Kafka 为核心消息总线进行传输。实时数据可直接下发至系统应用模块,由 Flink 引擎完成实时聚合;离线数据经由 Logstash 处理后送入数据存储模块,再通过 Spark 引擎进行补充分析,二者协同兼顾数据处理的时效与深度。

数据存储模块按数据类型分离存储。其中,InfluxDB 存储时序指标;ElasticSearch 支持日志的全文检索;Oracle 存储结构化链路元数据;Neo4j 存储服务依赖拓扑图,各数据库各司其职,避免性能瓶颈。

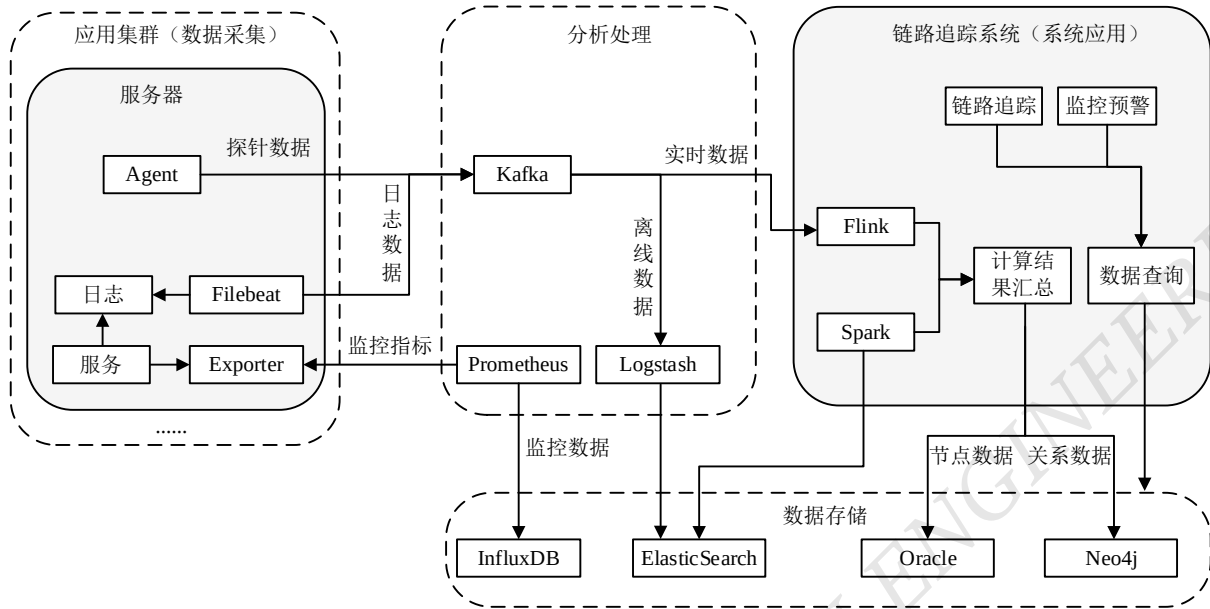


图2 分布式链路追踪系统架构图

系统应用模块提供链路追踪、监控预警、计算结果汇总与数据查询^[17]等核心功能。后端采用 Flink 和 Spark 双引擎支持实时与离线计算，前端支持调用链的可视化展示与交互查询，并通过布局优化提升用户体验。

分布式链路追踪系统通过分层解耦、多源数据融合与无侵入设计，在低干扰、高覆盖前提下，具备高吞吐、低延迟与强扩展能力，切实支持复杂系统的实时可观测与智能运维。

2.1 基于 eBPF 的 Agent 探针

为实现对网络链路调用信息的无侵入式采集，分布式链路追踪系统设计并实现了基于 eBPF 的内核级 Agent 探针^[18]。在 Linux 内核空间，TCP 协议栈事件触发 eBPF 程序，精准提取源/目的 IP、端口、进程 PID 等关键字段并写入 BPF Map。在用户空间，eBPF 加载器通过共享内存读取 BPF Map，触发数据处理器构建以五元组（源 IP、源端口、目的 IP、目的端口、传输层协议）为核心的网络连接事件流，并发送至 Kafka。该事件流与后续日志数据进行时空关联，从而实现调用链路的初步还原。其优势在于低侵入性、高覆盖率及接近零的性能开销，且无需修改应用代码。基于

