用于微服务根因定位的智能体记忆增强递归推理

Lingzhe Zhang 北京大学 北京，中国
zhang.lingzhe@stu.pku.edu.cn

Tong Jia*
北京大学
北京，中国
jia.tong@pku.edu.cn

Yunpeng Zhai
阿里巴巴集团
中国
zhaiyunpeng.zyp@alibaba-inc.com

Leyi Pan 清华大学 中国
p
anly24
@mails.tsinghua.edu.cn

Chiming Duan
北京大学
北京，中国
duanchiming@stu.pku.edu.cn

Minghua He
北京大学
北京，中国
hemh2120@stu.pku.edu.cn

Mengxi Jia 人工智能研究院 中国
jiamx1@chinatelecom.cn

Ying Li*北京大学
北京，中国
li.ying
@pku.edu.cn

arXiv:2601.02732v1 [cs.SE] 6 Jan 2026

摘要

随着当代微服务系统日益普及且愈发复杂——通常由数百甚至数千个细粒度、相互依赖的子系统组成——其故障发生得越来越频繁。因此，确保系统可靠性需要准确的根因定位。尽管人们已针对这一任务探索了许多传统的基于图的方法和深度学习方法，但这些方法往往严重依赖预定义的模式，难以适应不断变化的运行环境。因此，近期已有多种基于大语言模型的方法被提出。然而，这些方法仍面临两项主要局限：以症状为中心的浅层推理削弱了准确性，以及缺乏跨告警复用，导致重复推理和较高的延迟。在本文中，我们对站点可靠性工程师（SREs）如何定位故障根因进行了全面研究，并从多个组织的专业人员那里获得了相关洞见。研究发现，专家的根因分析呈现出三个关键特征：递归性、多维扩展和跨模态推理。受这些发现启发，我们提出了 AMER-RCL，即一种用于微服务根因定位的智能体记忆增强递归推理框架。AMER-RCL 采用递归推理根因定位引擎，这是一种多智能体框架，可针对每条告警执行递归推理，逐步细化候选原因；同时，智能体记忆会在时间窗口内逐步积累并复用先前告警中的推理结果，从而减少重复探索并降低推理延迟。实验结果表明，AMER-RCL 在定位准确率和推理效率两方面均始终优于最先进的方法。

CCS 概念

• **软件及其工程 → 软件维护**。

关键词

根因定位、智能体记忆、递归推理

ACM 引用格式：

LingzheZhang，Tong Jia*，Yunpeng Zhai，LeyiPan，Chiming Duan，Minghua He，Mengxi Jia 和 Ying Li*。2026年。用于微服务根因定位的智能体记忆增强递归推理。载于 *2026 IEEE/ACM 第48届国际软件工程大会（ICSE-SEIP '26），2026年4月12日至18日，巴西里约热内卢*。ACM，纽约州纽约，美国，12页。https://doi.org/10.1145/3786583.3786853

1 引言

现代微服务由于动态交互和不断演进的运行时环境而变得日益复杂 [65, 80]。这些系统通常由数千个细粒度、相互依赖的子系统组成，其中任何一个组件出现问题，都可能轻易导致顶层性能问题 [16, 27, 49, 64, 71, 74]。因此，为确保系统可靠性，及时定位这些问题的根因至关重要 [63, 70]。

由于子系统之间存在错综复杂的依赖关系，在微服务系统中定位根因本质上极具挑战性 [7, 13, 14, 42, 43, 47, 51, 57, 68, 75, 82]。每个请求通常都会经过复杂的调用链，涉及服务、实例、主机，以及服务调用和数据库查询等多种交互。这些交互具有动态性，加之系统异构，使根因定位变得极为困难。为应对这一问题，先前的研究主要采用基于图和基于深度学习的方法。基于图的方法通过因果图或依赖图捕捉服务间依赖关系，并常结合统计或机器学习技术加以增强 [17, 20, 21, 55, 56, 59, 79]。这些方法能够提供可解释的推理路径，但依赖僵化的图结构，因而难以适应异构平台和不断演变的环境。相比之下，基于深度学习的方法会自动从

*通讯作者。

本作品采用 CreativeCommons Attribution4.0 InternationalLicense 许可。
ICSE-SEIP '26，巴西里约热内卢 © 2026年 版权归所有者/作者所有。ACM
ISBN 979-8-4007-2426-8/2026/04
https://doi.org/10.1145/3786583.3786853

指标和追踪数据中学习时间、空间和结构依赖关系 [2, 3, 6, 9, 35, 45, 53]。然而，这些方法往往像黑箱一样运行，导致其预测难以解释，也限制了所学模型在不同部署环境间的迁移能力。总体而言，尽管这两种范式都颇为有效，但仍存在一些共同局限，包括缺乏跨平台通用性，以及对不断变化的系统状态适应不足。

幸运的是，大语言模型（LLM）展现出了解决这些局限的巨大潜力，越来越多的研究也开始探索其在根因分析中的应用 [12, 18, 32, 34, 36, 38, 40, 46, 48, 52, 66, 69, 72, 73, 76, 78]。

现有研究大致可分为三个方向：（i）早期探索，强调大语言模型用于根因分析的可行性与潜力 [12, 36, 37, 40, 52, 73]；（ii）系统化解决方案，采用多智能体架构来组织推理过程 [32, 34, 46, 48, 69, 72, 76]；以及（iii）知识增强方法，整合检索增强生成（RAG）等技术，使大语言模型能够利用领域知识 [18, 37, 38, 66, 78]。尽管这些方向前景可观，但在真实世界的微服务系统中实际部署基于大语言模型的根因定位仍面临若干关键挑战。

• **浅层、以症状为中心的推理会限制准确性**。根因定位不仅需要识别异常症状，还需要在复杂的服务依赖关系中区分真正的原因与下游影响。现有基于大语言模型的方法通常采用通用推理框架，例如 ReAct 或思维链（CoT） [18, 32, 34, 46, 48, 76]。虽然有些方法会纳入针对特定任务的启发式规则，但其推理过程往往仍停留在浅层，并以症状为中心，无法系统地区分主要异常与次要异常。因此，这些方法常常混淆症状与真正的原因，在复杂或级联故障场景中导致准确性不佳。

• **缺乏跨告警复用会导致冗余推理和高延迟**。微服务系统中的许多故障具有系统性或随机性（例如资源争用、间歇性网络问题或级联故障），只有分析多个告警中的聚合模式才能可靠地检测到这些故障 [17, 20, 21, 55, 56, 59, 79]。现有基于大语言模型的方法通常独立分析每个告警，并针对每个告警启动完整的多智能体推理 [18, 32, 34, 46, 48, 76]。例如，mABC [76] 会针对每个告警分别执行多智能体推理。在包含数百甚至数千个告警的大型异常窗口中，许多追踪数据、指标或日志条目彼此重叠，但这些方法既不会复用先前的分析，也不会共享中间推理结果。这会导致大量冗余计算、较高的资源消耗以及更长的推理延迟。

认识到这一差距后，我们首先开展了一项实证研究，以探究站点可靠性工程师如何定位根因，并深入了解他们如何进行深度分析来应对第一个挑战。研究揭示了人工根因分析的三个关键特征：**(i) 递归性**，即发现更深层的根因后，会促使站点可靠性工程师通过检查较低层的表现形式来迭代完善分析；**(ii) 多维扩展**，即在垂直方向（跨系统层）和水平方向（跨相关组件与服务）拓宽搜索范围；以及 **(iii) 跨模态推理**，即通过相关指标和日志的波动，进一步验证追踪数据所提示的候选根因追踪数据所提示的内容，会通过相关指标和日志的波动进一步验证。

基于这些见解，我们提出了**AMER-RCL**，一种面向微服务根因定位的**A**gentic**M**emory（智能体记忆）**E**nhanced（增强）**R**ecursive（递归）推理框架**R**oot **C**ause **L**ocalization。为应对第一个挑战，AMER-RCL 采用**递归推理 RCL** 引擎。这一多智能体框架针对每条告警执行递归推理，生成按优先级排序的候选根因集合。为应对第二个挑战，AMER-RCL 会处理微服务系统在一个时间窗口内产生的多条告警。每条告警首先通过因果图提取过程转换为图表示，以捕获异常请求路径及其上下文依赖。利用这些图，**智能体记忆**逐步记录先前的推理结果和中间思考，使系统能够跳过高度相似告警的冗余分析，同时将递归探索集中于显著差异，从而提高准确性和效率。

我们在 AIOPS 2022、Train-Ticket 和 FAMOS-Mall 数据集上开展实验，以评估 AMER-RCL。结果表明，在准确性方面，AMER-RCL 优于其他基于大语言模型和非基于大语言模型的根因定位方法，领先第二名方法约 16%。在批量告警分析的推理效率方面，AMER-RCL 相较于现有基于大语言模型的方法实现了超过 3.5× 倍的加速。总而言之，本文的主要贡献如下：

• 我们全面研究了站点可靠性工程师如何定位根因。研究发现，人工根因分析呈现出三个关键特征：递归性、多维扩展和跨模态推理。

• 受这些发现启发，我们提出了 AMER-RCL，这是一种用于微服务根因定位的智能体记忆增强型递归推理框架。

• 我们在三个数据集上评估了 AMER-RCL，以验证其有效性。实验结果表明，AMER-RCL 在根因定位准确率和推理效率方面均优于最先进的方法。

2 背景

本节介绍本文的必要背景，包括根因定位问题的形式化定义、传统根因定位方法概述，以及作为根因定位主要数据源的分布式追踪简介。

2.1 根因定位

分布式系统中的故障诊断通常分为两类：故障类别分类和根因定位。前者用于确定故障类型，例如中央处理器或内存异常；后者则精确找出导致问题的具体节点、服务或 Pod。根因定位对于最大限度缩短停机时间、确保系统稳定性尤为关键。

传统的根因定位方法通常会分析预定义时间窗口内的一组告警。这些方法构建依赖图，其中节点表示系统组件，边则表示从追踪数据或系统日志中推断出的因果关系。随后会应用基于图的算法（例如 PageRank），

