面向韧性分布式计算连续体系统的仿生智能体自愈框架

Alaa Saleh、Praveen Kumar Donta、Roberto Morabito、Sasu Tarkoma、Anders Lindgren、Qiyang Zhang、Schahram Dustdar、Susanna Pirttikangas 和 Lauri Lov´en

arXiv:2601.00339v1 [cs.AI] 1 Jan 2026

摘要——**人类生物系统凭借非凡的韧性维持生命，不断检测损伤、协调有针对性的响应，并通过自愈恢复功能。受这些能力启发，本文提出 ReCiSt，一种旨在提升分布式计算连续体系统（DCCS）韧性的仿生智能体自愈框架。现代 DCCS 集成了异构计算资源，范围涵盖资源受限的物联网设备到高性能云基础设施；其固有的复杂性、移动性和动态运行条件使其频繁遭遇故障，进而中断服务连续性。这些挑战凸显了对可扩展、自适应且可自我调节的韧性策略的需求。ReCiSt 将生物学中的止血、炎症、增殖和重塑阶段，映射为 DCCS 的计算层：遏制、诊断、元认知和知识。这四层借助大语言模型（LM）驱动的智能体，实现自主故障隔离、因果诊断、自适应恢复和长期知识整合。这些智能体能够解读异构日志、推断根本原因、优化推理路径，并在尽量减少人工干预的情况下重新配置资源。本文使用多个大语言模型在公开故障数据集上评估所提出的 ReCiSt 框架；由于类似方法十分有限，未纳入基线对比。尽管如此，在不同大语言模型下进行的评估结果证实，ReCiSt 能够在数十秒内实现自愈，智能体的 CPU 使用率最低为 10%。结果还表明，该框架具备深入分析能力，能够应对不确定性，并能通过调用数量不等的微型智能体实现韧性。**

索引词—**计算连续体系统、自愈、韧性、资源受限、多智能体系统**

一、引言

HUMAN 类是地球上最聪明的生物物种之一。人体作为一个高度优化的分布式生态系统，由约 37.2 万亿个特化细胞组成。这些细胞通过神经、心血管、呼吸、肌肉和骨骼等器官系统协同工作，各自执行局部计算，同时维持整体生理稳定。例如，神经系统体现了分布式处理：心脏含有约 40000 个神经元，用于调节心律；肠道（第二大脑）由近 5 亿个神经元组成，负责自主消化控制；每个视网膜含有超过 1 亿个神经元，用于处理视觉信号；脊髓则容纳数百万个神经元，以执行低延迟的反射反应 [1]。这种既有层级结构又去中心化的生物架构，与一种名为分布式计算连续体系统（DCCS）的现代计算范式相似 [2]，该范式将边缘设备、中间层雾节点和云基础设施整合为统一的计算织构，能够根据延迟、能耗约束和计算负载分配任务。DCCS 面临着重大运营挑战，原因在于其基础设施具有异构性且持续演进。各节点在计算能力、存储、连接性和可靠性方面各不相同，因此需要能够适应波动的工作负载、节点移动性和动态网络状况的编排机制。人体也具有类似的内在波动。

鉴于这些相似之处，将生物模型融入 DCCS，为应对其若干核心挑战提供了一种很有前景的方法，正如 [1] 中所讨论的那样。本文通过将生物学自我调节过程映射到 DCCS，以实现自愈行为并增强系统韧性，探索并模拟了这一方向。人体为分布式自我调节提供了一个天然模型 [3]，，因为它能够检测中断、隔离受损区域，并在维持整体稳定性的同时恢复功能。伤口愈合、免疫反应和分布式神经元决策等过程，协调异构组件间的感知、反馈和适应，展现出一种可扩展且去中心化的韧性形式。将这些原则应用于 DCCS 架构，使系统能够迅速遏制故障、重新配置受影响节点，并在动态且不确定的条件下维持服务连续性。

A. 生物学基础

生物伤口愈合是一个多阶段过程，具有内在能力，可检测组织损伤、启动有针对性的响应，并在无需外部干预的情况下恢复功能完整性。为提供必要的生物学背景

A. Saleh（通讯作者）、S. Pirttikangas 和 L. Lov´en 任职于奥卢大学应用计算中心（芬兰奥卢，邮编 90014）； P. Donta 任职于斯德哥尔摩大学计算机与系统科学系（瑞典斯德哥尔摩，邮编 106 91）； R. Morabito 任职于 EURECOM 通信系统系（法国比奥，邮编 06410）； S. Tarkoma 任职于赫尔辛基大学计算机科学系（芬兰赫尔辛基，邮编 00100）； A. Lindgren 任职于瑞典RISE研究院（瑞典斯德哥尔摩，邮编 166 40）以及吕勒奥理工大学计算机科学、电气与空间工程系（瑞典吕勒奥，邮编 971 87）； Q. Zhang 任职于北京大学计算机科学学院（中国北京，邮编 100087）； S. Dustdar 任职于维也纳工业大学分布式系统组（奥地利维也纳，邮编 1040）和 ICREA（西班牙巴塞罗那，邮编 08002）。经费来源：本研究获芬兰研究理事会通过 6G旗舰计划（资助编号 318927）和 CO2CREATION战略研究理事会项目（资助编号 372355）资助，获芬兰商业促进局通过神经发布/订阅研究项目（项目登记号 8754/31/2022）资助，并获欧洲区域发展基金资助（项目编号 A81568 、 A91867 ）。

供计算领域的读者参考，我们概述伤口愈合的四个阶段 [4]，包括止血、炎症、增殖和重塑期。

1）止血：是伤口愈合的初始阶段，也是身体对血管损伤作出的即时反应。组织损伤发生后，局部血管会收缩，以限制血流并减少失血。随后，血小板在凝血酶和暴露的纤维状胶原的作用下被激活，这一过程还受到胶原相关氨基酸脯氨酸和羟脯氨酸的支持。活化的血小板黏附于胶原基质并聚集，形成初始的血小板栓，同时释放纤维蛋白原等介质，促进进一步聚集。其他介质则增强血小板对胶原的黏附，并将更多血小板招募至损伤部位。与此同时，内皮细胞会产生前列环素，以防止血小板过度聚集。随后，血小板–纤维蛋白原复合物转化为纤维蛋白，形成稳定的聚合物网络。这一纤维蛋白网形成止血血凝块，封闭伤口并防止进一步失血。

2）炎症：在炎症阶段，身体会启动防御反应，以预防感染并清除细胞碎屑。血管舒张会增加流向伤口部位的血流，并促进免疫细胞、氧气和必需营养物质输送至该处。白细胞迁移至受伤区域，以清除细菌、病原体和受损细胞。与此同时，身体会释放多种生长因子，以刺激组织修复过程并招募更多参与愈合的细胞。

3）增殖：这一阶段着重于受损组织的愈合与重建。其特点是以新的细胞外基质取代暂时性纤维蛋白基质，新基质由胶原纤维、蛋白聚糖和纤维连接蛋白组成，从而恢复组织完整性和功能。此阶段的一个关键事件是血管生成，即形成新的毛细血管，以替代受损的脉管系统，并确保为再生组织提供充足的氧气和营养。成纤维细胞是这一阶段的主要效应细胞，在血小板和巨噬细胞释放的因子作用下迁移至伤口部位。它们沿着细胞外基质中纤维状结构的排列方向迁移，并受到局部蛋白水解酶分泌的促进。

一旦定位于伤口处，成纤维细胞便会增殖，并合成纤维连接蛋白、透明质酸、胶原蛋白和蛋白聚糖等基质成分，支持新基质的构建和细胞向内生长。血管生成通过氧依赖性调控得以进行：缺氧会增加缺氧诱导因子（HIF）的水平，进而诱导血管内皮生长因子（VEGF）促进新生血管形成；而再氧合会降解 HIF 并降低 VEGF 水平。随着血管化程度提高，成纤维细胞增殖减少，上皮化修复表皮屏障，肌成纤维细胞收缩伤口，使其面积缩小。

4）重塑/成熟：在组织修复的这一阶段，新形成的组织逐渐增强、重组并完善其功能。胶原纤维重新排列并重塑，以增强再生组织的抗拉强度和弹性，从而有助于恢复结构完整性。与此同时，

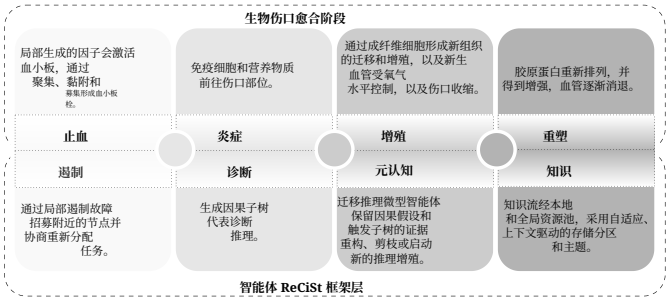


图 1：将生物伤口愈合阶段的功能映射到 ReCiSt 框架的自愈层

早期愈合阶段增生的血管网络开始退化。因此，伤口逐渐失去其特有的红色或粉红色，标志着组织成熟的完成。

B. 动机

基于这些观察，我们提出 ReCiSt，这是一种面向弹性 DCCS 的仿生智能体自愈架构。图 1 展示了生物伤口愈合阶段与 ReCiSt 框架中自愈层之间的映射关系。止血对应系统的即时故障响应，此时开始隔离并缓解故障；在 ReCiSt 中，这项功能由遏制层执行，该层通过协商将受影响的服务重新路由至稳定的邻近节点，以防止级联中断。炎症期反映了生物免疫反应，与诊断层相对应；诊断层在此收集并分析运行数据，以确定故障的性质和范围。在增殖期（与新组织和血管的形成相关），元认知层支持微型智能体增殖、动态推理，并通过更新路由表建立新的通信路径。最后，重塑期中生物组织得到强化，这对应知识层；知识层通过协调本地和全局会合点（RP），在分布式连续体系统中传播知识。

ReCiSt 框架旨在检测故障、诊断其根本原因、调节内部推理过程，并优化其分布式知识结构，从而在不确定条件下实现一致的系统性能。这些能力由大语言模型（大语言模型）驱动的智能体提供支持：它们执行局部遏制、开展因果发现、通过元认知机制调节内部推理，并管理能够响应情境漂移而重新组织的自适应知识共享结构。要实现这种程度的自适应能力，每个阶段都需要具备智能体能力，使系统能够调节自身的推理过程，并随着运行需求的变化调整内部决策结构 [5], [6]。

C. 贡献

- 我们提出 ReCiSt 框架，旨在提供一种自适应的智能体自愈系统，该系统通过……

防御性响应，并通过反射式本地遏制发起这些响应，同时发现受人体自愈机制（即伤口愈合）启发的因果依赖。

- ReCiSt 通过迁移式微型智能体，实现智能体内部推理的自适应自我调节。
- ReCiSt 通过本地和全局资源池，利用自适应、上下文驱动的存储支持知识共享。
- 我们的框架旨在有效处理异构数据集，这些数据集在规模、结构、故障特征和运行环境方面各不相同。