eBPF 的 Agent 探针工作流程如图 3 所示。

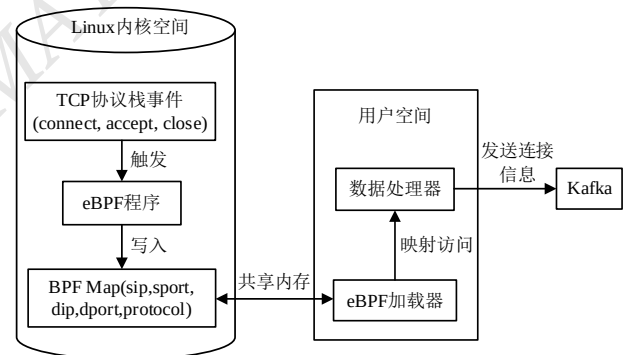


图3 基于 eBPF 的 Agent 探针工作流程图

由于 eBPF 对内核系统有最低要求，且需要 root 权限执行。因此，对于不支持 eBPF 的系统，本文设计了基于 Shell 脚本的备用采集方案。该采集方案通过定时轮询/proc/net/tcp、ss 或 netstat 命令输出，解析活跃连接状态，并结合 lsof 关联进程信息，实现粗粒度的调用关系捕获。其核心表达式如下：

$$C_{[t_0, t_1]} = \{(S_{ip}, S_{port}, D_{ip}, D_{port}, P_{protocol}) | S_{state} = E_{ESTABLISHED}\}$$

式中： $C_{[t_0, t_1]}$ 表示时间窗口 $[t_0, t_1]$ 内的活跃状态网络连接集合； S_{ip} , S_{port} , D_{ip} , D_{port} , $P_{protocol}$ 表示一条网络连接的元组，分别对应源 IP、源端口、目的 IP、目

的端口以及传输层协议； S_{state} 表示连接状态，当其取值为 ESTABLISHED 时，表示该连接已成功建立并处于活跃状态。

该采集方案实现简单、兼容性好，但存在采样延迟大、精度低、资源占用高等问题。

综上所述，eBPF 以其高精度、低开销的优势成为探针采集的首选方案；而基于 Shell 脚本的备用采集方案作为兼容性补充手段，保障了系统在异构环境下的部署可行性，二者共同构成数据采集模块的基础。