根据组件最有可能成为根因的可能性对其进行排序。

$$C^* = C^* = \arg\max_{C \in C} s(C, G, M) \quad (1)$$

形式化地，给定一组异常请求 $R = \{r_1, r_2, \ldots\ldots r_n\}$，构建组件图 $G = (C, E)$，其中 C 是组件集合，E 表示它们之间的依赖关系。根因组件 C^* 按照公式 1 确定，其中 $s(C, G, M)$ 表示计算得出的分数，该分数综合考虑了图拓扑和观测到的异常。

2.2 分布式追踪

为了支持软件系统中的细粒度故障诊断，分布式追踪已在工业环境中得到广泛采用，成为现代软件基础设施不可或缺的一部分 [39, 44, 54]。分布式追踪会记录请求在系统中传播时的详细执行过程，捕获时间、依赖关系和性能特征。每条轨迹由多个跨度组成，这些跨度记录了请求执行路径上的各项操作。

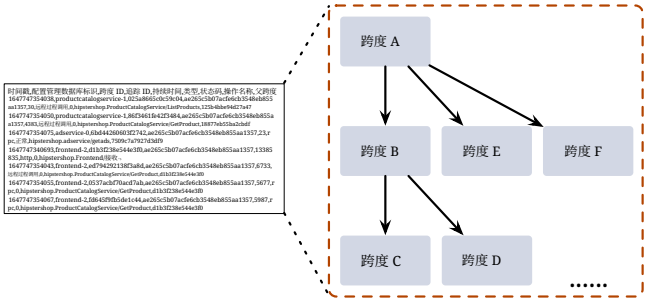

图 1：轨迹示例

如图 1 所示，一条完整的轨迹展现了请求端到端的历程，详细记录了每项中间操作及其对应的延迟。每个跨度都会记录关键信息，例如服务名称、操作、时间戳以及操作之间的因果关系。通过分析跨度的时间信息和依赖关系，分布式追踪能够在大规模分布式系统中实现精确的性能监控，并促进异常检测。这种结构化表示对于诊断延迟问题、识别服务瓶颈以及揭示微服务中的故障传播模式尤为有价值。

3 实证研究

在本节中，我们全面研究站点可靠性工程师如何定位根因。我们设计的方法结合了专家驱动建模和从业者验证。

首先，来自三个组织——北京大学、清华大学和阿里巴巴集团——的各一名研究人员共同合作，分析具有代表性的案例，并对根因定位过程进行了初步刻画。

其次，为了完善并验证这一框架，我们对共计 22 名专业开发人员开展了半结构化访谈——
参与者包括来自北京大学、清华大学、阿里巴巴集团和中国电信人工智能研究院的运维人员和站点可靠性工程师。在访谈中，参与者受邀审阅初始框架，结合自身运维经验进行反思，并提出修正或补充意见。随后，我们将收集到的反馈汇总，形成框架的最终版本，以确保其准确性和广泛适用性。

通过这一两阶段流程，我们得到了经验证的手动根因定位特征描述。为了系统地分析这一流程，我们围绕以下研究问题组织研究发现：

- **RQ1:** 站点可靠性工程师最初如何缩小根因的搜索空间?
- **RQ2:**他们如何系统地扩大搜索范围，以涵盖所有潜在根因?
- **RQ3:** 他们如何将实际根因与其他可能存在但并非因果因素的异常区分开来?

3.1 递归性

为了缩小根因的搜索空间，站点可靠性工程师从有问题的请求入手，检查其追踪数据，以识别被调用的服务和已执行的操作。如果在这些服务中检测到异常，他们会进一步调查这些服务所调用的下游服务，递归地缩小搜索范围，直到找出真正的根因。

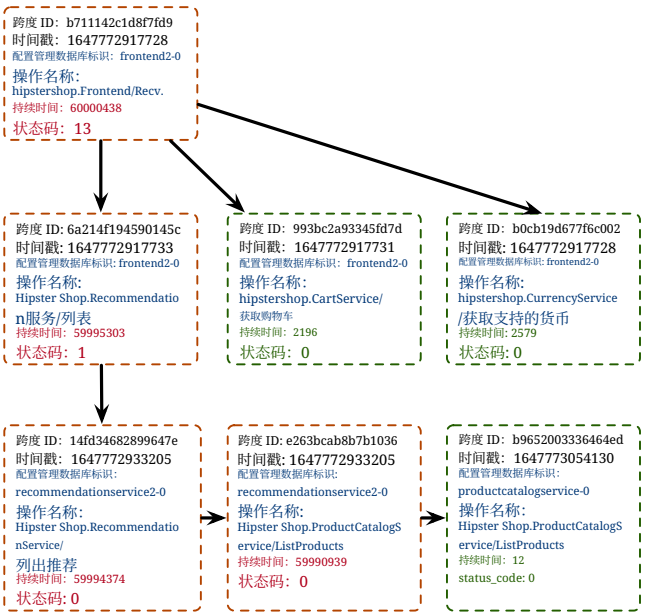

图2：调用链图示例-1

例如，我们从真实世界的 AIOPS 2022 数据集中提取了一个典型案例，如图2所示。在该案例中，一条 cmdb_id 为 frontend2-0 的请求遭遇了严重超时，最终导致错误（status_code = 13）。作为站点可靠性工程师，我们首先分析了该请求的追踪数据，并识别出三个主要操作：CartService/GetCart、RecommendationService/List，以及

CurrencyService/GetSupportedCurrencies。值得注意的是，只有 RecommendationService 操作出现了超时，这表明它很可能是触发告警的原因。

随后，我们对下游服务进行了更深入的调查，发现带有 cmdb_id recommendationservice2-0 的 RecommendationService 操作也发生了超时。进一步分析表明，在相同 cmdb_id 下，后续的下游操作 ProductCatalogService 也发生了超时。然而，组件 cmdb_id ' productcatalogservice-0' 上之后执行的操作并未出现任何问题。

这一递归调查过程有效地将潜在根因缩小至组件 cmdb_id frontend2-0 和 recommendationservice2-0，显著减少了用于识别底层性能问题的搜索空间。

总结。 通过分析追踪数据，站点可靠性工程师从告警追踪的入口跨度开始，递归检查下游操作，直到识别出异常组件，从而缩小根因的搜索空间。

3.2 多维扩展

前面的分析通过聚焦于表示单个 Pod 的 cmdb_id，缩小了搜索空间。然而，仅识别异常的 cmdb_id 并不足以定位真正的根因，因为故障往往会在多个组件之间传播。在实践中，站点可靠性工程师会沿着两个相辅相成的维度扩展调查范围：(i) **垂直扩展**，基于现有证据深入探索系统层次结构（服务 → 服务实例→ Pod → 主机）的更深层；以及 (ii) **水平扩展**，关联同一异常窗口内的告警，以发现跨服务或多追踪依赖。

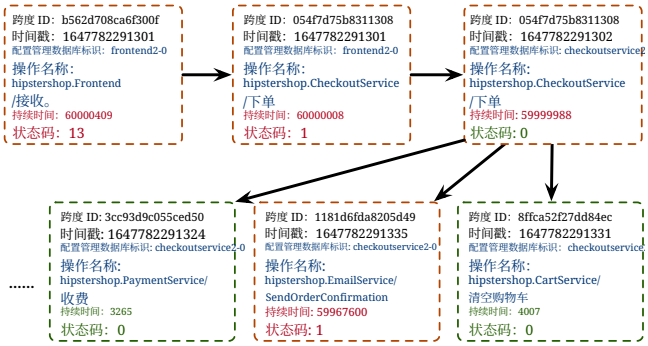

图 3：追踪图示例-2

对于垂直扩展，operation_name 通过将告警关联到相应的服务，提供了更细粒度的信息。例如，根据上述递归分析，RecommendationService 和 ProductCatalogService 都是潜在的根因。图 3 展示了一个更明显的案例：异常显然与 cmdb_id checkoutservice2-0 下的 EmailService 操作有关，这表明有必要进一步调查 EmailService。除了服务层之外，站点可靠性工程师还

检查底层物理基础设施。例如，从指标数据中可以实时观察到，recommendationservice2-0 和 checkoutservice2-0 均部署在节点-5 上。

这种关联表明，必须评估节点-5 本身的健康状况，以确定它是否导致了观察到的异常。

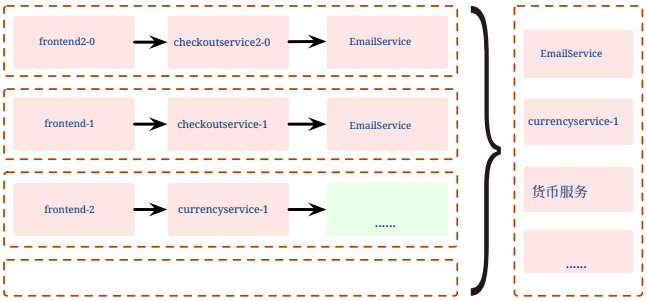

图4：多轨迹图示例

对于水平扩展，同一异常窗口内出现多个告警可提供更丰富的上下文证据。如图4所示，多个服务最终通过不同的 Pod 指向 EmailService，而其他告警则指出 currencyservice-1 中存在错误。这些相互重叠但又部分分歧的信号要求对轨迹和告警进行全面关联，使站点可靠性工程师能够区分系统性问题与孤立故障，并据此优化候选根因集合。

总结。 站点可靠性工程师沿垂直和水平方向扩展根因分析：垂直方向上，从低层实例追踪异常至更高层服务及其底层基础设施；水平方向上，则关联同一异常窗口内的告警和追踪数据，从而更全面地发现潜在故障源。

3.3 跨模态推理

通过分析递归性和多维扩展，站点可靠性工程师已从 Pod、服务和节点三个维度识别出潜在根因。然而，确切的根因仍不确定。在这一阶段，站点可靠性工程师通常会结合其他模态的数据，其中最常用的是指标数据，以确定最终根因。