我们实现了 ReCiSt 框架的原型，并使用不同的大语言模型在多个公开 DCCS 数据集上对其进行评估。结果表明，该框架能够有效自愈，同时缩短恢复时间、控制智能体资源开销并提升决策质量。

II. 相关工作

韧性正成为分布式基础设施领域的一项重要研究重点，在通信领域尤为如此。近期的 6G 路线图明确将韧性列为未来网络架构标准化的核心目标之一[7]。Altaweel 等人 [8]提出了一种面向任务关键型雾和边缘部署的基于身份的路由协议，可在动态网络状况下实现自适应路由。类似地，Nakayama 等人 [9] 为移动网络中的多路径通信设计了一种具有韧性的架构。在 [10],中，研究者通过无服务器运行时设计，探索了基于日志的故障容错方案，以应对动态工作负载。

值得关注的贡献通过以数据和工作负载为导向的策略凸显了韧性，这些策略利用机器学习（ML）进行预测与协调。Sen 等人 [11] 提出了一种弹性边缘云架构，结合服务器故障预测、优化后的虚拟机（VM）迁移和多跳路由，以确保服务无缝连续性。D´ıaz 等人 [12]采用联邦机器学习故障预测与优化启发式方法，选择部署配置以提升边缘工作负载的容错能力；Kashyap 等人 [13] 则专注于动态雾环境中的主动资源分配，通过预测每项任务的资源需求并指导任务分区来降低节点故障风险。这些解决方案虽然具备容错能力，但依赖大量训练数据和学习周期，因此难以快速响应突发或不可预测的系统变化。

近期的其他研究也强调了在不确定条件下区分故障与性能下降的困难。例如， [14] 提出采用基于闲谈协议的通信与迁移智能体，作为一种有效的去中心化检测机制。后续框架引入了知识驱动且动态生成的自愈智能体，使其能够进行预测、诊断和自主服务重新部署 [15]。其他进展则实现了多智能体工作流的自动生成与演化优化 [16] ，并开发了结构化的上下文管理架构，为大语言模型（LLM）驱动的智能体配备了以内存为版本控制文件系统的机制，以支持连贯的分布式推理 [17]。新兴的自学习范式进一步将任务生成、策略优化和奖励评估结合到闭环过程中，迭代优化智能体能力[18]。具有专门角色的 LLM 驱动多智能体系统通过联合分析流量模式、监控性能并检测可疑活动，进一步扩展了这些能力，从而确定用于自适应网络管理的优化缓解策略 [19]。

在通信网络中，智能体机制运用意图感知推理，在条件波动时支持实时资源分配 [20]。与此同时，融合层次推理、多流水线和经过微调的小型大语言模型的诊断流水线，能够提升云网络中的根因分析和故障定位能力 [21], ；基于大语言模型的主动容错框架则在边缘网络中协调遏制、诊断和恢复工具，以便在服务质量下降前缓解故障 [22]。这些研究试图实现完全自主且具备嵌入式自我改进能力的智能体机制，但在资源受限环境中尚未取得理想的解决方案。

鉴于我们重点关注仿生策略，我们发现该领域的现有研究仍相对有限。例如， [23]展示了生物机制如何帮助网络持续重组，从而在异构感知环境中维持可扩展性和鲁棒性。通过将分子生物学映射到系统级和设备级控制过程，类似的理念也启发了自主网络管理 [24]„ 并使硬件能够在出现结构故障时不断演进并自我修复 [25]。多智能体仿生框架进一步表明，在复杂的调度中断期间，自管理实体可以通过协商恢复性能[26]。近期的通信网络架构整合了进化、免疫和神经模型，以加快故障检测和修复 [27], ；受生物学启发的机器学习框架则运用群体和免疫隐喻，实现故障发生后的快速分布式恢复 [28]。主动推理已被提出作为 DCCS 的一种弹性策略 [29]„ 其根植于仿生模型，旨在模拟人脑持续推理和适应的能力。

大多数现有文献侧重于高效的故障检测和自主恢复，并以基于人工智能或机器学习的决策机制为支撑。然而，这些方法往往受限于对预训练模型、外部提供的决策策略以及大型数据集的依赖，因此难以处理此前未曾出现或不断演变的故障。此外，其计算和数据密集特性降低了它们在异构资源能力的分布式环境中的实用性。要克服这些局限，就需要自愈机制能够推理中断原因、适应内部决策过程，并动态重组知识结构，从而在不确定条件下维持性能。

III. 系统模型与问题表述

DCCS 可形式化表示为图 $G = (\mathcal{N}, E)$，其中 $\mathcal{N} = \{N_1, N_2, \ldots, N_n\}$ 是异构节点集合（涵盖物联网设备、边缘、雾和云资源）， E 是这些节点之间的通信链路集合。每个节点 N_i 由其属性（$\mathcal{C}_i$， $\mathcal{M}_i$， $\mathcal{S}_i$， $\mathcal{V}_i$）描述，其中 $\mathcal{C}_i$ 表示计算能力

（例如 CPU/GPU 周期），$\mathcal{M}_i$ 表示设备存储状态；设备状态包括宕机、可用（有可用于运行更多任务的空间）、繁忙（资源被任务占用）和恢复中，分别表示 $\mathcal{S}_i \in \{00, 11, 01, 10\}$；$\mathcal{V}_i \in \{00, 11, 01, 10\}$ 表示对应的 {低、中、高、严重} 漏洞。任务 $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_m\}$ 动态到达，需要映射到 N_i 上执行。假设节点 N_i 可以根据其（$\mathcal{C}_i$、$\mathcal{M}_i$、$\mathcal{S}_i$、$\mathcal{V}_i$）计算一个或多个 $\mathcal{T}$。时刻 t 的任务分配可由分配函数 $A(t) : \mathcal{T} \to \mathcal{N}$，$A(t, \tau_j) = N_i$ 指定，使得 $A(t, \tau_j) = N_i$ 在满足 $\mathcal{S}_i = \{10\}$ 以及 $c_j \le C_i$ 和 $m_j \le M_i$ 的条件下，将任务 τ_j 分配给节点 N_i。节点 N_i 与 N_j 之间的每条通信链路 $e_{ij} \in E$ 都具有带宽 $b_{ij} \ge \delta$，其中 δ 为带宽阈值。分配函数 $\mathcal{A}$ 下任务 τ_j 的端到端时延定义为 $l_j(A) \leftarrow l_j^{net}(A) + l_j^{com}(A)$，其中 $l_j^{net}(A)$ 为网络时延，$l_j^{com}(A)$ 为计算时延。系统的总体时延为 $\mathcal{L}(A) = \frac{1}{m} \sum_{j=1}^{m} l_j(A)$。

DCCS 中的故障通过故障场景 ω 表示，其中 $F_\omega(t) \subseteq \mathcal{N}$ 表示在时间 t 变得不可用的节点集合。当节点 $N_i \in F_\omega(t)$ 时，其运行状态转换为 $\mathcal{S}_i(t) = \text{down}$，从而中断当前分配给它的所有任务（即满足 $A(t, \tau_j) = N_i$ 的任务）。自愈过程启动后，该节点进入恢复状态，即 $\mathcal{S}_i(t^+) = \text{recovering}$，系统随后执行修复流程，以检测故障、隔离受影响的任务，并将其重新分配给运行状态和资源允许执行这些任务的存活节点。由此得到更新后的分配，定义为 $A_\omega(t^+) = \text{Heal}(A(t) \setminus F_\omega(t))$。对于场景 ω 下的每项任务 τ_j，我们定义完成指示变量 $I_j(A, \omega) = \{0 \text{ or } 1\}$：若 τ_j 在 $A_\omega(t^+)$ 下成功完成，则取值为 1，否则为 0；并据此计算分配策略的韧性，定义如下：

$$\mathcal{R}(A) = \mathbb{E}_\omega \left[\frac{1}{m} \sum_{j=1}^{m} I_j(A, \omega) \right], \tag{1}$$

它量化了即使发生节点故障、状态转换（宕机 $\to$ 恢复）以及相关的自愈操作时，系统仍能成功完成的任务预期比例。资源利用率通过综合考虑 CPU 和内存使用率来定义，如下：

$$\mathcal{U}(A) = \alpha \frac{\sum_{i \in \mathcal{N}} \text{cpu_load}(N_i)}{\sum_{i \in \mathcal{N}} \mathcal{C}_i} + (1-\alpha) \frac{\sum_{i \in \mathcal{N}} \text{mem_load}(N_i)}{\sum_{i \in \mathcal{N}} \mathcal{M}_i}, \tag{2}$$

其中，$0 < \alpha \le 1$ 控制计算利用率和内存利用率的相对重要性。DCCS 控制器旨在确定一种分配策略，以联合优化延迟、资源利用率和韧性。由此形成的多目标优化问题可表述为

$$\min_A \ \mathcal{L}(A), \qquad \max_A \ \mathcal{U}(A), \qquad \max_A \ \mathcal{R}(A), \tag{3a}$$

受以下条件限制：

$$\sum_{\tau_j \in \mathcal{T} : A(t, \tau_j) = N_i} c_j \le \mathcal{C}_i, \qquad \forall N_i \in \mathcal{N}, \tag{3b}$$

$$\sum_{\tau_j \in \mathcal{T} : A(t, \tau_j) = N_i} m_j \le \mathcal{M}_i, \qquad \forall N_i \in \mathcal{N}, \tag{3c}$$

$$\mathcal{S}_{A(t,\tau_j)} = \text{available}, \qquad \forall \tau_j \in \mathcal{T}, \tag{3d}$$

$$\mathcal{V}_{A(t,\tau_j)} \in \{\text{low}, \text{medium}\} \quad \text{for critical tasks } \tau_j, \tag{3e}$$

$$b_{ij} \ge \delta \qquad \forall e_{ij} \in E, \tag{3f}$$

$$A(t, \tau_j) \in \mathcal{N}, \qquad \forall \tau_j \in \mathcal{T}. \tag{3g}$$

IV. RECIST 框架

如动机部分所述，ReCiSt 框架通过自愈流水线运行，该流水线由遏制层、诊断层、元认知层和知识层构成，如图 2 所示。

A. 遏制层

该层作为 ReCiSt 框架的初始防线和即时响应机制，旨在确保服务不中断（图 2 左上角和算法 1）。系统必须准确判断何时偏离健康的运行状态。由于设备具有异构性、日志格式各异，且连续体中的感知能力不一致，ReCiSt 使用由大语言模型驱动的监控智能体来持续监控系统状态。每个智能体都会维护其 k 邻域内所有节点的最新信息，包括运行状态、脆弱性级别和活动任务集。故障识别始于 α_t 定期向每个节点广播轻量级控制信号。每个节点必须在预定义时间间隔内确认收到该探测，并报告其即时状态（t）以及概述当前运行负载的最简心跳向量。如果节点未能在 Δt 内响应，即在探测窗口期间未收到来自 N_i 的确认，智能体 α_t 便会将其判定为偏离预期行为的异常，并触发相应操作。需要注意的是，在此阶段，系统尚不清楚故障的确切原因或类型。检测到的偏离仅表明该节点已偏离正常运行条件。遏制层会将所有此类节点初步标记为故障节点，并将其纳入故障集 F（t）。$= \{ N_i \mid$ 遏制层在时间 $t \}$.标记了 N_i