2.2 日志标准化与解析

为补全调用链路上下文语义，分布式链路追踪系统要求应用输出标准化的网络调用日志，并遵循统一的日志模型：

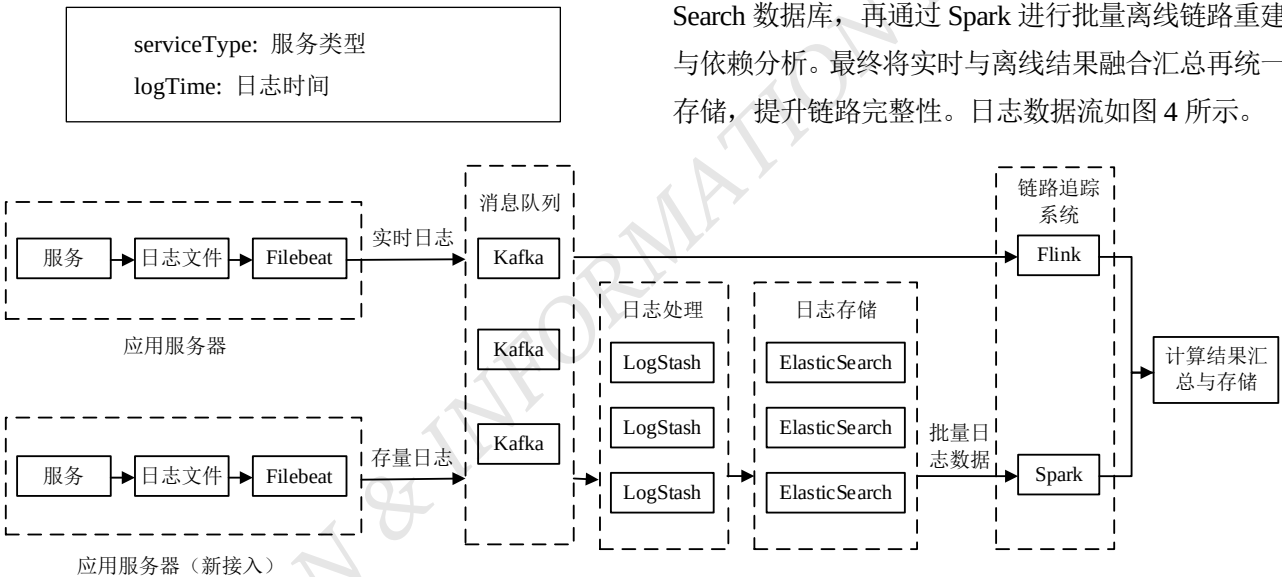
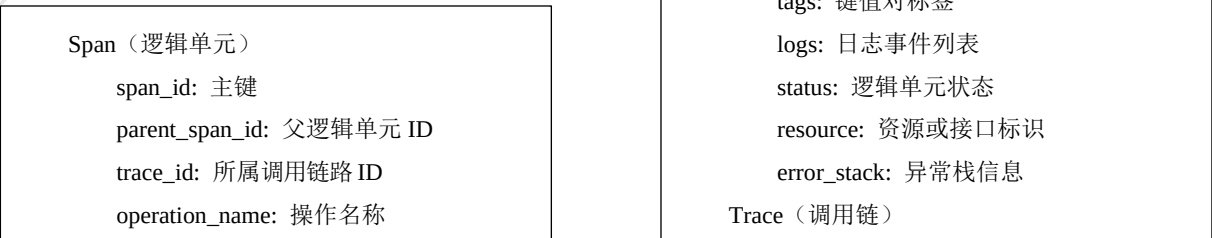


图 4 日志数据流图

2.3 链路建模与存储

为实现调用链路的统一语义表达，分布式链路追踪系统基于 OpenTracing 标准构建链路数据模型，以 Span、Trace 为核心的数据模型设计如下：



trace_id: 主键
 root_span_id: 根逻辑单元 ID
 start_time: 开始时间
 duration: 总持续时间
 service_count: 服务数量
 span_count: 逻辑单元总数
 status: 整体状态
 error_count: 错误逻辑单元数量
 tags: Trace 标签

在存储层面，系统采用多引擎协同架构，根据数据类型与应用场景精准匹配存储引擎，实现链路数据的高效组织与访问。其中，原始调用链记录写入 Oracle 关系型数据库，依托其强一致性与事务保障能力，确保核心链路数据完整性，支持溯源与查询；性能指标如调用耗时、每秒查询数（queries per second, QPS）、错误率等高频时序数据存入 InfluxDB 时序数据库，利用其高压缩比、高写入吞吐和高效时间窗口查询优势，优化指标写入与聚合效率，以满足实时监控与趋势分析的需求；服务间调用依赖关系以拓扑结构存入 Neo4j 图数据库，将服务实例作为节点、调用路径作为边，提升关联分析能力，支持动态拓扑建模；关键日志写入 Elastic-Search 全文检索数据库，构建可索引的日志仓库，实现海量链路数据的快速全文检索与聚合分析。四类引擎协同工作，在保证数据一致性的同时提升查询响应速度与存储空间利用率。尤其在大规模调用场景下，该架构增强了依赖分析、根因定位与智能预警的执行效率，避免一库多用导致的性能瓶颈，为分布式链路追踪系统提供稳定、可扩展的数据支撑。