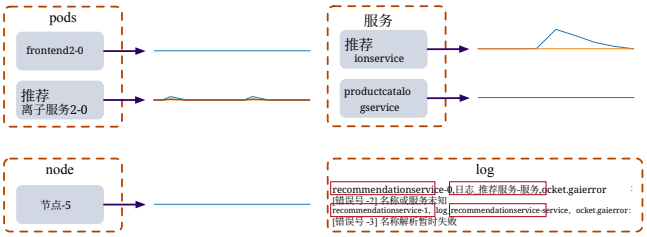

图5：调用链图示例-1的指标与日志检查

例如，在图2中，可能存在问题的 Pod 是 frontend2-0 和 recommendationservice2-0，可能存在问题的服务是 recommendationservice 和 productcatalogservice，可能存在问题的节点是节点-5。对指标数据进行分析后发现，如图5所示，frontend2-0、productcatalogservice,

以及节点-5均未出现显著波动。相比之下，recommendationservice2-0 和 recommendationservice 均出现明显变化，其中 recommendationservice 的波动更为显著。recommendationservice2-0 中的异常是由 recommendationservice 传播而来的。此外，日志显示，recommendationservice-0 和 recommendationservice-1 均将 recommendationservice 标记为异常，从而间接排除了特定单个 Pod 存在问题的可能性。最终，根因被确定为 recommendationservice，这与真实情况一致——Kubernetes 容器中发生了网络数据包损坏。

总结。站点可靠性工程师会分析其他数据模态，尤其是指标数据，以区分实际根因与其他潜在异常，并将波动最显著或最早出现的组件识别为真正的根因。

4 AMER-RCL

我们的实证研究展示了人类站点可靠性工程师如何系统地定位根因。基于这些发现，我们提出了**AMER-RCL**，即**智**能体**记忆增强递**归推理，用于微服务**根**因**定**位。图 6 展示了 AMER-RCL 的整体架构。

AMER-RCL 以微服务系统在一个时间窗口内的多个告警作为输入。每个告警首先通过因果图提取 流程转换为图表示，该流程会捕获异常请求路径及其上下文依赖关系。这些图为评估告警之间的相似性和差异提供了基础。借助智能体记忆，系统会逐步记录先前的推理结果以及递归推理 *RCL* 流程中的中间思考，形成持久的对话历史。这种双重作用使框架能够跳过高度相似案例的冗余分析，并在出现显著差异时引导深入探索。因此，结构化的图到对话处理流程与递归推理 *RCL*组件紧密耦合；该组件针对相关告警开展多智能体推理，并最终生成按优先级排序的候选根因列表。

递归推理 *RCL*模块由元智能体驱动，它采用递归推理 策略来协调专门的智能体，例如*TraceAgent*、*LogAgent*和*MetricAgent*，并配合整合器 整合中间发现。为确保这一递归过程兼具深度和可靠性，元智能体执行深度保证推理框架：通过初始推理 生成初步假设，通过批判性反思审查并完善这些假设，再通过最终审查 综合所有证据，列出按优先级排序的根因。

4.1 智能体记忆

智能体记忆被设计为一个结构化、可查询的存储库，将每条告警的因果图（作为键）与其对应的推理轨迹（作为值）关联起来。从概念上讲，它与递归推理 RCL 引擎相互独立：因果图通过确定性数据处理方法从原始告警中提取，而存储的推理轨迹则在运行时用于决定：当引擎接收到新告警时，是复用已有轨迹、从出现分歧处继续推理，还是重新启动分析。

4.1.1 因果图提取。对于每个输入告警，我们都会确定性地构建因果图，如公式 2 所示，其中 V 是节点集合，E 是有向边集合，$\mathbf{A}_v$ 是节点属性空间，$\mathbf{A}_e$ 是边属性空间。

$$G = (\mathcal{V}, \mathcal{E}, \mathbf{A}_v, \mathbf{A}_e) \tag{2}$$

每个节点 $v \in \mathrm{V}$ 都与一个属性向量 a_v =（服务_id, Pod_id,操作_名称, 指标_总结, 日志_总结）相关联，其中，指标_总结$_v$ = $[\mu_m, \sigma_m]$ 表示告警窗口内各项指标的均值 μ_m 和标准差 σ_m，日志_总结$_v$ = [计数$_{l_1}$、……、计数$_{l_k}$] 表示同一窗口内观测到的l_1、……、l_k类型的日志条目数量。

边（u，v） $\in$ E 表示直接调用或因果依赖（例如服务间调用），并带有属性 a_e =（调用_延迟，状态_码）；其中，调用_延迟是从 u到 v的观测调用时长，状态_码则表示其响应状态。

4.1.2 记忆积累与利用。每次执行递归推理 RCL 后，相应的因果图及关联推理步骤都会作为一条记忆条目存储在智能体记忆中，之后可在后续的 RCL 执行中加以利用。

记忆积累。如公式 3 所示，每条记录都是一个元组，其中 f_i 是 $G^{(i)}$ 的紧凑指纹，通过计算规范化图哈希来唯一标识图拓扑；e_i 是 $G^{(i)}$ 的稠密嵌入，通过 Graph2Vec [28] 编码器生成，用于高效的最近邻检索；$G^{(i)}$ 是该告警对应的完整因果图；meta_i 包含时间戳和告警 ID 等元数据；τ_i 是结构化推理记录，由一系列推理步骤组成，每个步骤都明确关联到 $G^{(i)}$ 中的节点。

$$M_i = (f_i, e_i, G^{(i)}, \mathrm{meta}_i, \tau_i) \tag{3}$$

记忆利用。给定一个新的告警图 G^{new}，系统首先对 $\{e_i\}$ 执行基于嵌入的最近邻搜索，然后按照公式 4 计算精确相似度，从记忆中检索相关条目。其中，S_{struct} 衡量图之间的结构相似度（例如，通过嵌入余弦相似度或 Weisfeiler–Lehman 核），S_{attr} 衡量节点属性的一致性（例如，基于节点级指标和日志计算归一化逆距离）。

$$S(G^{\mathrm{new}}, G^{(i)}) = \alpha \cdot S_{\mathrm{struct}}(G^{\mathrm{new}}, G^{(i)}) + (1-\alpha) \cdot S_{\mathrm{attr}}(G^{\mathrm{new}}, G^{(i)}) \tag{4}$$

根据相似度阈值，会出现三种情况：

(1) 高相似度：如果 $S \geq \tau_{\mathrm{skip}}$，则可以直接复用存储的推理 τ_i。(2) 部分相似度：如果 $S \in [\tau_{\mathrm{partial}}, \tau_{\mathrm{skip}})$，分歧节点

$$\mathcal{D} = \{v \in G^{\mathrm{new}} \mid \mathrm{dist}(a_v^{\mathrm{new}}, a_v^{(i)}) > \delta\} \tag{5}$$

表示应继续推理的点。这里，dist(·,·) 是归一化的节点属性距离（例如马氏距离或 z 分数偏差），用于捕捉指标、日志或其他诊断信号之间的差异。

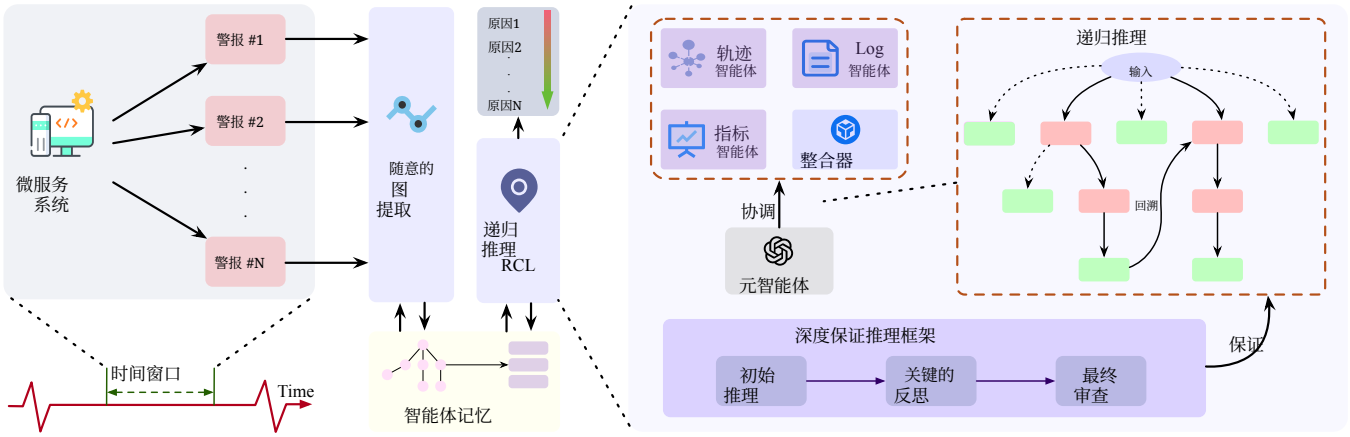

图 6：AMER-RCL 的架构

(3) 相似度低：如果 $S < \tau_{\text{partial}}$，则从头开始对 G^{new}进行推理。

4.2 专用代理

专用代理负责根据其指定角色，对不同类型的输入数据和中间信息执行有针对性的分析。每个专用代理都专注于其指定角色，处理追踪数据、日志、指标或中间信息，生成有助于整体推理过程的洞见。在本文中，我们主要使用四种代理：链路追踪智能体、日志代理、指标智能体和整合器。链路追踪智能体、日志代理和指标智能体分别分析各自的数据源并生成结构化观测结果，而整合器则将这些发现整合为连贯的总结，供元智能体在递归推理中使用。

4.2.1 链路追踪智能体。 追踪数据是根因定位最基础的数据源，因为它记录了各个服务之间的调用顺序。然而，由于每个请求都会生成一条完整的调用路径，追踪数据的规模可能非常庞大，使大语言模型难以处理如此广泛的上下文。为解决这一问题，链路追踪智能体被设计用于筛选和检索相关的追踪数据子集。

$$T(s) = \{\langle t, s', svc, op, d, \sigma\rangle \mid s' \in C(s)\} \tag{6}$$

形式化地，给定一个跨度标识符 s，链路追踪智能体会返回一组子跨度及其关联的元数据。我们将此函数表示为公式 6，其中 t表示时间戳， s' 表示子跨度标识符， svc 表示服务名称， op 表示操作名称， d 表示持续时间， σ 表示状态码，而 C(s) 是给定跨度 s的所有子跨度的集合。