当检测到节点 N_i 的状态变为 $\mathcal{S}_i(t) =$00（对于任意值的 $\mathcal{V}_i$）时，节点的内部神经反射系统会向其最近的 k_N 个候选节点广播局部遏制请求，表示为集合 $k_N = \{N_j \mid N_j$，并接收来自智能体}的反射请求。每个节点 $N_j \in k_N$ 都会通过报告其 $\mathcal{C}_j$ 和 $\mathcal{M}_j$，以及 $\mathcal{S}_j(t) = \{11\}$来作出响应。遏制层请求这些响应，以识别适合重新分配任务集 $\mathcal{T}_i$ 的邻居；该任务集先前运行于 N_i上。当所有作出响应的邻居都针对每个 N_j报告$\mathcal{S}_j(t) = 11$ 时，即满足适用性条件，随后节点 N_i 转变为可用的协调状态 $\mathcal{S}_i(t^+) = 10$。如果任何单个邻居的计算能力和内存（记为 $\mathcal{C}_j$）

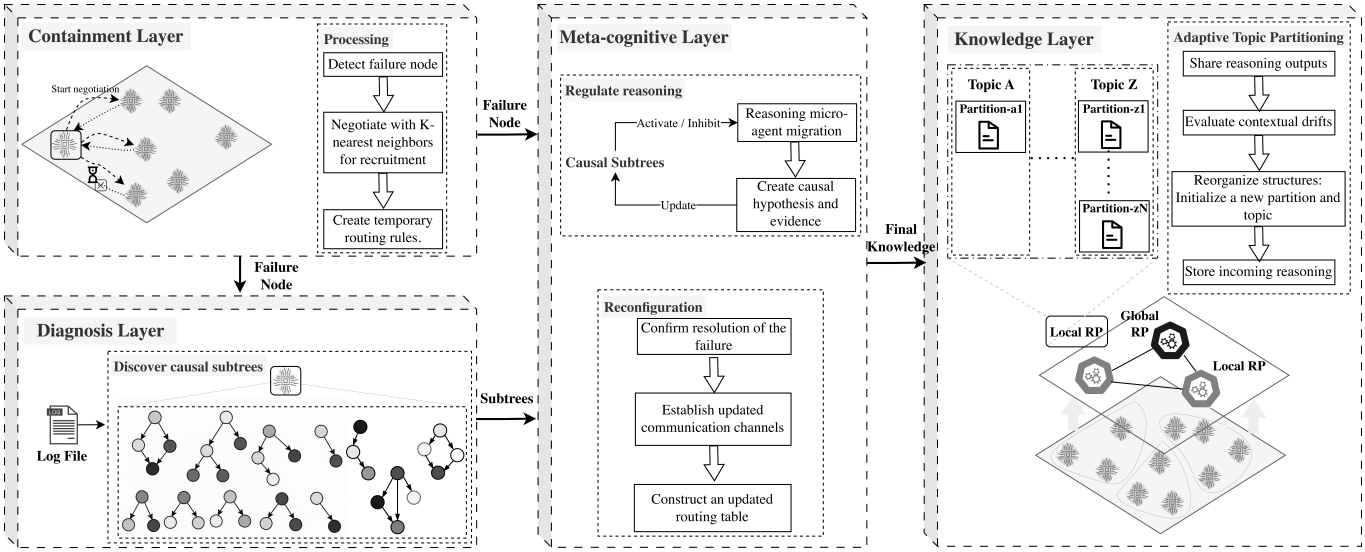


图 2：展示不同智能体层（遏制层、诊断层、元认知层、知识层）的详细 ReCiSt 框架

算法 1： ReCiSt 框架的遏制层

输入： N_i ▷ N_i 是节点数量
输出： $F(t)$ ▷ 故障集
$p = Null$ ▷ p已接受的节点，初始为空

智能体提示

- 调用工具 → 检查邻居的状态 $\mathcal{N}_i^{(k)}$:
对于每个 $N_i \in \mathcal{N}_i^{(k)}$ **执行**
  如果 N_i (t) $\geq \Delta t$ **则**
    $\mathcal{S}_i(t) = 00$
    $F(t) \leftarrow F(t) \cup N_i$
- 决策无故障邻域集合 k_N:
$k_N = \mathcal{N}_{(k)_i} - F(t)$ ▷ k_N是 k近邻
- 调用邻居代理 → 协商
对每个 $N_j \in k_N$ **执行**

  邻居代理提示

  决策并返回 11_ ，01，or 10 $\mathcal{S}_j(t) \leftarrow \mathcal{C}_j \& \mathcal{M}_j$ ▷ 计算能力和 N_j 的内存

- 解析响应:
如果 $\mathcal{S}_j(t) == 11$ **则**
  $p \leftarrow p \cup N_j$
- 分析 N_i 任务和依赖关系。
- 决策将 N_i 任务分配给 p

RETURN $F(t)$

以及 $\mathcal{M}_j$，不足以执行完整的任务集 $\mathcal{T}_i$，这意味着 $\mathcal{C}_j < \mathcal{C}_i$ 的 $\mathcal{T}_i$ 或 $\tau_x \forall x \leq |\mathcal{T}_i|$以及 $\mathcal{M}_j < \mathcal{M}_i$ 的 $\mathcal{T}_i$ 或 $\tau_y \forall y \leq |\mathcal{T}_i|$，随后，遏制层从 k_N 候选节点中识别出一组 p 节点，以协作分配并执行 $\mathcal{T}_i$ 或 τ_x的工作负载。通过这一过程，它们协作建立动态插接结构，从而能够形成临时路由规则。主代理分析任务依赖关系，并在 N_j 之间重新分配N_i 计算工作负载，以降低级联故障的风险。识别出的故障节点将传输至后续层，以进行进一步分析和修复。

B. 诊断层

遏制层发出异常偏离信号后，系统便进入诊断层（图 2 左下方），这对应于生物伤口愈合中的炎症阶段。在人体生理学中，炎症是最初的分析性响应：白细胞迁移至伤口部位，识别损伤性质，并在组织修复开始前评估其严重程度。类似地，在 ReCiSt 框架中，诊断层会确定遏制层所识别的偏离节点的根本原因、范围和结构特征。在此阶段，系统会进行结构化因果分析，从而生成精确且机器可解释的故障表征。

对于每个 $N_i \in F(t)$，诊断层会检索在时间区间 $[t - \Delta_d, t]$内生成的相应系统日志、网络日志、自定义日志等，其中 Δ_d 为诊断窗口大小。设 $\mathcal{L}_i$ 表示节点 N_i的聚合日志结构：$\{\mathcal{L}_i = \mathcal{L}_i^{sys} \cup \mathcal{L}_i^{net} \cup \mathcal{L}_i^{cust} \cup ...\}$，其中每个分量均反映不同的运行维度。

诊断层将原始日志条目转换为一组结构化的可观测实体。每个实体对应与故障相关的事件、性能指标、内部状态转换或资源指标。这些实体被实例化为诊断变量集中的节点$\mathcal{X}_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$，该变量集构成了

算法 2： ReCiSt 框架的诊断层

输入： $F(t)$

对于每个 $N_i \in F(t)$ **执行**

/*智能体利用 $\mathcal{L}_i$ 的 N_i*/

智能体提示

$\mathcal{X}_i \leftarrow Null$ ▷ 节点提取

对于每个 $l \in \mathcal{L}_i$ **执行**

提取 x_{ij} ▷ x_{ij} 是事件，指标，状态，组件

添加每个 $x_{ij} to \mathcal{X}_i$

$\mathcal{E}_i \leftarrow Null$ ▷因果分解

对于每个 $(x_{ia}, x_{ib}) \in \mathcal{X}_i$ **执行**

如果 $\Phi(x_{ia}, x_{ib}) = 1$ **则**

$\mathcal{E}_i$.添加$(x_{ia} \rightarrow x_{ib})$

$G_i^{diag} \leftarrow Null$ ▷ 返回结构化子树

$G_i^{diag} = (\mathcal{X}_i, \mathcal{E}_i)$

调用算法 3

故障的基于图的表示基础。每个 $x_{ij} \in \mathcal{X}_i$ 都是从 $\mathcal{L}_i$中提取的符号或数值描述符，例如 CPU 异常、内存峰值、链路退化指标、任务停滞或错误代码。接下来，该层推断 $\mathcal{X}_i$中各元素之间的因果依赖。这些依赖关系定义了一个内部变化如何导致另一个变化，从而形成刻画故障或异常操作的因果结构。对于每个有序对 $(x_{ia}, x_{ib}) \forall a, b \in m$，系统会评估两者之间是否存在因果关系。因果关系编码如下：

$$x_{ia} \rightarrow x_{ib} \quad \text{iff} \quad \Phi(x_{ia}, x_{ib}) = 1, \tag{4}$$

其中，$\Phi(\cdot)$ 是由语言模型驱动的关系识别函数，用于整合时间先后关系、日志语义和学习得到的因果先验。针对 N_i 识别出的所有这些因果关系构成一个有向边集：

$$\mathcal{E}_i = \{ (x_{ia}, x_{ib}) \mid \Phi(x_{ia}, x_{ib}) = 1 \}. \tag{5}$$

因此，数对 $(\mathcal{X}_i, \mathcal{E}_i)$ 构成一个有向图 G_i^{diag}：

$$G_i^{diag} = (\mathcal{X}_i, \mathcal{E}_i), \tag{6}$$

该图表示观测到的故障背后的细粒度因果结构。

为增强鲁棒性，诊断层采用由多个并行推理子树组成的集成模型。每棵子树对应一条经过训练的专用因果推理路径，用于捕捉特定类型的依赖关系，例如资源过载、网络不稳定、任务级争用、热异常或固件事件。设该集成模型表示为 $\Psi_i = \{\psi_i^{(1)}, \psi_i^{(2)}, \ldots, \psi_i^{(q)}\}$，其中 $\psi_i^{(k)}$ 是应用于节点 N_i的第 k 棵子树。每个 $\psi_i^{(k)}$ 从 G_i^{diag} 中提取一个局部因果子图 $T_i^{(k)}$：

$$T_i^{(k)} = (\mathcal{X}_i^{(k)}, \mathcal{E}_i^{(k)}), \quad \text{with } \mathcal{X}_i^{(k)} \subseteq \mathcal{X}_i,\ \mathcal{E}_i^{(k)} \subseteq \mathcal{E}_i. \tag{7}$$