3 实验与分析

3.1 实验环境配置

为全面评估本文提出的分布式链路追踪系统的有效性 & 性能，本文搭建了基于 Kubernetes 1.30 的实验环境。该环境共包含 40 台物理服务器，每台服务器均配置海光 C86 7390 CPU、32 GB 内存，操作系统为麒麟 V10。实验设计了链路追踪验证、异常依赖分析、系统性能测试、异常定位与预警四组场景，以一

个最多包含 200 个微服务节点的测试平台（数据已脱敏）为实验对象，并选取当前业界广泛使用的开源链路追踪平台 SkyWalking 作为对照组，进行对比分析。

3.2 实验结果分析

3.2.1 链路追踪验证

链路追踪验证实验场景通过模拟用户登录、查询、下单的完整流程，验证分布式链路追踪系统对完整调用链路的追踪能力、trace_id 的跨服务传递以及可视化渲染与节点聚合能力，实验结果如图 5、6 所示。

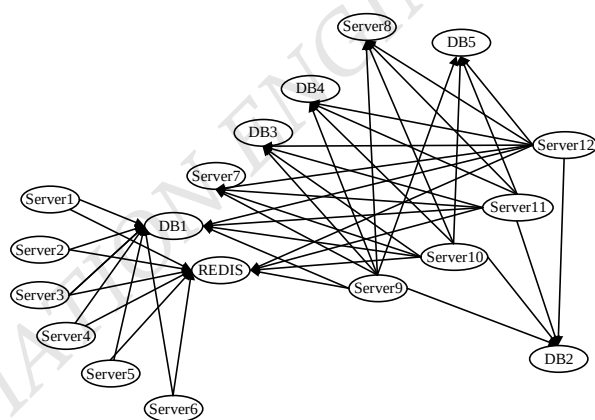


图 5 调用链路追踪效果示意图（20 个服务，50 个节点）

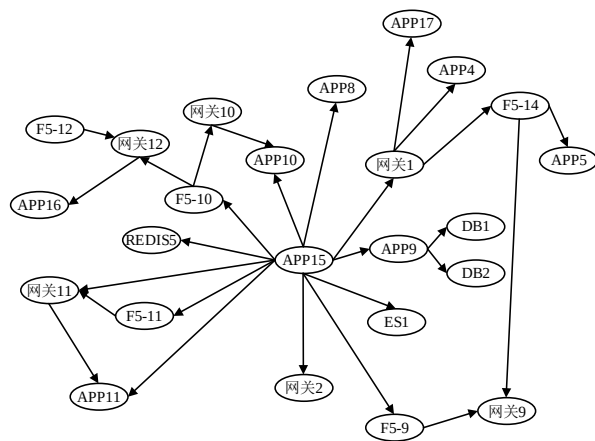


图 6 调用链路追踪效果示意图（60 个服务，200 个节点）

由图 5、6 可知：调用链路追踪图能够直观呈现服务间的调用关系与调用路径，有助于用户快速理解系统整体拓扑结构。此外，当某服务部署了多个实例节点时，系统会自动聚合这些节点的调用信息，以单

一服务节点形式呈现在图中，从而有效降低视图复杂度，提升用户的浏览与分析体验。上述结果表明，系统在调用链可视化与节点聚合方面达到了预期设计目标，验证了链路追踪功能在实际环境中的可行性与有效性。

3.2.2 异常依赖分析

在大规模集群中，服务按保障等级划分，高等级服务应避免依赖低等级服务，以防止可靠性降级，此类调用视为违规调用；同时，应杜绝服务间循环调用及无必要却实际发生的无效调用。为验证本系统对异常依赖类型的识别能力，人工注入循环调用、无效调用和违规调用三类异常依赖类型，并比对注入记录与本系统、SkyWalking 的检测结果，评估本系统异常检测的覆盖率与准确性。实验结果如图 7 所示。