4.2.2 日志代理。日志数据提供了系统事件和组件行为的文本证据，这些证据对于诊断异常至关重要。然而，日志量通常极其庞大，只有其中一部分与特定事件相关。为此，日志代理会有选择地检索指定时间窗口内、目标组件中最有可能表明存在异常的日志。

具体而言，给定输入时间戳 t_0 和目标键组件 C（例如，Pod 或服务），令 L(C) 表示与 C 及其相关实体关联的所有日志的集合。日志代理会返回落在时间区间 $[t_0 - \delta, t_0 + \delta]$ 内且满足相关性标准的日志子集，如公式 7 所示。

$$L(t_0, \delta, C) = \{l \in \mathcal{L}(C) \mid t(l) \in [t_0 - \delta, t_0 + \delta] \wedge \phi(l) = 1\} \tag{7}$$

这里， $t(l)$表示日志条目 l的时间戳，而 $\phi(l)$是一个二元函数，用于指示 l 是否可能与异常分析相关。在实践中， $\phi(l)$ 的确定会考虑消息类型、错误代码，以及与其他告警或追踪事件的关联等因素。通过以这种方式筛选日志，日志代理能够高效缩小上下文规模，同时保留对递归推理过程最具信息价值的条目。

4.2.3 指标智能体。指标数据表示系统组件的运行时状态。在成熟的微服务环境中，有成千上万种不同的指标持续记录数值，其数据量往往超过追踪数据。此外，我们的实证研究表明，在系统异常期间，大多数指标保持稳定，没有显著波动。因此，我们开发了一种专用的指标智能体，用于有选择地检索预定义时间窗口内出现明显偏差的指标，以供大语言模型分析。

$$|m(t) - \mu_m| > n \times \sigma_m \tag{8}$$

形式化地，给定输入时间戳 t_0 和目标键组件 C（例如，Pod 或服务），令 M(C) 表示与 C 及其相关组件（包括与 Pod 对应的节点）关联的指标集合。对于每个指标 $m \in$ M(C)，令 μ_m和 σ_m 分别表示其历史均值和标准差。指标智能体会检查时间区间$[t_0 - \delta, t_0 + \delta]$ 内的指标值，并应用 n-西格玛检验：如果该区间内存在符合公式 8 定义的时间 t，则检测到异常。在这种情况下，智能体会返回长度为 δ的时间段内对应的波动数据，如公式 9 所示。

$$Q(t_0, \delta, C) = \left\{ m(t) \;\middle|\; \begin{array}{c} m \in \mathcal{M}(C),\ t \in [t_0 - \delta, t_0 + \delta], \\ |m(t) - \mu_m| > n \times \sigma_m \end{array} \right\} \quad (9)$$

4.2.4 整合器。该智能体作为一个集成组件，在递归推理过程中系统地汇总专用代理产生的中间推断。它不进行新的分析，而是将多步观察综合为统一的结构化表示，呈现对每个假设根因的关键证据支持。形式化地，对于每个候选根因，整合器会生成包含两个字段的标准化输出：root_cause，表示已识别的异常来源；以及 reason，概述根据先前的智能体推理步骤汇总得出的理由。

4.3 元智能体

元智能体在递归推理策略下协调专用代理，执行一种深度有保证的推理框架，从而保证该过程的深度和可靠性。

4.3.1 递归推理。递归推理源于我们对专业站点可靠性工程师如何定位根因所进行的实证研究。与按线性方式推进的传统思维链推理不同，递归推理会根据先前的结果，迭代生成逐步指令。每条指令都会调用下级智能体，以检索后续分析所需的数据。如果某条分析路径已穷尽，却仍未找到合理的根因，流程便会系统地回溯至先前已识别但尚未充分探索的候选项，从而确保对整个推理空间进行全面而彻底的搜索。

算法1递归推理算法

要求： 入口跨度 s_0
确保： 潜在根因的排序集合 R

```
函数RecursiveRCL(s)
I ←MetaAgent.GenerateInstruction(s)
D_t ← TraceAgent(I) {trace as 主要探索来源}
如果 MetaAgent.Suspect(s, D_t) 那么
    D_l ←LogAgent(I)
    D_m ←MetricAgent(I)
    如果MetaAgent.确认(s, D_l, D_m) 则
        返回{s} {confirmed as root cause}
    结束 if
结束 if
R ← ∅
对于每个子节点 c ∈元智能体.可疑子节点(s, D_t) 执行
    R ← R ∪RecursiveRCL(c)
结束循环
返回 R
结束函数

R ←RecursiveRCL(s_0)
返回 整合器(R)
```

在该框架中，递归推理主要由追踪数据驱动，同时利用日志和指标数据进行跨模态验证。如算法1所述，分析从告警轨迹的入口跨度开始。在每一步中，元智能体都会为当前候选跨度生成推理指令，而*TraceAgent*则检索相应的轨迹信息，作为评估根因可能性的主要依据。

被标记为可疑的候选项将由*LogAgent* 和 *MetricAgent*进一步检查，它们提供上下文和系统层面的证据，以 corroborate 异常。只有在这些模态中均得到确认的跨度，才会被加入潜在根因集合。对于未经确认的跨度，元智能体 会递归地识别并评估可疑的子跨度。该过程会持续进行，直至不再存在可行候选项；同时通过回溯确保全面覆盖尚未探索的路径。随后，整合器将所有中间结果综合为结构化、排序后的根因列表。

4.3.2 深度保证推理框架。 尽管递归推理算法（算法1）为根因定位提供了有原则的机制，但我们观察到，大语言模型往往会过早终止递归，从而忽略更深层的异常。为解决这一问题，我们引入深度保证推理框架 ，在执行 RecursiveRCL 期间显式强制进行足够深度的探索。该框架由三个互补阶段组成——初始推理、批判性反思和最终审查——每个阶段都会调节递归推理过程，以确保其深度与可靠性。

初始推理。 在 RecursiveRCL 开始时，元智能体根据入口跨度 s_0 生成推理指令，而 TraceAgent 则对轨迹树进行广泛探索。为加快候选跨度的早期扩展，此阶段有意不启用指标代理和日志代理。因此，初始递归会生成一条初步推理轨迹，如公式 10 所示，其中 A 表示专用代理集合。此步骤为后续细化提供了粗粒度的搜索前沿。

$$\Gamma_0 = \texttt{RecursiveRCL}\left(s_0 \,\middle|\, A \setminus \{A_{\text{metric}}, A_{\text{log}}\}\right) \quad (10)$$

批判性反思。 随着 RecursiveRCL 的推进，模型可能会在找到一个看似合理的候选项后停止，从而导致浅层推理。为克服这一倾向，关键反思阶段会强制对 Γ_0 中所有仍有可疑之处的跨度继续进行递归。在此阶段，会引入指标代理和日志代理，以确认或排除异常，从而深入递归探索。具体而言，细化后的推理轨迹由公式 11 得出，确保对轨迹树的更深层进行系统检查。

$$\Gamma_1 = \bigcup_{s \in \Gamma_0} \texttt{RecursiveRCL}(s \mid A) \quad (11)$$

最终审查。即使经过递归细化，大语言模型仍可能表现出近因偏差，过度强调递归后期遇到的跨度。为缓解这一问题，最终审查阶段会整合 Γ_0 和 Γ_1中所有递归调用的输出。该阶段仅调用整合器，将中间结果综合为最终排序后的根因集合，如公式 12 所示。

$$R_f = A_{\text{consolidator}}(\Gamma_0 \cup \Gamma_1) \tag{12}$$

通过审查完整的递归推理轨迹，该框架保证不会过早丢弃任何候选根因。

5 评估

为评估 AMER-RCL，我们开展了一系列实验研究，以探讨以下研究问题：

- **评估-RQ1：** 与基线方法相比，AMER-RCL 在根因定位方面的准确度如何?
- **评估-RQ2：** 与基于大语言模型的方法相比，AMER-RCL 在推理延迟方面的效率如何?
- **EV-RQ3：** 不同大语言模型骨干的选择如何影响 AMER-RCL 的准确率?
- **EV-RQ4：** AMER-RCL 的各个组件分别对其整体准确率有何贡献?

5.1 实验设置

5.1.1 数据集。我们在三个具有代表性的数据集上评估所提出的方法：AIOPS 2022 配套 [1] 数据集、TrainTicket 数据集 [80, 81],和 FAMOS-Mall [8]数据集。

AIOPS 2022 是一个从成熟且正在运行的微服务电商平台收集的大规模真实世界数据集。该数据集包含丰富的运维数据和多样的异常案例，因此被广泛用作异常检测与诊断的基准。

TrainTicket 是一个开源微服务基准系统，用于模拟典型的火车票预订服务，并已广泛应用于以往的故障诊断研究。我们在由六台虚拟机组成的 Kubernetes 集群上部署 TrainTicket，每台虚拟机均配备 8 核 Intel Xeon 2.60 GHz 中央处理器和 24 GB RAM，并运行 CentOS 8。该部署总共包含 90 个服务实例。为评估系统的鲁棒性，我们使用 ChaosMesh 注入故障，涵盖十种不同的故障类型。

FAMOS-Mall 是一个电子商务后端微服务系统，提供用户管理、购物车和商品管理等核心服务。与合成基准不同，FAMOS-Mall 部署在真实的生产环境中，并作为阿里云应用实时监控服务（ARMS）的演示系统，因此能够反映真实的运行状况。

5.1.2 基线方法。我们主要将 AMER-RCL 与三种具有代表性的基于大语言模型的根因定位方法进行比较。mABC [76] 采用受区块链启发的多智能体协作框架，其中多个基于大语言模型的智能体分别分析每条告警，并通过投票机制确定最终根因。RCAgent [48] 采用基于 ReAct 的推理与行动范式，联合处理多条告警：控制器智能体执行思考–行动–观察循环，调用各种工具对告警进行整体分析。基于思维链的方法 [50] 采用朴素的思维链推理策略，逐条处理告警并依次生成结论。