所有此类子树的并集构成了整合后的诊断结构，

$$\bar{G}_i^{diag} = \bigcup_{k=1}^{q} T_i^{(k)}, \tag{8}$$

该图编码了对故障的多层级、多因果解释。这个汇总图存储在节点的诊断内存中，随后传输到元认知层，用于推理路径重构和微型智能体生成。至此，每个故障节点 N_i 或链路都不再仅仅被视为孤立的故障，而是被表示为一个结构化因果系统，用以解释感知到的偏差为何会发生。

C. 元认知层

经过诊断阶段后，系统进入元认知层（图 2 中部），对应于生物伤口愈合中的增殖阶段。在人体组织中，增殖涉及成纤维细胞活动、细胞外基质沉积和血管生成，这些过程共同恢复伤口部位周围的结构支撑并重建功能连接。在 ReCiSt 框架中，元认知层在认知层面发挥类似作用：它重组并扩展从诊断层得出的因果结构，管理推理微型智能体的生成与调控，并完善解释路径，使系统能够从原始故障描述推进到一组连贯且具有实际操作价值的假设。本层的流程如算法 3 所示。

本层将式（6）细化为一组可行的解释和纠正路径。为此，与 N_i 关联的主智能体会生成一群推理微型智能体，如式（9）所示，

$$\mathcal{A}_i^{micro} = \{a_i^{(1)}, a_i^{(2)}, \ldots, a_i^{(r)}\}, \tag{9}$$

其中，r 根据因果图的复杂度自适应确定。每个微型代理 $a_i^{(k)}$ 在 $\bar{G}_i^{diag}$ 内使用深度优先搜索（DFS）遍历路径，以构建故障的因果解释。路径定义为诊断变量的有序序列：

$$p^{(k)} = (x_{i\ell_1}, x_{i\ell_2}, \ldots, x_{i\ell_\eta}), \quad \text{with } (x_{i\ell_j}, x_{i\ell_{j+1}}) \in \mathcal{E}_i, \tag{10}$$

其中，η 是微型代理探索的深度。对于每条路径 $p^{(k)}$，微型代理都会生成一个结构化假设

$$H_i^{(k)} = \Lambda(p^{(k)}, \mathcal{L}_i), \tag{11}$$

其中，$\Lambda(\cdot)$ 是一个语言模型驱动算子，它将路径结构与从日志 $\mathcal{L}_i$中提取的支持性证据相结合。每个假设 $H_i^{(k)}$ 都会被赋予一个通过式（12）计算的元认知评估分数

$$\Gamma(H_i^{(k)}) = w_1 C_{coh}(H_i^{(k)}) + w_2 C_{safe}(H_i^{(k)}) + w_3 C_{util}(H_i^{(k)}), \tag{12}$$

其中，C_{coh} 衡量因果连贯性，C_{safe} 量化推断解决方案的安全性，C_{util} 反映操作可行性，而（w_1，w_2，w_3）为归一化权重。

元认知层的核心自适应机制是对微型智能体增殖的调节，这一调节由以下内容引导：

算法 3： ReCiSt 框架的元认知层——work

输入： G_i^{diag} ， N_i

输出： $Updatedknowledge$ ▷ (故障主题、原因和解决方案)

- 调用工具 →深度优先遍历:

$P^{(k)} \leftarrow$ 深度优先遍历(G_i^{diag}) ▷ 创建所有来自子树的可能路径

对每个 $p^{(k)} \in P^{(k)}$ **执行**

微型智能体提示

- 调用微型智能体 $\mathcal{A}_i^{micro} \rightarrow$ 生成因果候选项 $a_i^{(1)}$， $a_i^{(2)}$， ....., $a_i^{(r)}$

$H_i^{(k)} \leftarrow a_i^{(k)}$.生成($H_i^{(k)} = \Lambda(p^{(k)}, \mathcal{L}_i)$，)

评估器智能体提示

分数 $\Gamma(H_i^{(k)}) \leftarrow$ 解析($w_1 C_{coh}(H_i^{(k)}) + w_2 C_{safe}(H_i^{(k)}) + w_3 C_{util}(H_i^{(k)},)$)

解析并决策($\Gamma(H_i^{(k)})$):

if $\Gamma(H_i^{(k)}) < \theta_{pro}$ ▷分数有害 **则**

调用 $\mathcal{A}_i^{micro}$

如果分数可接受 **则**

存储候选项(status="支持")

调用 微型代理($p+1$)

如果分数为拒绝 **那么**

调用 微型代理($p+1$)

if $\Gamma(H_i^{(k)}) \geq \theta_{inh}$ ▷分数为最佳 **那么**

$\mathcal{K}_i^{meta} \leftarrow \mathcal{K}_i^{meta} \cup H_i^* =$ 最大化$_k \Gamma(H_i^{(k)})$

$F(t) \leftarrow F(t) \cap N_i$

返回 $\mathcal{K}^{meta}{}_i$

$\Gamma(H_i^{(k)})$产生的反馈。类似于基质完整性较低时成纤维细胞会更快增殖，当当前假设的置信度较低或安全性较差时，系统会增加微型智能体的数量。形式上，当 $\Gamma(H_i^{(k)}) < \theta_{pro}$ 时，就会触发增殖，其中 θ_{pro} 是增殖阈值。触发后，集合 $\mathcal{A}_i^{micro}$ 会扩展，并通过在 $\bar{G}_i^{diag}$ 中插入辅助节点或改变遍历方向来生成新的探索路径。这与生物学中的血管生成相似：新的血管会延伸至受损区域以恢复连通性；在这里，则是添加新的推理边，以增强认知探索的多样性和深度。

相反，当假设具有较高的置信度和安全性时，系统会通过强制执行条件 $\Gamma(H_i^{(k)}) \geq \theta_{inh}$来抑制进一步增殖，从而遏制微型智能体的继续生成。激活与抑制之间这种由反馈驱动的平衡，能够稳定推理生态系统，并防止计算开销不必要地增加。随着微型智能体不断积累证据并完善假设，因果图 $\bar{G}_i^{diag}$ 会发生结构性重组。令

算法4： ReCiSt 框架的知识层

输入： $\mathcal{K}_i^{meta}$

输出： $\mathcal{Z}^{RP} = \{Z_1, Z_2, Z_j, ..., Z_u\}$ ▷ 知识存储在内存中

$\mathbf{e}_{topic} = \phi_{topic}(z_i) \ \forall \ \mathcal{K}_i^{meta}$ ▷ 计算嵌入

使用 "text-embedding-3-small" 模型处理新主题

$M_T(z_i, Z_j) =$语义文本相似度($\mathbf{e}^{topic}, \mathbf{e}_{Z_j}$) ▷ 计算相似度

通过语义文本相似度

如果 最大值$_j$ $M_T(z_i, Z_j) < \theta_{topic}$ **则**

$\mathcal{Z}^{RP} \leftarrow \mathcal{Z}^{RP} \cup \{Z_{u+1} = \{P_{u+1}^{(1)}\}$

else

$\mathbf{e}_{reason} = \phi_{reason}(H_i^*)$

$M_R =$语义文本相似度($\mathbf{e}_{reason}$,$\mathbf{e}_{P_{j^*}^{(k)}}$)

如果 最大值$_k$ $M_R < \theta_{reason}$ **then**

$Z_{j^*} \cup P^{(m_{j^*}+1)\,j^*}$

返回 $\mathcal{Z}^{RP}$

$\Delta\mathcal{X}_i^{(k)}$ 和 $\Delta\mathcal{E}_i^{(k)}$ 表示由假设 $H_i^{(k)}$ 引起的修改。更新后的认知结构如式 13 所示，

$$\bar{G}_i^{meta} = \left(\mathcal{X}_i \cup \bigcup_k \Delta\mathcal{X}_i^{(k)}, \mathcal{E}_i \cup \bigcup_k \Delta\mathcal{E}_i^{(k)} \right). \tag{13}$$

公式13与能够强化伤口区域的新暂时性组织的生物形成直接类似。当$\bar{G}_i^{meta}$达到稳定配置后，元认知层会识别出一个最优假设:

$$H_i^* = \arg\max_k \Gamma(H_i^{(k)}), \tag{14}$$

该假设作为节点 N_i的最终因果解释和初步纠正策略。随后，该输出会传播至知识层；知识层对应于伤口愈合中的重塑阶段，负责全局系统重新对齐、路由更新，以及恢复节点的重新整合。

D. 知识层

元认知层为故障节点 N_i选定最优假设 H_i^* 后，系统进入知识层（图 2右侧），该层映射了生物伤口愈合中的重塑阶段。在生理学中，重塑会强化细胞外基质、重组组织纤维，并将修复区域重新整合到更大的功能结构中。类似地，知识层会整合、重构并传播恢复过程中产生的经过细化的因果知识和纠正知识，从而支持DCCS内稳定的长期适应。该层的操作流程详见算法4。

知识层围绕一组本地和全局资源池构建，这些资源池充当分布式智能体的自适应协调与存储节点。每个资源池都维护一个结构化知识库，由面向主题的片段组成。一个主题对应一个故障类别，其关联表征包括因果解释、假设，

以及从元认知层得出的纠正策略。对于给定的资源池，设存储主题的集合记为$\mathcal{Z}^{RP} = \{Z_1, Z_2, \ldots, Z_u\}$，其中每个主题 Z_j包含多个分区，每个分区代表一种独特或语义上不同的推理结果： $Z_j = \{P_{(1)j}, P_j^{(2)}, \ldots, P_{(j}^{m_j}\}$。

当新的知识包 $\mathcal{K}_i^{meta}$从元认知层到达时，首先提取其主题描述符，并将其编码为嵌入表示$\mathbf{e}_{topic} = \phi_{topic}(z_i)$，其中 z_i 是与 $\mathcal{K}_i^{meta}$关联的文本或符号主题标签，而 $\phi_{topic}(\cdot)$ 是用于主题编码的嵌入模型。每个存储主题 Z_j 也有一个代表性嵌入$\mathbf{e}^{Z_j}$。新主题与现有主题之间的语义接近度通过以下方式计算

$$M_T(z_i, Z_j) = \text{STS}\big(\mathbf{e}_{topic}, \mathbf{e}_{Z_j}\big), \tag{15}$$

其中 STS 表示语义文本相似度运算符。如果所有存储主题中的最大相似度满足公式 16

$$\max_j M_T(z_i, Z_j) < \theta_{topic}, \tag{16}$$

其中 θ_{topic}为主题匹配阈值，则创建一个新主题 Z_{u+1}并将其添加到 $\mathcal{Z}^{RP} \leftarrow \mathcal{Z}^{RP} \cup \{Z_{u+1} = \{P_{u+1}^{(1)}\}\}$。否则，将新的知识实例关联到最匹配的主题 Z_{j^*}，其中 $j^* = \arg\max_j M_T(z_i, Z_j)$。