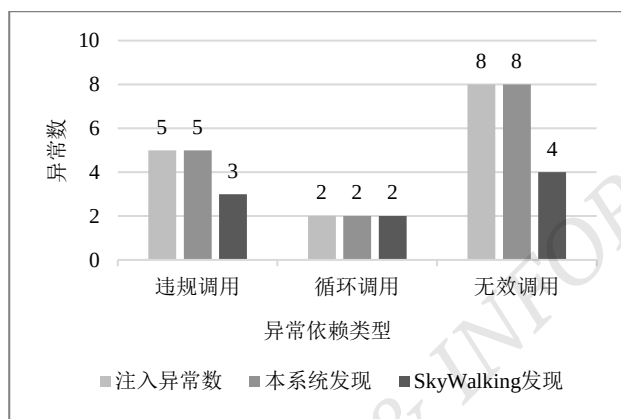


图 7 异常依赖检测结果对比图

由图 7 可知：本系统可有效识别违规调用、循环调用与无效调用，三类异常的检测覆盖率均达到 100%。相比之下，SkyWalking 在违规调用和无效调用上分别存在 2 个和 4 个漏检，覆盖率分别为 60% 和 50%。分析其原因，SkyWalking 具有一定侵入性，需在应用中植入 Agent，部分服务因技术栈不兼容或安全策略限制无法安装，导致链路数据采集不完整，从而产生漏检。

3.2.3 系统性能测试

系统性能测试实验场景采用逐步增加 QPS 的方式，评估本系统的性能开销及其对微服务平台性能的影响。

测试要求本系统的数据延迟时间 ≤ 100 ms，接口平均响应时间 ≤ 50 ms，并以服务及 Agent 探针平均 CPU 使用率、平均内存占用率为评估指标，实验结果如表 1、2 所示。

表 1 服务及 Agent 探针资源占用实验结果 %

QPS	平均 CPU 使用率		平均内存占用率	
	本系统	对照组	本系统	对照组
100	5.0	5.6	4.8	4.9
200	12.2	13.1	4.9	5.0
500	20.4	22.7	5.3	5.5
1 000	40.9	43.6	6.4	6.7
2 000	83.3	89.5	7.9	8.6

表 2 接口响应、数据延迟时间实验结果 单位：ms

QPS	接口平均响应时间		数据平均延迟时间	
	本系统	对照组	本系统	对照组
100	12	13	21	11
200	14	17	21	11
500	17	22	21	12
1 000	25	36	21	14
2 000	40	58	21	18

由表 1 和表 2 可知：本系统占用资源较低。即便在 QPS 高达 2000 的情况下，平均 CPU 使用率、平均内存占用率及接口平均响应时间均低于对照组。本系统的数据平均延迟时间恒为 21 ms，主要由于数据推送链路中数据采集及批处理间隔所致；虽然略长于对照组，但系统延迟稳定性更高（标准差小于 1 ms）。以上指标完全满足设计要求，验证了系统的高性能特性。

3.2.4 异常定位与预警

为验证本系统对业务集群服务故障的发现能力，本实验通过在多个服务中引入延迟与异常响应，并通过控制异常数量，验证本系统在少量异常（<10 个/s）与大量异常（>100 个/s）场景下的故障处理能力。实验结果如表 3 所示。

表 3 异常定位与预警延迟时间

单位: s

异常数量	异常类型	发现故障延迟时间		定位故障延迟时间		发出预警延迟时间	
		本系统	对照组	本系统	对照组	本系统	对照组
少量异常 (<10 个/s)	慢调用	≈1.0	≈10	≈1.1	—	≈1.1	≈10.1
	调用异常	≈1.0	≈10	≈1.1	—	≈1.1	≈10.1
大量异常 (>100 个/s)	慢调用	≈1.5	≈10	≈1.6	—	≈1.6	≈10.1
	调用异常	≈1.5	≈10	≈1.6	—	≈1.6	≈10.1