为了进行更全面的比较，我们还纳入了几种非基于大语言模型的方法。(1) **基于追踪的方法：** CRISP [79] 将追踪数据表示为关键路径，并采用轻量级启发式方法识别根因。TraceContrast [59] 采用序列表示、对比序列模式挖掘和频谱分析，定位多维根因。Trac-eRCA [19]使用频繁项集挖掘和频谱分析定位根因实例。MicroRank [55] 构建追踪覆盖树，以捕捉请求与服务实例之间的依赖关系，并利用 PageRank 算法对潜在根因进行排序。(2) **基于指标的方法：** RUN [20] 采用时间序列预测进行神经格兰杰因果发现，并结合个性化PageRank算法推荐排名靠前的k 根因。Microscope [21] 构建因果图，并采用深度优先搜索策略检测前端服务中的异常。

5.1.3 评估指标与设置。我们采用 top-k 召回率（Recall@k）和平均倒数排名（MRR）评估根因定位准确率，遵循先前的研究 [21, 55, 59]。除非另有说明，我们使用 Claude-3.5 Sonnet 作为大语言模型骨干，将指标智能体中的n-sigma 阈值设为 $n = 3$，并将异常窗口大小设为 60 秒。

5.2 准确率

我们在三个基准数据集上，将 AMER-RCL 与非基于大语言模型的方法和基于大语言模型的方法进行比较。如表1所示，AMER-RCL 的表现始终优于所有最先进基线方法。在 AIOPS 2022 数据集上，AMER-RCL 的 MRR 达到 76.30%，比表现最强的基线 mABC 高 18.25%，比表现最佳的非基于大语言模型模型 TraceRCA 高 34.11%。在 Train-Ticket数据集上，AMER-RCL 的 MRR 达到 81.93%，比 mABC 高 14.67%，比 TraceConstract 高 27.58%。在 FAMOS-Mall 数据集上，AMER-RCL 的 MRR 达到 75.39%，比 mABC 高 14.34%，比 TraceConstract 高 16.68%。总体而言，在所有数据集上的平均表现中，AMER-RCL 比排名第二的方法（mABC）提升了 15.75%。

值得注意的是，mABC 在 AIOPS 2022 和 Train-Ticket 数据集上的表现优于 RCAgent，而 RCAgent 在 FAMOS-Mall 上表现更好。我们认为，这种差异源于告警特征以及各方法摄取证据的方式不同。FAMOS-Mall 的告警相对同质，因此 RCAgent 的批量联合分析能够利用不同告警之间早期且一致的相互印证，并得出有把握的结论。相比之下，AIOPS 和 Train-Ticket 中的告警异质性更高；将大量差异明显的告警一并输入时，批量输入可能超出实际可处理的上下文范围和注意力限制，导致模型优先考虑部分证据，并过早停止深入探查。mABC 通过独立分析告警并以投票方式汇总结果的策略，能够更稳健地应对这种异质性，但计算上存在冗余，而且局部分析意见不一致时可能产生嘈杂或不一致的投票结果。AMER-RCL 通过以下方式避免这些缺陷：(i) 对高度相似的告警复用先前的推理，以避免重复工作；(ii) 在出现分歧的节点启动有针对性且深度有保障的递归分析，充分探查异质情况。这种混合策略解释了 AMER-RCL 为何能在同质和异质数据集上均保持优异表现。

表1：与最先进方法相比的根因定位准确率（Rk 表示 Recall@k）。

范式	方法	AIOPS 2022					Train-Ticket					FAMOS-Mall				
		R1	*R3*	*R5*	*R10*	平均倒数排名	*R1*	*R3*	*R5*	*R10*	平均倒数排名	*R1*	*R3*	*R5*	*R10*	平均倒数排名
非基于大语言模型的	CRISP	3.92	29.79	48.88	51.01	18.68	31.45	41.77	48.19	55.26	38.53	24.81	44.48	50.83	60.52	35.34
	TraceConstract	24.06	36.51	37.69	40.81	30.29	48.37	55.33	63.78	78.21	54.35	45.37	72.83	82.84	89.34	58.71
	TraceRCA	36.43	43.13	44.56	51.88	42.19	42.73	46.98	47.17	49.32	45.34	30.13	51.08	57.96	69.18	41.17
	MicroRank	8.45	24.57	37.24	42.89	18.42	24.15	42.10	50.16	62.33	34.73	22.99	43.94	50.05	59.72	33.69
	RUN	4.69	11.23	11.88	25.81	10.44	40.03	47.19	51.31	56.77	48.38	34.33	56.07	64.21	75.34	46.70
	MicroScope	9.23	28.81	33.37	51.25	20.39	32.37	40.18	43.26	48.70	36.88	26.49	47.84	56.30	65.08	37.91
基于大语言模型	CoT	8.96	15.89	34.72	48.33	17.56	22.12	39.75	48.66	61.19	32.14	31.16	50.94	58.13	66.16	42.33
	RCAgent	13.15	23.11	38.56	54.87	22.53	38.51	53.14	60.47	69.98	47.39	46.86	75.17	83.10	91.18	61.05
	mABC	51.68	60.33	65.33	78.46	58.05	57.53	74.67	83.15	90.02	67.26	45.39	74.11	82.53	89.93	59.96
AMER-RCL（我们的方法）		**68.67**	**81.50**	**90.36**	**96.71**	**76.30**	**72.97**	**89.15**	**93.47**	**96.52**	**81.93**	**66.55**	**84.38**	**90.06**	**95.58**	**75.39**

5.3 效率

我们进一步将 AMER-RCL 与具有代表性的基于大语言模型的方法进行比较，评估其推理效率。如表2所示，AMER-RCL 显著提升了速度：平均而言，其速度比 RCAgent 快 3.53×，比 mABC 快 31.05×，比 CoT 快 14.61×，充分展现了其实用效率。

表2：推理速度比较（秒/查询）

方法	AIOPS 2022	Train-Ticket	FAMOS-Mall
CoT	1517.34	543.58	1401.51
RCAgent	322.16	305.84	378.23
mABC	2731.89	1490.15	2890.12
AMER-RCL（我们的方法）	**111.45**	**53.86**	**56.59**

RCAgent 的速度相对较快，因为它会在一个批次中处理多个告警，避免重复初始化；而 mABC 和思维链则逐一分析告警，其中思维链由于推理过程较浅，速度略快。值得注意的是，AMER-RCL 利用智能体记忆复用先前的推理结果，从而跳过冗余计算，性能优于 RCAgent。在 FAMOS-Mall 数据集上，这种加速尤为明显，因为告警大多具有同质性：AMER-RCL 只需分析具有代表性的告警即可收敛，而 RCAgent 尽管采用批处理，仍会对相似告警进行一些冗余推理。这些结果凸显了 AMER-RCL 将递归式深度推理与高效的跨告警复用相结合的能力。

5.4 大语言模型骨干影响

接下来，我们探讨 EV-RQ3。我们使用 AIOPS 2022 数据集——该数据集根据记录的云床和日期进一步划分为六个子集，分别标记为 **A**、**B**、 **Γ**、 **Δ**、**E** 和 **Z**——评估 AMER-RCL 在不同大语言模型骨干上的性能。除 Claude-3.5-Sonnet 外，我们还使用四种最先进的大语言模型进行实验：DeepSeek-R1-Qwen，这是一款基于 DeepSeek 发布的蒸馏数据微调的 Qwen2.5-32B 模型；Qwen-2.5-Max 和 Qwen-2.5-Plus，这两款闭源模型均通过应用程序接口调用；以及 Meta 发布的原始开源模型 Llama-3.1-70B。如表 3 所示，AMER-RCL 的性能与底层大语言模型骨干的推理能力密切相关。

表 3：采用不同大语言模型的 AMER-RCL（平均倒数排名 ）

模型	A	B	Γ	Δ	E	Z
Claude-3.5-sonnet	**73.89**	**81.23**	**92.87**	**66.81**	**68.05**	**74.92**
DeepSeek-R1-qwen	35.21	60.98	70.12	71.63	47.21	63.78
Qwen-2.5-max	12.21	20.12	32.14	12.21	32.14	12.71
Qwen-2.5-plus	41.83	72.87	54.91	51.62	37.58	54.19
Llama-3.1-70B	6.81	21.32	16.23	16.91	5.86	7.65

在我们的实验中，Claude-3.5-Sonnet 始终取得最佳结果，其次是 DeepSeek-R1-Qwen。不过，由于模型规模相对较小（32B），DeepSeek-R1-Qwen 在数据集 A 和 B 上的表现不如 Qwen-2.5-Plus。总体而言，使用 Claude-3.5-Sonnet 的 AMER-RCL 比排名第二的模型平均高出 18.14%。这些发现表明，AMER-RCL 能够有效利用大语言模型的语义理解和逻辑推理能力，并在与更强大的推理引擎配合时取得更好的结果。

5.5 消融研究

为回答 EV-RQ4，我们记录了 AMER-RCL 的完整推理路径，并评估其关键组件的贡献：深度保证推理框架（由初始推理（IR）、关键反思（CR）和最终审查（FR）组成），以及智能体记忆（Mem）。

表4：消融实验（平均倒数排名）

方法	A	B	Γ	Δ	E	Z
AMER-RCL	**73.89**	**81.23**	**92.87**	**66.81**	**68.05**	**74.92**
不使用记忆	71.13	78.57	90.24	64.34	65.42	72.41
不使用最终审查	70.45	72.65	84.15	64.13	61.23	71.77
不使用关键反思和最终审查	35.93	71.75	68.14	61.31	41.41	66.58

如表4所示，仅采用初始推理时，六个子集的平均倒数排名为57.52%，作为基线。加入批判性反思后，定位效果显著提升，例如子集A 的平均倒数排名从35.93%提高到70.45%，子集E 的平均倒数排名从41.41%提高到61.23%，展现了强制进行深度推理的有效性。最终审查进一步提升了性能，带来了额外收益，例如在