在选定的主题 Z_{j^*}中，系统接下来会评估与 H_i^*相关的原因或解释是否与现有分区匹配。其嵌入表示为$\mathbf{e}_{reason} = \phi_{reason}(H_i^*)$，与每个已存储原因嵌入$\mathbf{e}_{P_{(k)j^*}}$的相似度计算为 M_R =语义文本相似度$\big(\mathbf{e}_{reason}, \mathbf{e}_{P_{j^*}^{(k)}}\big)$。如果最大相似度满足 $\max_k M_R < \theta_{reason}$，知识层就会在主题 Z_{j^*}内创建一个新分区 $P_{(m_{j^*}+1)j^*}$。否则，该知识实例会强化与其最匹配的现有分区，因此不会发生结构性增长。

随着主题和分区不断演变，该层会执行自适应重组，以确保连贯性并减少冗余。当某个主题中的两个分区表现出较高的相似度，即

$$\text{STS}\left(\mathbf{e}_{P_{j^*}^{(k_1)}}, \mathbf{e}_{P_{j^*}^{(k_2)}}\right) \geq \theta_{merge}, \tag{17}$$

将它们合并为统一表示。相反，如果某个分区表现出显著的内部语义漂移（通过偏差度量 DIV(·)量化），且该值超过分歧阈值，则使用式 18 将该分区拆分为多个子分区。

$$\text{DIV}\big(P_{j^*}^{(k)}\big) > \theta_{split}. \tag{18}$$

为维持全局系统一致性，每个本地 RP 会定期通过以下方式将其主题和分区与全局 RPs 同步

$$\mathcal{Z}_{global}^{RP} \leftarrow \text{MERGE}\big(\mathcal{Z}_{global}^{RP}, \mathcal{Z}_{local}^{RP}\big), \tag{19}$$

冲突通过基于相似度的合并规则和版本控制元数据来解决。通过在主题和分区层面进行重组，这一层逐步强化了 ReCiSt 框架。这一过程类似于伤口愈合的重塑期：组织在此期间重新组织、得到强化，并融入周围结构。由此形成的知识拓扑支持分布式智能体之间的可扩展协调，促进稳健的上下文感知决策，并维持 DCCS 的长期韧性。

E. ReCiSt 的计算效率

尽管 ReCiSt 框架包含运行时间取决于模型规模、推理硬件和系统负载的大语言模型驱动及智能体驱动的操作，但仍可通过渐近复杂度分析其算法结构。每次大语言模型调用或提示驱动的推理步骤都视为一次预言机操作，其均摊成本为常数 $O(1)$，因为其延迟不会随 DCCS 的规模而变化。在此假设下，遏制层（算法 1）执行两项主要操作：在大小为 d_i的邻接集 $\mathcal{N}_i$ 上进行邻居状态探测，耗时为 $O(d_i)$；以及计算 k个最近邻居 $\mathcal{N}_i^{(k)}$，由于需要进行距离评估并使用堆进行 top-k 选择，耗时为 $O(d_i \log k)$。因此，遏制层对每个受监控节点的耗时为 $O(d_i \log k)$。诊断层（算法 2）会解析每个故障节点 N_i 的大小为 L_i 的日志文件 $\mathcal{L}_i$，并构建包含 m_i个提取变量的诊断图，因此成对因果评估的成本为 $O(m_i^2)$；其总复杂度为 $O(L_i + m_i^2)$。元认知层（算法 3）会生成 r_i 个推理微型智能体，并探索从子树 Ψ_i中提取的 p_i 条诊断路径。由于大语言模型推理步骤被视为预言机操作，复杂度可简化为 $O(p_i + r_i)$。知识层（算法4）会将每个主题嵌入与 $\mathcal{Z}^{RP}$ 中的 u个存储主题进行比较，然后针对所选主题 Z_j的 m_j 个分区评估推理嵌入，每次知识更新的耗时为 $O(u + m_j)$。对于故障节点 N_i，总体自愈成本为$O(d_i \log k + L_i + m_i^2 + p_i + r_i + u + m_j)$，这反映了 ReCiSt 流水线的结构复杂度，不受大语言模型推理开销影响。

V. 性能评估

本节介绍所提出的 **ReCiSt** 框架的实验评估。首先介绍实验设置和性能指标，随后讨论每个数据集的数值结果。

A. 设置

ReCiSt 框架在 Google Colab 中实现，使用 Intel(R) Xeon(R) CPU 来表示云计算环境。针对 ReCiSt 评估的大语言模型部署在云端，并通过 OpenAI API 访问。实验设置包括分布在地理位置不同的 k 计算节点之间的 n 智能体。我们假设每个智能体会 i 调用其 K近邻，随后使用基板管理控制器，从未在指定时间范围内响应的节点开始进行日志采集。文中还将介绍计算机、网络和通信配置以及数据集的详细信息。

1) 模型： 实验评估中使用的所有智能体均采用 LangChain 1.0.8 [31]实现。内存模块是使用 Pydantic [32]构建的自定义存储库类型。为生成嵌入，我们使用了 text-embedding-3-small 模型，这是由 OpenAI [33]开发的小型嵌入模型。为估计语义相似度，我们采用了 all-MiniLM-L6-v2，这是一种句子转换器模型 [34]，专为语义文本相似度任务设计，可计算文本输入之间具有意义的相似度分数。在评估中，我们采用适用于分布式计算连续体计算资源的推理模型。这一设置要求模型具备高级的多步推理能力，并能够在异构且受限的计算环境中高效运行。为满足这些要求，我们依赖多种 OpenAI 模型 [33]，包括 o4-mini-2025-04-16，这是 o 系列中最新的小型模型，具有较强的推理能力和较低的延迟。gpt-5-mini-2025-08-07 模型具有快速性能和较强的推理能力。gpt-5-nano-2025-08-07 是速度最快的选项。gpt-5.1-2025-11-13 模型则用于智能体任务。

2）性能指标：ReCiSt 框架的自愈过程主要依赖其智能体（分布于四个层级）以及准确、及时的决策过程。因此，我们的评估指标围绕智能体操作的有效性展开 [35]。具体而言，包括自愈所需时间、为避免不确定性而进行分析的深度、自愈过程中调用的微型智能体数量，以及故障诊断、协商、解决方案发现和解决方案存储所产生的计算开销。我们还进一步分析了故障条件下的智能体决策质量，包括成功、有益和有害响应的比率，以及智能体推理所导致的结构复杂度。以下定义这些指标。

自愈耗时：该指标衡量从检测到故障，到完成 ReCiSt 流水线全部四个阶段并成功恢复所经过的时间。

智能体 CPU 消耗：该指标量化智能体在执行故障诊断、与其他智能体协商以及自愈过程中的 CPU 使用率。

子树深度复杂度：此指标显示智能体开始分析故障时的深度。

微型智能体调用次数：此指标显示微型代理调用的次数，反映每个恢复过程所需的协调和通信开销。

决策质量：该指标以已接受、已拒绝、有害或最佳类别的形式，展示每个微型智能体的决策情况，具体定义如下：接受率：提供建设性且可执行指导的支持性响应数量与响应总数之比。有害率：可能降低 CPU 利用率、延迟或内存消耗等关键绩效指标，并对系统行为产生负面影响的响应数量与响应总数之比。拒绝率：缺乏充分证据或理由的响应数量与响应总数之比。最佳率：展现出充分证据支持、清晰因果推理并符合操作安全原则的响应数量与响应总数之比

。推理深度率（*RDR*）：系统级微代理调用次数与已实例化微型智能体总数之比。

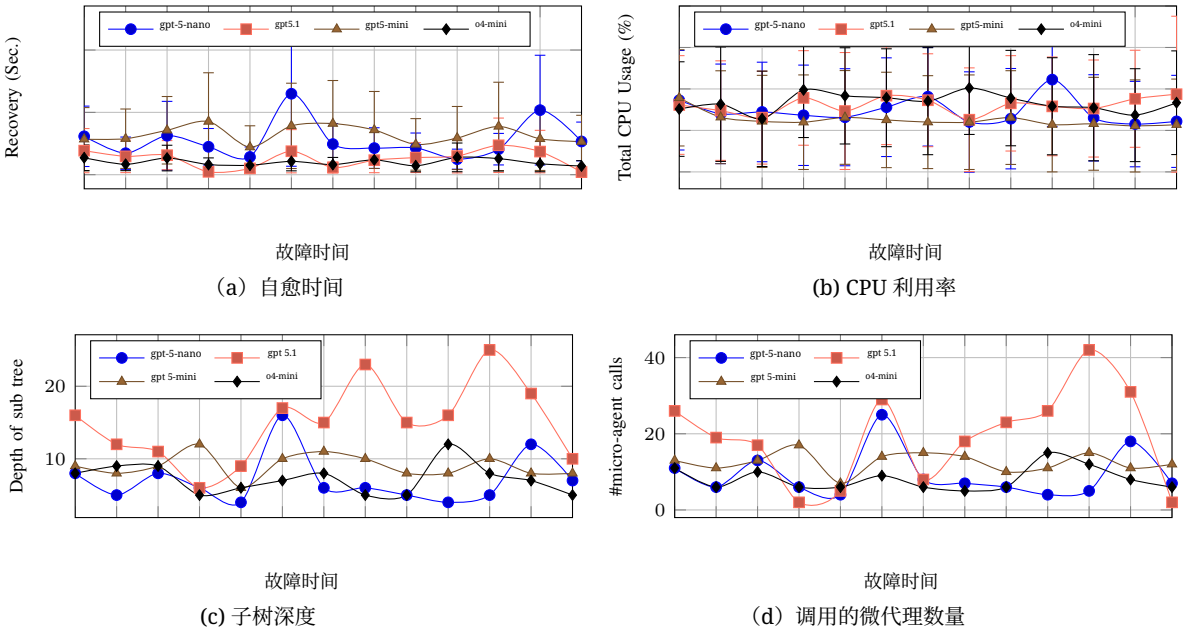
B. 结果与分析

在本节中，我们评估多个数据集上的性能并分析数值结果，即来自 IEEE DataPort 的云无状态数据集[36] 和 Loghub [37], [38]。Loghub 集合包含来自 ZooKeeper、Hadoop、OpenSSH 和 Blue Gene/L 超级计算机等系统的不同类别日志。后续各节将按数据集分别进行性能评估和讨论。最后，我们将总结全部五类数据集的决策质量指标。