注：“—”表示不支持定位故障功能；“≈”表示近似值；预警延迟时间为 5 次实验的平均值。

由表 3 可知：本系统依托多源数据融合方法可在 2s 内发现故障、定位故障、发出预警；对照组只能依赖单一数据来源，且无法自动定位故障。

4 结论

面对微服务与云原生架构下分布式系统可观测性不足的挑战，本文提出了一种多源数据融合的无侵入式链路追踪方法。该方法构建了覆盖数据采集、分析处理、数据存储和系统应用的链路追踪体系；通过 eBPF 探针与标准化日志的协同采集，实现无侵入式网络调用信息捕获，有效解决传统埋点方案覆盖不全与链路断层问题；基于 OpenTracing 标准构建统一的 Span 与 Trace 数据模型，并采用关系型数据库、时序数据库、图数据库及全文检索数据库的多模态存储架构，分别支撑链路数据管理、服务依赖分析与性能指标监控，提升了系统的可扩展性与查询效率。通过对链路追踪数据进行前端可视化渲染与节点聚合，实现了直观、友好且易使用的界面，有效提升了运维效率。实验结果表明，依据该方法实现的系统在功能完整性、分析准确性与运行效率方面均满足设计目标，适用于复杂生产环境下的链路追踪与故障诊断需求。该方法有效提升了分布式环境下各类应用服务的可观测能力，为后续智能化运维提供了高质量的数据基础与技术支撑，具有良好的实用价值与推广前景。

©The author(s) 2024. This is an open access article under the CC BY-NC-ND 4.0 License (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

参考文献

- [1] 杨勇,李影,吴中海.分布式追踪技术综述[J].软件学报,2020,31(7):2019-2039.
- [2] 陈恒法,黄爱仪,邹竟辉.一种可观测性微服务链路监控系统设计实现[J].电脑知识与技术,2024,20(22):38-41;44.
- [3] 张爱华,白金峰.基于 OpenTelemetry+Jaeger 的分布式系统调用链路监控方案[J].软件工程,2023,26(9):48-51.
- [4] 杨帆.基于 zipkin 协议的分布式调用跟踪方案[J].福建电脑,2018,34(1):124;152.
- [5] 吴晟.Apache SkyWalking, 开源监控生态[J].软件和集成电路,2021(6):62.
- [6] SIGELMAN B H, BARROSO L A, BURROWS M, et al. Dapper, a large-scale distributed systems tracing infrastructure [R]. Google Technical Report (dapper-2010-1), 2010. Google Technical Report
- [7] 吕鸿润,李清,沈耿彪,等.带内网络遥测方法综述[J].软件学报,2023,34(8):3870-3890.
- [8] 刘畅.基于 eBPF 的容器网络可观测性方法与实践[D].杭州:浙江大学,2021.
- [9] 刘锦鸣.分布式链路异常日志采集方法研究[J].机电信息,2022(23):36-38.
- [10] HOLLINGSWORTH J K, MILLER B P. Dynamic control of performance monitoring on large scale parallel systems[C]// Proceedings of the 7th International Conference on Supercomputing, 1993:185-194.
- [11] 黄梓程,陈鹏飞,余广坝,等.面向 Java 微服务系统的透明请求追踪及采样方法[J].软件学报,2023,34(7):3167-3187.
- [12] 曾颖青,唐晖,许丽娟.基于 OpenTracing 分布式日志跟踪设计与实现[J].软件,2023,44(3):26-31.
- [13] 金伟,李明桂,张若愚,等.面向多源异构大数据的引接处理技术[J].通信技术,2025,58(8):852-859.

- [14] 刘岍,李杨,段宏,等.知识图谱构建技术综述[J].计算机研究与发展,2016,53(3):582-600.
- [15] 林志达,张华兵,张今革.基于时序数据库的监控数据存储方法研究[J].电子元件与信息技术,2020,4(1):73-74;100.
- [16] 李永春,丁华福.Lucene 的全文检索的研究与应用[J].计算机技术与发展,2010,20(2):12-15.
- [17] 佟业新,曲新奎,杨皓然,等.基于微服务分布式链路的服务质量优化策略[J].计算机系统应用,2024,33(9):140-152.
- [18] YANG W, CHEN P, LIU K, et al. Nahida: In-band distributed tracing with eBPF[J]. arXiv preprint arXiv:2311.09032, 2023.