子集B 上达到 → 90.24%，以及在子集 Γ上达到84.15%。最后，智能体记忆通过利用此前的推理结果，带来持续的提升，例如子集A 上达到 → 73.89%，子集Z上达到 → 74.92%，凸显了其在减少冗余推理、整合多个告警中的知识方面的作用。

总体而言，纳入批判性反思使性能平均提升了 13.21%，最终审查进一步提升了 2.96%，而智能体记忆又带来了 2.61% 的提升。这些发现凸显了 AMER-RCL 设计的有效性与可靠性。

5.6 有效性威胁

尽管 AMER-RCL 展现出令人鼓舞的结果，但仍应承认其存在若干局限。首先，关于基线方法的实现与配置：CRISP、MicroRank 和 mABC 均提供了公开源代码，我们直接使用了这些代码。对于其他最先进的方法，我们则根据各自论文中的描述进行了实现。实现完成后，我们通过实验对结果进行了微调，并为所有基线选择了最佳配置。其次，尽管我们的实证研究受益于多位专业站点可靠性工程师提供的见解，但其诊断流程固有的主观性可能会引入偏差。

6 相关工作

6.1 根因定位

根因定位旨在识别导致系统异常的服务、组件或操作。现有方法通常利用系统指标或分布式追踪数据，从互补的角度进行分析：指标突出显示异常的资源使用情况或性能偏差，而追踪数据则揭示详细的服务交互和请求流。

基于指标的方法分析响应时间、吞吐量和资源利用率等关键绩效指标。Microscope [21] 构建因果图，并采用深度优先搜索检测前端异常。CIRCA [17] 利用基于回归的假设检验和后代调整构建因果贝叶斯网络，以推断存在问题的组件。RUN [20] 将时间序列预测用于神经格兰杰因果发现，并结合个性化PageRank算法，高效推荐排名前 k 的根因。

基于追踪数据的方法跟踪执行路径和组件间交互，以实现细粒度定位。Micro-Rank [55] 构建追踪覆盖树，并应用PageRank为潜在根因评分。TraceRank [56] 将频谱分析与基于PageRank的随机游走相结合，以 pinpoint 异常服务。CRISP [79] 执行关键路径分析，深入定位延迟问题；而 TraceConstruct [59] 利用序列表示、对比序列模式挖掘和频谱分析，高效定位多维根因。

6.2 基于大语言模型的故障管理

大语言模型凭借先进的语义理解和逻辑推理能力，显著推动了故障管理领域的发展 [65]，并日益成为研究的焦点。人们提出了许多基于大语言模型的方法，以应对故障管理的各个方面，包括异常检测、故障诊断和自动化缓解 [4, 5, 10, 11, 15, 22–26, 29–31, 33, 41, 58, 60–62, 67, 77]。

一些研究专门针对故障管理开发了基础模型。例如，Lag-Llama [33]、Timer [26]和 TimesFM [5] 预训练了用于基于指标的异常检测的基础模型。类似地，ShellGPT [41]训练了一个能够自动生成 Shell 脚本以实现自动化缓解的模型。其他方法则采用微调策略，使大语言模型适用于故障管理任务。例如，AnomalyLLM [23] 和 UniTime [24] 采用全量微调进行异常检测，而 OWL [10]和 LogLM [25]则利用参数高效微调技术构建日志分析模型。

由于这些基于训练的方法需要大量计算资源，越来越多的方法转而依赖基于提示的技术。例如，RCACopilot [4] 和 Xpert [15] 利用上下文学习来组织诊断流程，从而确保根因分析的准确性。LM-PACE [60] 应用思维链推理，增强 GPT-4 分析事件报告的能力；Hamadanian 等人 [11] 则将这一方法扩展至根据事件报告生成缓解方案。此外，RAGLog [29] 和 Lo-gRAG [77] 利用检索增强生成（RAG），通过检索历史日志来提升基于日志的异常检测能力。

7 结论

本文提出了 AMER-RCL，这是一种面向微服务根因定位的智能体记忆增强递归推理框架。AMER-RCL 的设计源于我们对站点可靠性工程师（SRE）如何执行根因分析的实证研究。该框架采用递归推理根因定位引擎，为每条告警递归地细化深入的候选原因，并利用智能体记忆逐步积累和复用同一时间窗口内先前告警的推理结果，从而减少冗余探索并降低推理延迟。实验结果表明，在定位准确率和推理效率方面，AMER-RCL 均持续优于最先进的方法。

未来，我们将进一步探索如何利用小规模模型实现更准确、高效的根因定位。此外，我们还考虑将该方法扩展至涵盖整个故障管理流程。

致谢

本研究得到中国广东省重点研发计划（编号：2020B010164003）资助。

参考文献

[1] AIOPS 2022 锦标赛。https://competition.aiops.cn/。[2] 杨才，韩彪，李杰，赵娜和苏金树。2021。Modelcoder：一种基于故障模型的微服务系统自动根因定位框架。载于*2021 IEEE/ACM第29届国际服务质量研讨会（IWQOS）*。IEEE，1–6。[3] 陈浩，魏克刚，李安，王涛和张文博。2021。利用深度学习对微服务进行基于追踪的智能故障诊断。载于*2021 IEEE第45届计算机、软件与应用大会（COMPSAC）*。IEEE，884–893。[4] 陈银芳，谢怀兵，Minghua Ma，Yu Kang，高欣，史刘，曹云杰，高学东，范浩，温明等。2024。利用大语言模型对云事件进行自动根因分析。载于第十九届欧洲计算机系统会议论文集。674–688。[5] Abhimanyu Das，Weihao Kong，Rajat Sen和Yichen Zhou。2024。一种用于时间序列预测的仅解码器基础模型。载于第四十一届国际