1) 云无状态数据集：云无状态系统数据集包含从三台基于云的 Linux 虚拟机收集的测量数据，采样间隔为 5 秒。这些虚拟机配置为 1 个 vCPU、1 GB 内存和 10 GB 磁盘，均在动态变化的工作负载下运行。每条记录都包含一个时间戳，以及通过 Prometheus 收集的资源利用率和性能指标，包括 CPU 使用率和内存使用率（利用率百分比）、入站带宽和出站带宽（吞吐量，单位为 GB/s 或 MB/s）、每秒请求数（请求数/秒）、响应时间（延迟，单位为秒或毫秒）以及状态（系统状态，0 表示健康，1 表示不健康）。这些变量共同构成了设备负载、网络活动、服务吞吐量和运行响应能力的全面时间表示。故障表现为响应时间升高和带宽消耗不规律，这是由网络相关瓶颈引起的，并伴随拥塞和延迟峰值。这些不健康时段反映了数据流波动和工作负载变化导致的性能限制，部分情况下还出现了 CPU 负载上升。

图 3a 展示了沿数据集时间线，四种模型在多个故障实例中的自愈耗时。值得注意的是，所有模型都成功实现了自愈，但耗时各不相同。我们的分析发现，o4- mini 模型的自恢复时间最短，通常保持在 300 秒（≈5 分钟）以内。同样，gpt-5.1 的恢复时间也较短，通常低于 400 秒（≈6–7 分钟），其中一些实例的恢复速度非常快，仅需 43 秒。相比之下，gpt-5-nano 的恢复时间从≈250秒（≈4 分钟）延长至超过 1,200 秒（≈20 分钟）。gpt-5- mini 模型的恢复时间较长，经常超过 700–800 秒（≈12–13 分钟），表明其自愈速度较慢。图 3b 描述了数据集中所述故障发生时，不同模型在多个故障实例中的 CPU 利用率。我们注意到，在运行 ReCiSt 代理时，所有模型的 CPU 利用率总体保持稳定。从数值上看，gpt- 5-mini 保持在约 ≈13%，o4-mini 运行在类似的≈13% 范围内，gpt-5.1 平均约为 ≈15%，而 gpt-5-nano 接近 ≈14 %，表明计算开销得到了有效控制。

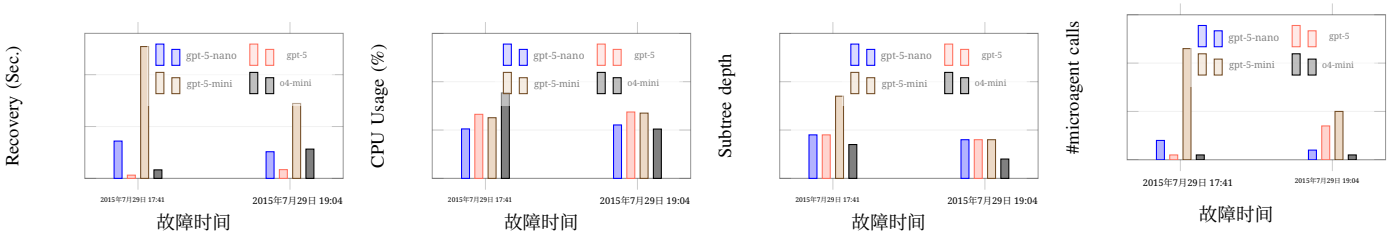
图 3c 和图 3d 展示了深度优先搜索中的路径数量，以及用于自愈的微型智能体数量。从图 3c 可以看出，gpt-5.1 的结构复杂度最高，



（a）自愈时间 (b) CPU 利用率

(c) 子树深度 （d）调用的微代理数量

图 3：使用所提出的 ReCiSt 框架和四种大语言模型对云无状态数据集进行性能评估



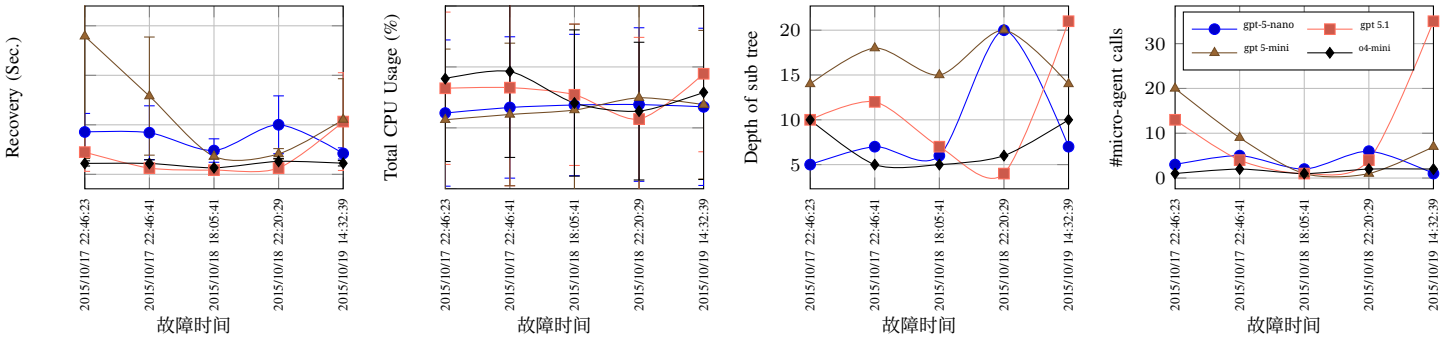
(a) 自愈时间 (b) CPU 利用率 (c) 子树深度 (d) 微代理调用

图4：所提出的自愈ReCiSt框架对Zookeper数据集的性能评估

它始终生成更大的子树，包含 6 到 25 条路径。尽管如此，gpt-5.1 和 o4-mini 都无需遍历整个搜索空间便解决了故障。这一模式凸显了它们在初始推理结构较为庞大的情况下，仍具备强大的选择性路径消除和早期解检测能力。相比之下，gpt-5-nano 探索的子树包含 ≈4 到 16 条路径，微代理调用次数为 4 到 25 次。图 3d 清楚地显示，gpt-5-mini 的结构复杂度适中，生成了 6 到 12 条路径，微代理调用次数为 7 到 15 次；而 o4-mini 则在推理深度与计算开销之间取得平衡，生成了 5 到 12 条路径，调用次数为 5 到 12 次。

*2) Zookeper*数据集：ZooKeeper 是一种集中式服务，支持配置管理、命名、分布式同步和组服务。日志数据采集自一套包含32台机器的实验室部署环境。在该环境中，观察到的故障表现为连接中断或断开。图4a展示了四种模型在 ZooKeeper 数据集时间线上两个故障实例的自愈时间。结果显示，gpt-5.1 的自恢复时间最短，通常为30至80秒。此外，o4-mini 的恢复时间略长于 gpt-5.1，即80至280秒。相比之下，gpt-5-mini 的恢复时间较长，超过1200秒，体现出其恢复行为更深入但速度较慢。图4b展示了 ZooKeeper 数据集中所报告的故障发生时刻，两个故障实例期间不同模型的 CPU 利用率。可以观察到，所有模型的 CPU 利用率均保持稳定。例如，gpt-5-mini 的结果为 ≈13%，o4-mini 的运行范围接近14%，gpt-5.1 的平均值为 ≈13%，而 gpt-5-nano 接近11%，表明其计算开销可控。根据图4c，gpt-5.1 和 gpt-5-nano 的结构复杂度相近，子树包含8至9条路径，而 o4-mini 的结构更紧凑。图4d进一步显示，gpt-5-mini 的结构复杂度较高，生成8至17条路径以及 ≈10-23 次微代理调用；而 o4-mini 的推理深度和计算开销较低，包含4至7条路径，调用约1次。gpt-5-nano 的开销也较低，仅需2至4次微代理调用，体现了其在 Zookeper 数据集上的高效推理能力。

3）Hadoop 数据集：Hadoop 是一种大数据处理框架。在 Loghub 中，Hadoop 日志数据来自一个在故障注入场景下运行的五节点集群，场景包括机器宕机、网络断连和磁盘已满事件，



（a）自愈时间　（b）CPU 利用率　（c）子树深度　(d) 调用的微代理数量

图 5：在四种大语言模型下，使用所提出的 ReCiSt 框架对*Hadoop* 进行性能评估

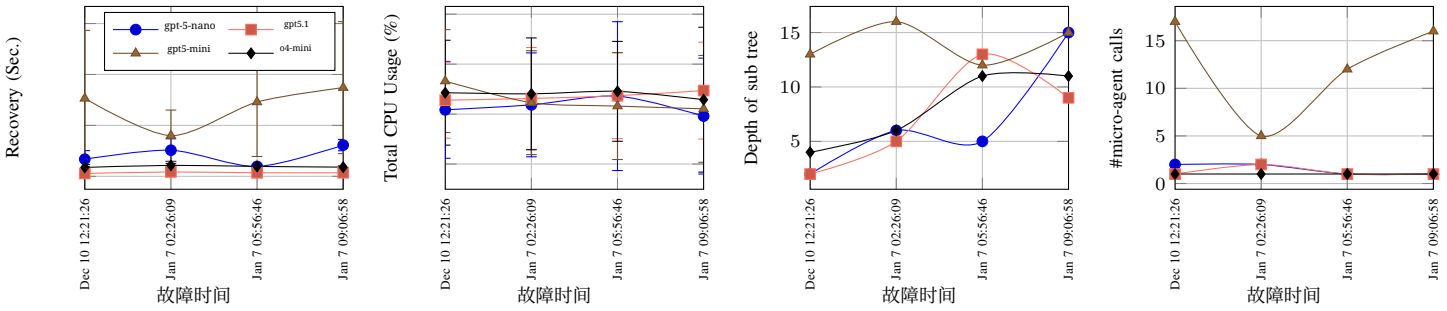
以及其结果如图 5 所示。

从图 5a 可以看出，在 Hadoop 数据集的时间线上，四种受评估模型在五个故障实例中观察到的自愈时间。图中显示，gpt-5.1 的恢复时间相对较短，通常低于 100 秒。此外，o4-mini 的恢复行为也很迅速，自愈时间大多集中在约 60–130 秒的狭窄范围内。相比之下，gpt-5-nano 的自愈时间介于 200 至 500 秒之间，略高于 gpt-5.1 和 o4-mini。最后，gpt-5-mini 的结果最慢，时间从 200 秒延伸至超过 1,000 秒。图 5b 展示了 Hadoop 数据集的 CPU 利用率及其故障情况。所有模型在 Hadoop 数据集上的 CPU 使用率都保持在良好控制范围内。具体而言，gpt-5-nano 和 gpt-5-mini 的平均 CPU 利用率均为 ≈14 %，而 gpt-5.1 和 o4-mini 的利用率略高，约为 15%。从图 5c 可以看出，不同受评估模型的故障复杂度存在显著差异。例如，gpt-5-mini 的结构复杂度最高，其子树包含 ≈14–20 条路径。相比之下，o4-mini 的结构较为简单，通常仅包含约 5–10 条路径。从图 5d 可以看出，微代理调用模式存在明显差异。例如，在观察到的五个故障实例中，o4-mini 仅依赖 1–2 次微代理调用。gpt-5-nano 需要 1 至 6 次微代理调用。相比之下，gpt-5.1 调用的微代理数量更多，其中一个案例多达 35 次，表明其推理过程十分密集。尽管过程中经历了许多起伏，四种模型最终都成功地在 ReCiSt 框架下，使 Hadoop 数据集达到自愈状态。