作者简介:

曾佳,男,1993年生,学士,工程师,主要研究方向:软件工程、网络安全、大数据分析。E-mail: jzeng@travelsky.com.cn

王航,男,1982年生,硕士研究生,工程师,主要研究方向:软件工程、DevOPS、人工智能。E-mail: hangwang@travelsky.com.cn

杨晨,男,1991年生,学士,工程师,主要研究方向:软件工程、认证安全、云计算。E-mail: yangc@travelsky.com.cn

卢晓燕,女,1985年生,硕士研究生,高级工程师,主要研究方向:软件工程、人工智能、质量管理。E-mail: xylu@travelsky.com.cn

张倍嘉,女,1990年生,学士,助理工程师,主要研究方向:质量管理、技术管理。E-mail: zyj@travelsky.com.cn

李豪,男,1994年生,学士,助理工程师,主要研究方向:软件工程、人工智能。E-mail: li_hao@travelsky.com.cn

2027 年“磷化工过程具身智能”专题征稿启事

磷化工过程与具身智能的深度融合是磷化工产业绿色低碳转型、实现高质量发展的前沿重点研究方向,旨在将具身智能的感知、认知、决策与行为执行能力融入磷化工全流程,破解行业高污染、高资源消耗及高危场景作业等痛点难点,实现磷矿开采、磷酸制备、磷石膏处理、高端产品合成等环节的智能化升级,全面提升生产效率、资源利用率与安全生产水平,具有重要的科学研究价值与广阔的产业应用前景。随着大模型技术与工业智能的创新突破,磷化工过程与具身智能的融合发展面临新的机遇与挑战。本刊组织“磷化工过程具身智能”专题,聚焦于探索多模态感知融合的磷化工具身智能大模型,研究基于行业机理与数据驱动的认知决策方法,构建磷化工关键环节灵巧操作、智能调控与无人值守的具身智能模型,支撑磷化工产业从“资源依赖”向“技术驱动”转型,推动其向绿色化、智能化、高端化发展。特邀本刊编委黄自鑫教授、刘皓教授、丁芝侠教授、程莉教授、李璇教授作为特邀主编,策划并主持该专题,现真诚邀请广大专家、学者积极撰文,踊跃投稿。

一、专刊征稿范围(包括但不限于)

- 1、磷化工复杂场景具身智能建模、仿真与优化调控
- 2、基于大模型的磷化工过程具身智能集成系统
- 3、磷化工过程多模态具身感知与融合理论
- 4、磷化工过程人机协作与多具身智能装备协同
- 5、磷化工过程物理数字孪生系统建模与仿真
- 6、磷化工过程数据建模与具身智能预警技术
- 7、磷化工工艺过程具身智能柔顺调控技术
- 8、基于磷化工机理的知识化认知与决策模型
- 9、基于具身智能的磷化工厂区安全监测
- 10、磷化工厂区具身智能巡检装备自主导航与路径规划
- 11、磷化工厂区具身智能巡检装备安全控制
- 12、磷化工具身智能装备操作技能学习与优化
- 13、磷化工高危场景具身机器人灵巧操作技术
- 14、磷化工物料处理具身智能装备精准操作
- 15、磷化工过程隐私与数据安全

二、投稿要求: 请参考本刊官方网站 <http://www.giim.ac.cn/scipub/postdoc/> 或扫描右边本刊微信公众号的二维码获取“投稿须知”。

三、投稿截止时间: 2026 年 12 月 20 日

四、出版时间: 2027 年第 1 期(2027 年 2 月 28 日)

五、投稿方式

1、登录 <http://gzxg.cbpt.cnki.net/WKE/WebPublication/index.aspx?mid=gzxg> 网上投稿或 aie@giim.ac.cn 邮箱投稿。

2、投稿时请务必注明是“磷化工过程具身智能”专题投稿。

六、编辑部联系方式 邮箱: aie@giim.ac.cn; 电话: (020)87687999