机器学习大会。[6] Ruomeng Ding，Chaoyun Zhang，Lu Wang，Yong Xu，Minghua Ma，Xiaomin Wu，Meng Zhang，Qingjun Chen，高欣，高学东等。2023。Trace-diag：面向大规模微服务系统的自适应、可解释且高效的根因分析。载于第*31*届*ACM*欧洲软件工程联合会议暨软件工程基础研讨会论文集。1762–1773。[7] Chiming Duan，Minghua He，Pei Xiao，Tong Jia，Xin Zhang，Zhewei Zhong，Xiang Luo，Yan Niu，Lingzhe Zhang，Yifan Wu等。2025。LogAction：通过主动域适应实现基于日志的一致性跨系统异常检测。*arXiv*预印本 *arXiv:2510.03288*（2025）。[8] Chiming Duan，Yong Yang，Tong Jia，Guiyang Liu，Jinbu Liu，Huxing Zhang，Qi Zhou，Ying Li和Gang Huang。2025。FAMOS：通过有效的多模态数据融合对微服务系统进行故障诊断。载于*2025 IEEE/ACM*第*47*届国际软件工程大会（*ICSE*）。IEEE计算机学会，610–610。[9] Yu Gan，Yanqi Zhang，Kelvin Hu，Dailun Cheng，Yuan He，Meghna Pancholi和Christina Delimitrou。2019。Seer：利用大数据驾驭云微服务性能调试的复杂性。载于第二十四届编程语言与操作系统的体系结构国际会议论文集。19–33。[10] Hongcheng Guo，Jian Yang，Jiaheng Liu，Liqun Yang，Linzheng Chai，Jiaqi Bai，Junran Peng，Xiaorong Hu，Chao Chen，Dongfeng Zhang等。2023。Owl：面向IT运维的大语言模型。*arXiv*预印本 *arXiv:2309.09298*（2023）。[11] Pouya Hamadanian，Behnaz Arzani，Sadjad Fouladi，Siva Kesava Reddy Kakarla，Rodrigo Fonseca，Denizcan Billor，Ahmad Cheema，Edet Nkposong和Ranveer Chandra。2023。人工智能驱动的网络事件管理的全景视角。载于第*22*届*ACM*网络热点主题研讨会论文集。180–188。[12] Yongqi Han，Qingfeng Du，Ying Huang，Jiaqi Wu，Fulong Tian和Cheng He。2024。单次故障根因分析的潜力：大语言模型与小型分类器的协作。载于第*39*届*IEEE/ACM*自动化软件工程国际会议论文集。931–943。[13] Minghua He，Chiming Duan，Pei Xiao，Tong Jia，Siyu Yu，Lingzhe Zhang，Weijie Hong，Jin Han，Yifan Wu，Ying Li等。2025。团结就是力量：通过交互式多任务学习实现端到端的基于日志的故障诊断。*arXiv*预印本 *arXiv:2509.24364*（2025）。[14] Minghua He，Tong Jia，Chiming Duan，Pei Xiao，Lingzhe Zhang，Kangjin Wang，Yifan Wu，Ying Li和Gang Huang。2025。言行一致：你的基于日志的软件可靠性维护系统真的可靠吗？*arXiv*预印本 *arXiv:2509.24352*（2025）。[15] Yuxuan Jiang，Chaoyun Zhang，Shilin He，Zhihao Yang，Minghua Ma，Si Qin，Yu Kang，Yingnong Dang，Saravan Rajmohan，Qingwei Lin等。2024。Xpert：利用大语言模型和查询推荐赋能事件管理。载于*IEEE/ACM*第*46*届国际软件工程大会论文集。1–13。[16] Yuyuan Kang，Xiangdong Huang，Shaoxu Song，Lingzhe Zhang，Jialin Qiao，Chen Wang，Jianmin Wang和Julian Feinauer。2022。是否分离：关于分层LSM树中乱序时间序列数据的处理。载于*2022 IEEE*第*38*届国际数据工程会议（*ICDE*）。IEEE，3340–3352。[17] Mingjie Li，Zeyan Li，Kanglin Yin，Xiaohui Nie，Wenchi Zhang，Kaixin Sui和Dan Pei。2022。基于因果推断的在线服务系统根因分析与干预识别。载于第*28*届*ACM SIGKDD*知识发现与数据挖掘会议论文集。3230–3240。[18] Yichen Li，Yulun Wu，Jinyang Liu，Zhihan Jiang，Zhuangbin Chen，Guangba Yu和Michael R Lyu。2025。COCA：利用代码知识对分布式系统进行生成式根因分析。载于*2025 IEEE/ACM*第*47*届国际软件工程大会（*ICSE*）。IEEE，1346–1358。[19] Zeyan Li，Junjie Chen，Rui Jiao，Nengwen Zhao，Zhijun Wang，Shuwei Zhang，Yanjun Wu，Long Jiang，Leiqin Yan，Zikai Wang等。2021。通过轨迹分析实现微服务系统的实用根因定位。载于*2021 IEEE/ACM*第*29*届国际服务质量研讨会（*IWQOS*）。IEEE，1–10。[20] Cheng-Ming Lin，Ching Chang，Wei-Yao Wang，Kuang-Da Wang和Wen-Chih Peng。2024。利用神经格兰杰因果发现进行微服务根因分析。载于*AAAI*人工智能会议论文集，第38卷。206–213。[21] JinJin Lin，Pengfei Chen和Zibin Zheng。2018。Microscope：利用微服务环境中的因果图精准定位性能问题。载于面向服务的计算：第*16*届国际会议，*ICSOC 2018*，中国杭州，*2018*年*11*月*12*日至*15*日，论文集*16*。Springer，3–20。[22] Hongyi Liu，Yinping Ma，Xiaosong Huang，Lingzhe Zhang，Tong Jia和Ying Li。2025。ORA：利用在线检索增强语言模型预测高性能计算平台上的作业运行时间。载于第*39*届*ACM*国际超级计算会议论文集。884–894。[23] Shuo Liu，Di Yao，Lanting Fang，Zhetao Li，Wenbin Li，Kaiyu Feng，XiaoWen Ji和Jingping Bi。2024。AnomalyLLM：利用大语言模型对动态图进行少样本异常边检测。*arXiv*预印本 *arXiv:2405.07626*（2024）。[24] Xu Liu、Junfeng Hu、Yuan Li、Shizhe Diao、Yuxuan Liang、Bryan Hooi 和 Roger Zimmermann。2024。UniTime：一种语言增强的统一模型，用于跨领域时间序列预测。载于《ACM Web Conference 2024论文集》。4095–4106。[25] Yilun Liu、Yuhe Ji、Shimin Tao、Minggui He、Weibin Meng、Shenglin Zhang、Yongqian Sun、Yuming Xie、Boxing Chen 和 Hao Yang。2024。LogLM：从任务驱动到指令驱动的自动化日志分析。arXiv预印本 arXiv:2410.09352（2024）。[26] Yong Liu、Haoran Zhang、Chenyu Li、Xiangdong Huang、Jianmin Wang 和 Mingsheng Long。2024。Timer：生成式预训练变换器是大型时间序列模型。载于第41届国际机器学习会议论文集。32369–32399。[27] Nabor C Mendonça、Pooyan Jamshidi、David Garlan 和 Claus Pahl。2019。开发自适应微服务系统：挑战与方向。《IEEE Software》38卷2期（2019），70–79。[28] Annamalai Narayanan、Mahinthan Chandramohan、Rajasekar Venkatesan、Lihui Chen、Yang Liu 和 Shantanu Jaiswal。2017。Graph2Vec：学习图的分布式表示。arXiv预印本 arXiv:1707.05005（2017）。[29] Jonathan Pan、Wong Swee Liang 和 Yuan Yidi。2024。RAGLog：使用检索增强生成进行日志异常检测。载于2024年IEEE公共安全技术世界论坛（WFPST）。IEEE，169–174。[30] Leyi Pan、Zheyu Fu、Yunpeng Zhai、Shuchang Tao、Sheng Guan、Shiyu Huang、Lingzhe Zhang、Zhaoyang Liu、Bolin Ding、Felix Henry 等。2025。Omni-SafetyBench：音视频大语言模型安全性评估基准。arXiv预印本 arXiv:2508.07173（2025）。[31] Leyi Pan、Shuchang Tao、Yunpeng Zhai、Zheyu Fu、Liancheng Fang、Minghua He、Lingzhe Zhang、Zhaoyang Liu、Bolin Ding、Aiwei Liu 等。2025。d-TreeRPO：迈向更可靠的扩散语言模型策略优化。arXiv预印本 arXiv:2512.09675（2025）。[32] Changhua Pei、Zexin Wang、Fengrui Liu、Zeyan Li、Yang Liu、Xiao He、Rong Kang、Tieying Zhang、Jianjun Chen、Jianhui Li 等。2025。Flow-of-Action：标准操作流程增强型基于大语言模型的多智能体系统，用于根因分析。载于《ACM Web Conference 2025 伴随论文集》。422–431。[33] Kashif Rasul、Arjun Ashok、Andrew Robert Williams、Arian Khorasani、George Adamopoulos、Rishika Bhagwatkar、Marin Biloš、Hena Ghonia、Nadhir Hassen、Anderson Schneider 等。2023。Lag-Llama：迈向时间序列预测基础模型。载于R0-FoMo：大型基础模型中少样本与零样本学习的鲁棒性。[34] Rui Ren。2025。基于蒙特卡洛树搜索方法的多智能体故障定位系统。arXiv预印本 arXiv:2507.22800（2025）。[35] Rui Ren、Yang Wang、Fengrui Liu、Zhenyu Li、Gareth Tyson、Tianhao Miao 和 Gaogang Xie。2023。Grace：利用图卷积网络对微服务进行可解释的根因分析。载于2023年IEEE/ACM第31届服务质量国际研讨会（IWQoS）。IEEE，1–4。[36] Devjeet Roy、Xuchao Zhang、Rashi Bhave、Chetan Bansal、Pedro Las-Casas、Rodrigo Fonseca 和 Saravan Rajmohan。2024。探索用于根因分析的基于大语言模型的智能体。arXiv预印本 arXiv:2403.04123（2024）。[37] Komal Sarda、Zakeya Namrud、Marin Litoiu、Larisa Shwartz 和 Ian Watts。2024。利用大语言模型自动修复微服务应用：一项实验研究。载于第32届ACM软件工程基础国际会议伴随论文集。358–369。[38] Shiwen Shan、Yintong Huo、Yuxin Su、Yichen Li、Dan Li 和 Zibin Zheng。2024。Face It Yourselves：一种基于大语言模型的两阶段策略，通过日志定位配置错误。arXiv预印本 arXiv:2404.00640（2024）。[39] Junxian Shen、Han Zhang、Yang Xiang、Xingang Shi、Xinrui Li、Yunxi Shen、Zijian Zhang、Yongxiang Wu、Xia Yin、Jilong Wang 等。2023。以网络为中心的分布式追踪：利用DeepFlow实现无需代码的微服务故障排查。载于ACM SIGCOMM 2023会议论文集。420–437。[40] Honghao Shi、Longkai Cheng、Wenli Wu、Yuhang Wang、Xuan Liu、Shaokai Nie、Weixv Wang、Xuebin Min、Chunlei Men 和 Yonghua Lin。2024。增强集群韧性：基于大语言模型智能体的自主智能集群诊断系统及评估框架。arXiv预印本 arXiv:2411.05349（2024）。[41] Jie Shi、Sihang Jiang、Bo Xu、Jiaqing Liang、Yanghua Xiao 和 Wei Wang。2023。ShellGPT：用于Shell语言理解的生成式预训练变换器模型。载于2023年IEEE第34届软件可靠性工程国际研讨会（ISSRE）。IEEE，671–682。[42] Yongqian Sun、Zihan Lin、Binpeng Shi、Shenglin Zhang、Shiyu Ma、Pengxiang Jin、Zhenyu Zhong、Lemeng Pan、Yicheng Guo 和 Dan Pei。2025。基于图自编码器的微服务系统可解释故障定位。《ACM软件工程与方法学汇刊》34卷2期（2025），1–28。[43] Dongjie Wang、Zhengzhang Chen、Jingchao Ni、Liang Tong、Zheng Wang、Yanjie Fu 和 Haifeng Chen。2023。用于根因定位的相互依赖因果网络。载于第29届ACM SIGKDD知识发现与数据挖掘会议论文集。5051–5060。[44] Kangjin Wang、Ying Li、Cheng Wang、Tong Jia、Kingsum Chow、Yang Wen、Yaoyong Dou、Guoyao Xu、Chuanjia Hou、Jie Yao 等。2022。大规模刻画作业微体系结构特征：数据集与分析。载于第51届国际并行处理会议论文集。1–11。