4) OpenSSH 数据集： OpenSSH 是一种基于 SSH 协议、广泛用于安全远程访问的工具。相应的 OpenSSH 数据集包含与认证相关事件和一般系统活动有关的日志记录。在该数据集中，可能的故障与认证尝试中断或未成功有关，包括由暴力破解攻击或错误登录尝试引起的 SSH 认证故障。这些故障及其自愈所需的时间如图 6a 所示。由此我们注意到，gpt-5.1 是最快的自恢复代理，通常耗时在 27 到 42 秒之间。另一方面，o4-mini 也表现出比 gpt-5-nano 和 gpt-5-mini 更快的恢复速度，耗时范围为≈87–107 秒。但是，gpt-5-nano 和 gpt-5-mini 的恢复时间较长，超过 300–800 秒。如图 6b 所示，在 CPU 利用率方面，gpt-5.1 的 CPU 利用率最高，约为 15%，其次是 o4-mini，约为 14%。相比之下，gpt-5-nano 和 gpt-5-mini 的利用率均较低，接近 13%。在分析深度和微代理调用次数方面，分别如图 6c 和图 6d 所示，gpt-5-mini 展现出最复杂的结构，其子树包含 ≈13–16 条路径，并需要约 5–17 次微代理调用。相比之下，o4-mini、gpt-5-nano 和 gpt-5.1 展现出更简单的恢复结构，通常涉及 ≈2–15 条路径，仅需 1–2 次微代理调用。

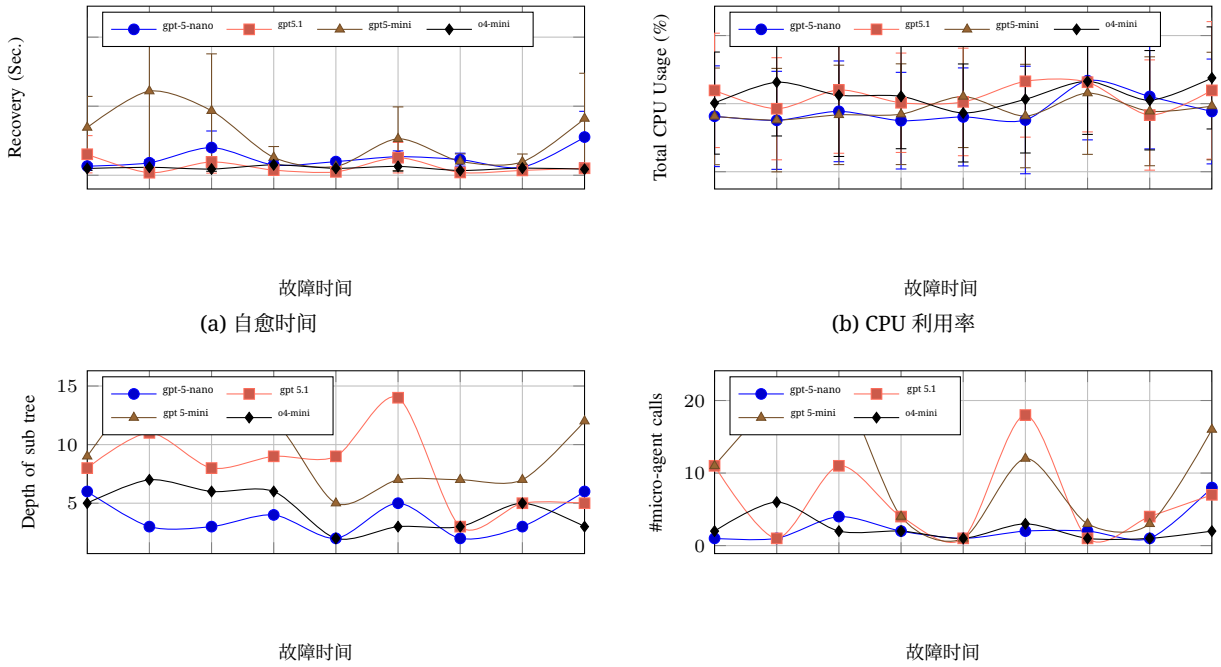
5) Blue Gene/L (BGL) 数据集： Blue Gene/L (BGL) 数据集是一个开放日志集合，来自一个由 131,072 个处理器和 32768 GB 内存组成的超级计算系统。该数据集包含反映多种系统故障的告警消息，包括致命 I/O 错误、硬件和操作系统故障、内核级故障、通信中断、机器检查中断、内核崩溃，以及 CPU 或缓存相关的硬件故障。图 7a 描绘了所评估模型沿 BGL 数据集时间线在九个故障实例上的自愈时间。结果显示，gpt-5.1 和 o4-mini 是最快的自恢复智能体：gpt-5.1 所需时间低于 250 秒，而 o4-mini 低于 150 秒。相比之下，gpt-5-nano 和 gpt-5-mini 的恢复速度较慢，自愈时间分别超过 500 秒和 1,200 秒。图 7b 绘制了 BGL 数据的 CPU 使用率，其中显示了 BGL 数据集中报告故障时间对应的九个故障实例。这表明，gpt-5.1 和 o4-mini 的 CPU 利用率略高，为 ≈11%，而 gpt-5-nano 和 gpt-5-mini 的利用率较低，接近 9%。从图 7c 和图 7d 可以看出，gpt-5-mini 和 gpt-5.1 呈现出更复杂的诊断结构，其子树跨度约为 3–15 条路径，并需要 1 至 22 次微代理调用。相比之下，o4-mini 和 gpt-5-nano 展现出更简单的诊断结构，通常涉及约 2–7 条 DFS 路径和 1–8 次微代理调用。

6）智能体决策质量：我们还评估了微型智能体在推理过程中的决策，以量化其决策质量。在此过程中，我们记录了每个



(a) 自愈时间 (b) CPU 利用率 (c) 子树深度 (d) 调用的微代理数量

图 6：在四种大语言模型下，使用所提出的 ReCiSt 框架对*OpenSSH* 进行的性能评估



(a) 自愈时间 (b) CPU 利用率

(c) 子树深度 (d) 调用的微代理数量

图 7：使用四种大语言模型评估所提出的 ReCiSt 框架在 BGL 数据集上的性能

故障案例中建设性指导响应的比例、可能导致关键绩效指标下降并对系统行为产生负面影响的响应比例、证据支持不足的响应比例，以及证据充分的响应比例。这些评估进一步阐明了因果推理和对操作安全原则的遵循情况。此外，我们还将这些测量结果与推理深度比率进行关联分析，该比率定义为基于子树深度的系统级微型代理调用次数占比。结果汇总见表 I。

从表 I 可以看出，CloudStateless 数据集在 gpt-5-mini 上获得了支持性响应，其接受率高达≈95%，完整推理深度率为 100%。尽管有害响应率较高，达到 31.8%，但其深度且一致的推理表现仍然出色。相比之下，gpt-5 nano 的拒绝率较高，为 ≈24%，并且严重依赖扩展的微型代理级推理链条 ≈98%，但这并未转化为响应质量的提升 ≈7%。我们确认，数据集中的结果表明，仅有更深入的推理过程并不足以确保输出高质量或安全。此外，o4-mini 在多种故障中的有害输出率较低 ≈5%，同时推理深度率较高（9%），最佳解决方案的比例也较高 ≈53%，显示出其内部推理的校准效果更佳。类似地，在 ZooKeeper 和 OpenSSH 数据集中，四个模型均未产生已拒绝或有害的响应。此外，gpt-5-nano、gpt-5.1 和 o4-mini 的最佳响应率均达到 100 %。然而，这一表现是在推理深度相对较低的情况下取得的（例如，gpt-5-nano 为 27.9%，o4-mini 为 15%）。这清楚地表明，当故障模式简单时，无需深入推理也能实现有效恢复。在 BGL 和 Hadoop 等更复杂的环境中，表现差异则更加显著。例如，在 BGL 中，gpt-5-mini 在.....之间取得平衡

表 I：五个数据集和四种模型中智能体响应的决策质量

数据集	模型名称	Best	已接受	已拒绝	有害	RDR
云无状态	GPT-5 Nano	0.077	0.753	0.244	0.077	0.986
	gpt 5.1	0.462	0.733	0.162	0.154	0.728
	GPT-5 Mini	0	0.955	0.051	0.318	1
	o4 mini	0.538	0.628	0.301	0.055	0.905
ZooKeeper	GPT-5 Nano	1	0.625	0	0	0.279
	gpt 5.1	1	0.429	0	0	0.392
	GPT-5 Mini	0.5	0.978	0	0	0.942
	o4 mini	1	0	0	0	0.15
OpenSSH	GPT-5 Nano	1	0.25	0	0	0.288
	gpt 5.1	1	0.125	0	0	0.195
	GPT-5 Mini	0.75	0.914	0	0	0.733
	o4 mini	1	0.125	0	0	0.170
BGL	GPT-5 Nano	0.778	0.431	0	0	0.499
	gpt 5.1	0.556	0.611	0	0	0.569
	GPT-5 Mini	0.556	0.781	0	0.111	0.639
	o4 mini	1	0.315	0.222	0.111	0.402
Hadoop	GPT-5 Nano	1	0.56	0	0	0.315
	gpt 5.1	0.6	0.7	0	0	0.622
	GPT-5 Mini	0.8	0.539	0.2	0	0.349
	o4 mini	1	0.3	0	0	0.187

较高的已接受响应率 ≈78.1%，并兼具可观的推理深度 ≈63.9% 和 11% 的有害率。这些结果证实，反馈驱动的元认知控制能够在不同领域中有效平衡响应质量、安全性与计算开销。

VI. 结论

本文提出了**ReCiSt**，这是一种受生物学启发的智能体自愈框架，将生物伤口愈合阶段映射为自主遏制、诊断、元认知推理和知识重构层，以增强 DCCS 的韧性。我们基于语言模型驱动的智能体实现并测试了该方法，并使用多个大语言模型进行了评估，包括 gpt-5.1、gpt-5-mini、gpt-5-nano 和 o4-mini；评估所用数据集涵盖公开且异构的数据集，具体包括云无状态系统数据集，以及 LogHub 数据集中的 ZooKeeper、Hadoop、OpenSSH 和 BGL。这些数据集涵盖了多种故障类型，从网络中断和身份验证故障到大规模超级计算错误。总体而言，gpt-5.1 和 o4-mini 等模型恢复速度最快，能够在数十秒内实现自愈，同时将 CPU 利用率保持在 10% 以下。尽管过程中起伏不少，但所有模型均证实了该方法适用于具有韧性且异构的计算环境。我们工作的主要局限在于实验是在受控实验环境中使用离线日志数据集并通过基于云的执行开展的。当前设置未明确测试实时网络拥塞、硬件异构性、移动性、安全约束以及长期运行漂移等实时因素。未来，我们将在真实的连续体环境中部署并验证该框架，并评估其在实时工作负载和不断演变的故障条件下的鲁棒性。

参考文献

[1] P. K. Donta、B. Sedlak、I. Murturi、V. C. Pujol 和 S.Dustdar，《计算连续体系统中的类人分布式智能》，*IEEE Internet Computing*，第 29 卷，第 2 期，第 61–68 页，2025年。

[2] S. Dustdar、V. C. Pujol 和 P. K. Donta，“分布式计算连续体系统”，IEEE Transactions on Knowledge and Data Engineering，第35卷第4期，页码4092–4105，2023年。[3] O. Babaoglu 等，“面向分布式计算的生物学设计模式”，ACM Trans. Auton. Adapt. Syst.，第1卷第1期，页码26–66，2022年9月。在线。获取地址：https://doi.org/ 。/ 。S. Enoch 和 D. J. Leaper，“伤口愈合基础科学”，Surgery (oxford)，第26卷第2期，页码31–37，2008年。 [在线]。获取地址：https://doi.org/ 10 1383 23 2 37 60352 。/surg. 。. 。A. Saleh、S. Tarkoma、[5] A. Lindgren、P. K. Donta、S. Dustdar、S. Pirttikangas 和 L. Lov´en，“MemIndex：面向多智能体系统的智能体式事件驱动分布式内存管理”，ACM Trans. Auton. Adapt. Syst.，2025年11月，刚刚已接受。 [在线]。获取地址：https://doi.org/ 10 1145 3774946 。/ D. Fisch、D. Fisch、M. [6] Janicke、E. Kalkowski 和 B. Sick，“动态变化环境中的知识获取技术”，ACM Trans. Auton. Adapt. Syst.，第7卷第1期，2012年5月。 [在线]。获取地址：https://doi.org/ 10 1145 2168260 2168276 。/ 。H. Alves 等，[7] “G韧性——白皮书”，2025年。 [在线]。获取地址：https://arxiv.org/abs/ 6 2509 09005 。A. Altaweel、C. Yang、R. Stoleru、[8] S. Bhunia、M. Sagor、M. Maurice 和 R. Blalock，“RSock：面向移动雾/边缘网络的弹性路由协议”，Ad Hoc Netw.，第134卷，C期，2022年9月。 [ ] 10 1016 2022 102926 在线。获取地址：[9] https://doi.org/ 。/j.adhoc. 。F. Nakayama、P. Lenz 和 M. Nogueira，“面向便携式辅助生活通信的韧性管理架构”，IEEE Trans. on Netw. Serv. Manag.，第19卷第3期，页码2536–2548，2022年。[10] S. Qi、H. Feng、X. Liu 和 X. Jin，“通过非对称日志记录实现有状态无服务器计算的高效容错”，ACM Trans. Comput. Syst.，第43卷第1–2期，2025年6月。 [在线]。获取地址：https://doi.org/ 10 1145 3725985 。/ T. Sen、H. Shen、W. Saad 和 T. Doan，“支[11] 持信息物理系统的弹性稳健边缘云网络系统”，载于2021年IEEE第18届移动自组织与智能系统国际会议，2021年，页码234–242。[12] J. Diaz-de Arcaya、A. I. Torre-Bastida、L. Bonilla、J. L´opez-de Armentia、R. Mi˜n´on、G. Zarate 和 A. Almeida，“Akats：利用联邦机器学习技术实现边缘计算环境弹性部署的系统”，载于2023年第8届智能与可持续技术国际会议（SpliTech），2023年，页码1–4。[13] V. Kashyap、R. Ahuja 和 A. Kumar，“雾计算环境中基于预测分析的负载均衡与容错”，Cluster Computing，第28卷第5期，2025年4月。 [在线]。获取地址：https://doi.org/ 10 1007 10586 024 04984 5 。/s - - - J. Nikolic、[14] N. Jubatyrov 和 E. Pournaras，“分布式系统中的自愈困境：故障纠正与容错”，IEEE Trans. on Netw. Serv. Manag.，第18卷第3期，页码2728–2741，2021年。[15] H. Fang、P. Yu、C. Tan、J. Zhang、D. Lin、L. Zhang、Y. Zhang、W. Li 和 L. Meng，“知识驱动自治网络中的自愈：背景、挑战与未来方向”，IEEE Network，第38卷第6期，页码425–432，2024年。[16] Y. Wang、S. Liu、J. Fang 和 Z. Meng，“EvoAgentX：用于演进智能体工作流的自动化框架”，2025年。 [在线]。获取地址：https://arxiv.org/abs/ 2507 03616 。J. Wu，“Git上下文控制器：[17] 像Git一样管理基于大语言模型的智能体上下文”，2025 [ ] 2508 00031 。在线。获取地[18] 址：https://arxiv.org/abs/ 。W. Sun、X. Cheng、J. Fan、Y. Xu、X. Yu、S. He、J. Zhao 和 K. Liu，“面向搜索环境中的智能体式自学习大语言模型”，2025年。 [ ] 2510 14253 在线。获取地址：[19] https://arxiv.org/abs/ 。A. Qayyum、A. Albaseer、J. Qadir、A. Al-Fuqaha 和 M. Abdallah，“面向智能自组织网络的大语言模型驱动多智能体架构”，IEEE Network，页码1–10，2025年。[20] X. Wu、Y. Wang、J. Farooq 和 J. Chen，“利用大语言模型驱动的智能体人工智能提升开放无线接入网在下一代网络中的韧性”，2025年3月。 [在线]。获取地址：http://dx.doi.org/10.36227/techrxiv.174284755.59863143/v1[21] C. Wang 等，“面向生产级网络中的基于大语言模型的故障定位”，载于ACM SIGCOMM 2025会议论文集，系列：SIGCOMM '25。美国纽约州纽约市：ACM，2025年，页码 496 511 [ ] 10 1145 3718958 3750505 – 。在线。获[22] 取地址：https://doi.org/ 。/ 。H. Fang、D. Zhang、C. Tan、P. Yu、Y. Wang 和 W. Li，“用于主动容错边缘网络的大语言模型增强自治智能体”，载于IEEE INFOCOM 2024 - IEEE计算机通信会议研讨会（INFOCOM WKSHPS），2024年，页码1–2。[23] G. Eleftherakis、F. Baxhaku 和 A. Vasilescu，“无线传感器网络的仿生自适应架构”，载于第26届泛希腊信息学会议 ，系列。PCI'22。

美国纽约州纽约市：ACM，2023年，第 116–122 页。 [在线]。可访问：https://doi.org/ 10 1145 3575879 3575976 。/。S. [24] Balasubramaniam、D. Botvich、W. Donnelly、M. O. Foghl´u 和 J. Strassner，“面向自主管理网络的生物启发式自我治理与自我组织”，载于第一届生物启发式网络、信息与计算系统模型国际会议论文集，系列。BIONETICS ’06。美国纽约州纽约市：ACM，2006年，第 30–页。 [在线]。可访问：https://doi.org/ 10 1145 1315843 1315880 。/。 [25] M. Samie、G. Dragffy 和 T. Pipe，“面向可进化容错硬件系统的新型生物启发式自修复算法”，载于第十一届遗传与进化计算大会伴随会议：最新研究论文，系列。GECCO ’09。美国纽约州纽约市：ACM，2009年，第 2143–2148 页。 [在线]。可访问：https://doi.org/ 10 1145 1570256 1570291 。/。A. Madureira、F. Santos [26] 和 I. Pereira，“用于制造业动态调度的自管理智能体”，载于第十届遗传与进化计算大会伴随会议，系列。GECCO ’08。美国纽约州纽约市：ACM，2008年，第 2187–2192 页。 [在线]。可访问：https://doi.org/10.1145/1388969。1389045[27] T. K、K. Kanagasabapathi、V. A、R. R、A. M. Tripathi 和 G. G，“利用生物启发式算法构建自愈网络：增强现代网络的韧性”，载于*2025*年第三届通信、安全与人工智能国际会议（*ICCSAI*），第 3 卷，2025年，第 865–871 页。[28] S. O. Husain、V. A. Narayana、T. Annapurna、B. S. Kumar、V. Shunmu-gapriya 和 T. Srihari，“用于计算网络自主故障恢复的生物启发式自愈机器学习框架”，载于*2025*年元宇宙与计算当前趋势国际会议（*ICMCTC*），2025年，第 1–4 页。[29] P. K. Donta、A. Lapkovskis、E. Mingozzi 和 S. Dustdar，“韧性源于设计——面向分布式连续体智能的主动推理”，2025年。 [ ] 2511 07202 在线。可访问：[30] https://arxiv.org/abs/。P. K. Donta、B. Sedlak、V. Casamayor Pujol 和 S. Dustdar，“分布式连续体系统的治理与可持续性：一种大数据方法”，《大数据期刊》，第10卷第1期，第53页，2023年。 [ ] 10 1186 40537 023 00737 0 在线。可访问：[31] 2025, https://doi.org/。/s - - - LangChain，“Langchain”，https://www.langchain.com/，最后访问日期：2026年1月5日。[32] Pydantic，“pydantic”，https://docs.pydantic.dev/latest/api/base model/，2025年，最后访问日期：2026年1月5日。[33] OpenAI，“OpenAI模型”，https://platform.openai.com/docs/models，2025年，最后访问日期：2026年1月5日。[34] huggingface，“all-minilm-l6-v2”，https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2，2025年，最后访问日期：2026年1月5日。[35] P. K. Donta、Q. Zhang 和 S. Dustdar，“以人工智能为中心的计算连续体系统中的性能测量”，载于*2025*年第*11*届*IEEE*物联网世界论坛（*WF-IoT*），2025年，第1–6页。[36] N. Chairatana 和 R. Chawuthai，“云无状态系统性能指标与状态”，2024年。 [在线]。可访问：https://dx.doi.org/10。 21227 8 2 2 40 / wf - y J. Zhu、S. He、P. He、J. Liu 和 M. R. Lyu，“[37] Loghub：用于人工智能驱动日志分析的大型系统日志数据集”，载于*2023*年第*34*届*IEEE*国际软件可靠性工程研讨会（*ISSRE*）。美国加利福尼亚州洛斯阿拉米托斯：IEEE计算机学会，2023年10月，第355–366页。 [在线]。可访问：https://doi.ieeecomputersociety.org/ 10 1109 59848 2023 00071 。/ISSRE。Z. [38] Jiang、J. Liu、J. Huang、Y. Li、Y. Huo、J. Gu、Z. Chen、J. Zhu 和 M. R. Lyu，“日志解析技术的大规模评估：我们还有多远？”，载于第*33*届*ACM SIGSOFT*国际软件测试与分析研讨会论文集，丛书：ISSTA 2024。美国纽约州纽约市：ACM，2024年，第223–234页。 [在线]。可访问：https://doi.org/10.1145/3650212.3652123