[45] Lu Wang、Chaoyun Zhang、Ruomeng Ding、Yong Xu、Qihang Chen、Wentao Zou、Qingjun Chen、Meng Zhang、高学东、范浩等。2023。利用人类反馈的分层强化学习对微服务系统进行根因分析。载于第*29*届*ACM SIGKDD*知识发现与数据挖掘会议论文集。5116–5125。[46] Qi Wang、Xiao Zhang、Mingyi Li、Yuan Yuan、Mengbai Xiao、Fuzhen Zhuang 和 Dongxiao Yu。2025。TAMO：在云原生系统中，借助工具辅助型大语言模型智能体和多模态观测数据进行细粒度根因分析。*arXiv*预印本 *arXiv:2504.20462*（2025）。[47] Tingting Wang、Guilin Qi 和 Tianxing Wu。2024。KGroot：一种知识图谱增强的根因分析方法。《*Expert Systems with Applications*》255（2024），124679。[48] Zefan Wang、Zichuan Liu、Yingying Zhang、Aoxiao Zhong、Jihong Wang、Fengbin Yin、Lunting Fan、Lingfei Wu 和 Qingsong Wen。2024。RCAgent：利用工具增强型大语言模型的自主智能体进行云根因分析。载于第*33*届*ACM*国际信息与知识管理大会论文集。4966–4974。[49] Muhammad Waseem、Peng Liang、Mojtaba Shahin、Amleto Di Salle 和 Gastón Márquez。2021。微服务系统的设计、监控与测试：从业者视角。《*Journal of Systems and Software*》182（2021），111061。[50] Jason Wei、Xuezhi Wang、Dale Schuurmans、Maarten Bosma、Fei Xia、Ed Chi、Quoc V Le、Denny Zhou 等。2022。思维链提示能够引发大语言模型的推理。《神经信息处理系统进展》35（2022），24824–24837。[51] Zhe Xie、Shenglin Zhang、Yitong Geng、Yao Zhang、Minghua Ma、Xiaohui Nie、Zhenhe Yao、Longlong Xu、Yongqian Sun、Wentao Li 等。2024。通过潜在空间中的干预识别，在有限可观测性下对微服务进行根因分析。载于第*30*届*ACM SIGKDD*知识发现与数据挖掘会议论文集。6049–6060。[52] Zhiqiang Xie、Yujia Zheng、Lizi Ottens、Kun Zhang、Christos Kozyrakis 和 Jonathan Mace。2024。Cloud Atlas：利用语言模型和因果洞察，高效定位云系统故障。*arXiv*预印本 *arXiv:2407.08694*（2024）。[53] Linwei Yang、Jing Li、Kuanzhi Shi、Songlin Yang、Qingfu Yang 和 Jiangang Sun。2022。MicroMILTS：基于互信息和LSTM自编码器的微服务故障定位。载于*2022*年第*23*届亚太网络运营与管理研讨会（*APNOMS*）。IEEE，1–6。[54] Yong Yang、Long Wang、Jing Gu 和 Ying Li。2022。捕获请求执行路径，以理解服务行为并在不插桩代码的情况下检测异常。《*IEEE*服务计算汇刊》16卷2期（2022），996–1010。[55] Guangba Yu、Pengfei Chen、Hongyang Chen、Zijie Guan、Zicheng Huang、Linxiao Jing、Tianjun Weng、Xinmeng Sun 和 Xiaoyun Li。2021。MicroRank：在微服务环境中通过扩展频谱分析定位端到端延迟问题。载于*2021*年万维网大会论文集。3087–3098。[56] Guangba Yu、Zicheng Huang 和 Pengfei Chen。2023。TraceRank：在云原生系统中利用拆分的端到端追踪数据进行异常服务定位。《软件：演化与过程》35卷10期（2023），e2413。[57] Qingyang Yu、Nengwen Zhao、Mingjie Li、Zeyan Li、Honglin Wang、Wenchi Zhang、Kaixin Sui 和 Dan Pei。2024。IT服务中告警与事件智能管理综述。《网络与计算机应用杂志》（2024），103842。[58] Yunpeng Zhai、Shuchang Tao、Cheng Chen、Anni Zou、Ziqian Chen、Qingxu Fu、Shinji Mai、Li Yu、Jiaji Deng、Zouying Cao 等。2025。AgentEvolver：迈向高效的自我演化智能体系统。*arXiv*预印本 *arXiv:2511.10395*（2025）。[59] Chenxi Zhang、Zhen Dong、Xin Peng、Bicheng Zhang 和 Miao Chen。2024。基于追踪的微服务系统性能问题多维根因定位。载于*IEEE/ACM*第*46*届软件工程国际会议论文集。1–12。[60] Dylan Zhang、Xuchao Zhang、Chetan Bansal、Pedro Las-Casas、Rodrigo Fonseca 和 Saravan Rajmohan。2024。LM-PACE：利用大语言模型进行置信度估计，以有效定位云事件的根因。载于第*32*届*ACM*软件工程基础国际会议伴随论文集。388–398。[61] Lingzhe Zhang、Liancheng Fang、Chiming Duan、Minghua He、Leyi Pan、Pei Xiao、Shiyu Huang、Yunpeng Zhai、Xuming Hu、Philip S Yu 等。2025。并行文本生成综述：从并行解码到扩散语言模型。*arXiv*预印本 *arXiv:2508.08712*（2025）。[62] Lingzhe Zhang、Tong Jia、Mengxi Jia 和 Ying Li。2025。LogDB：分布式数据库的基于多变量日志的故障诊断（由MultiLog扩展而来）。*arXiv*预印本 *arXiv:2505.01676*（2025）。[63] Lingzhe Zhang、Tong Jia、Mengxi Jia、Ying Li、Yong Yang 和 Zhonghai Wu。2024。分布式数据库的基于多变量日志的异常检测。载于第*30*届*ACM SIGKDD*知识发现与数据挖掘会议论文集。4256–4267。

[64] Lingzhe Zhang、Tong Jia、Mengxi Jia、Hongyi Liu、Yong Yang、Zhonghai Wu 和 Ying Li。2024。迈向近乎零运行时采集开销：基于 Raft 的分布式存储系统系统故障异常诊断。IEEE服务计算汇刊（2024）。[65] Lingzhe Zhang、Tong Jia、Mengxi Jia、Yifan Wu、Aiwei Liu、Yong Yang、Zhonghai Wu、Xuming Hu、Philip Yu 和 Ying Li。2025。大语言模型时代的智能运维综述。计算调查（2025）。[66] Lingzhe Zhang、Tong Jia、Mengxi Jia、Yifan Wu、Hongyi Liu 和 Ying Li。2025。ScalaLog：使用大语言模型进行可扩展的基于日志的故障诊断。收录于 ICASSP 2025-2025 IEEE声学、语音与信号处理国际会议（ICASSP）。IEEE，1–5。[67] Lingzhe Zhang、Tong Jia、Mengxi Jia、Yifan Wu、Hongyi Liu 和 Ying Li。2025。XRAGLog：一种使用检索增强生成的高效、上下文感知型基于日志的异常检测方法。收录于 AAAI 2025 防止和检测大语言模型错误信息研讨会（PDLM）。[68] Lingzhe Zhang、Tong Jia、Xinyu Tan、Xiangdong Huang、Mengxi Jia、Hongyi Liu、Zhonghai Wu 和 Ying Li。2025。E-log：面向数据库的细粒度弹性基于日志的异常检测与诊断。IEEE服务计算汇刊（2025）。[69] Lingzhe Zhang、Tong Jia、Kangjin Wang、Weijie Hong、Chiming Duan、Minghua He 和 Ying Li。2025。使用多智能体递归思维进行微服务系统的自适应根因定位。arXiv 预印本 arXiv:2508.20370（2025）。[70] Lingzhe Zhang、Tong Jia、Kangjin Wang、Mengxi Jia、Yong Yang 和 Ying Li。2024。通过减少事件增强基于日志的异常检测模型：一项实证研究。收录于第18届 ACM/IEEE实证软件工程与测量国际研讨会论文集。538–548。[71] Lingzhe Zhang、Yunpeng Zhai、Tong Jia、Chiming Duan、Minghua He、Leyi Pan、Zhaoyang Liu、Bolin Ding 和 Ying Li。2025。MicroRemed：微服务修复中的大语言模型基准测试。arXiv 预印本 arXiv:2511.01166（2025）。[72] Lingzhe Zhang、Yunpeng Zhai、Tong Jia、Chiming Duan、Siyu Yu、Jinyang Gao、Bolin Ding、Zhonghai Wu 和 Ying Li。2025。ThinkFL：通过强化微调实现微服务系统的自我改进故障定位。arXiv 预印本 arXiv:2504.18776（2025）。[73] Lingzhe Zhang、Yunpeng Zhai、Tong Jia、Xiaosong Huang、Chiming Duan 和 Ying Li。2025。Agentfm：面向分布式数据库、由大语言模型驱动的多智能体角色感知型故障管理。收录于第33届 ACM软件工程基础国际会议论文集。525–529。[74] Ling-Zhe Zhang、Xiang-Dong Huang、Yan-Kai Wang、Jia-Lin Qiao、Shao-Xu Song 和 Jian-Min Wang。2024。时间分层压实：一种面向基于 LSM树的时序数据库的弹性压实方案。《先进工程信息学》59（2024），102224。[75] Shenglin Zhang、Sibo Xia、Wenzhao Fan、Binpeng Shi、Xiao Xiong、Zhenyu Zhong、Minghua Ma、Yongqian Sun 和 Dan Pei。2024。微服务系统故障诊断：全面综述与分析。ACM软件工程与方法学汇刊（2024）。[76] Wei Zhang、Hongcheng Guo、Jian Yang、Zhoujin Tian、Yi Zhang、Yan Chaoran、Zhoujun Li、Tongliang Li、Xu Shi、Liangfan Zheng 等。2024。mABC：用于微服务架构根因分析的区块链启发式多智能体协作。收录于计算语言学协会 EMNLP 2024 论文集。4017–4033。[77] Wanhao Zhang、Qianli Zhang、Enyu Yu、Yuxiang Ren、Yeqing Meng、Mingxi Qiu 和 Jilong Wang。2024。LogRAG：通过检索增强生成进行半监督式基于日志的异常检测。收录于2024年 IEEE国际Web服务会议（ICWS）。IEEE，1100–1102。[78] Xuchao Zhang、Supriyo Ghosh、Chetan Bansal、Rujia Wang、Minghua Ma、Yu Kang 和 Saravan Rajmohan。2024。使用上下文学习和 GPT-4 自动分析云事件根因。收录于第32届 ACM软件工程基础国际会议伴随论文集。266–277。[79] Zhizhou Zhang、Murali Krishna Ramanathan、Prithvi Raj、Abhishek Parwal、Timothy Sherwood 和 Milind Chabbi。2022。{CRISP}：大规模微服务架构的关键路径分析。收录于2022年 USENIX年度技术会议（USENIX ATC 22）。655–672。[80] Xiang Zhou、Xin Peng、Tao Xie、Jun Sun、Chao Ji、Wenhai Li 和 Dan Ding。2018。微服务系统的故障分析与调试：工业调查、基准系统与实证研究。IEEE软件工程汇刊 47，2（2018），243–260。[81] Xiang Zhou、Xin Peng、Tao Xie、Jun Sun、Chenjie Xu、Chao Ji 和 Wenyun Zhao。2018。面向软件工程研究的微服务系统基准测试。收录于第40届国际软件工程大会伴随论文集。323–324。[82] Zhouruixing Zhu、Cheryl Lee、Xiaoying Tang 和 Pinjia He。2024。HeMiRCA：利用异构数据源对微服务进行细粒度根因分析。ACM软件工程与方法学汇刊 33，8（2024），1–25。