OpenAIOps

AIOPS | 2025 CCF国际AIOps挑战赛
2025 CCF International AIOps Challenge

基于大模型智能体的根因定位框架

腾讯科技WXG技术架构部团队 FastReject

颜秋宇 罗宇 黄皓佳

主办单位：中国计算机学会（CCF）
承办单位：中国计算机学会互联网专委会、中国科学院计算机网络信息中心、中国移动研究院、清华大学
协办单位：华为2012实验室、阿里云、中兴通讯、中国移动九天团队、南开大学、西安电子科技大学、清华大学计算机科学与技术系、神州灵云

OpenAIOps AIOPS | 2025 CCF国际AIOps挑战赛
2025 CCF International AIOps Challenge

目录
CONTENTS

第一章节 方案概况
第二章节 方案实现
第三章节 总结与展望

第一章节

方案概况

1.1 背景：赛题数据分析与面临的挑战

赛题：赛道一 基于大模型智能体的微服务根因定位

目标：融合指标、日志和调用链数据，**快速**、**准确**地识别出导致服务异常的根本原因组件或节点

面临的挑战

多模态数据融合理解

- LLM统一理解和关联不同模态的数据

领域知识注入与适应性

- 知识的更新与沉淀自动化迭代

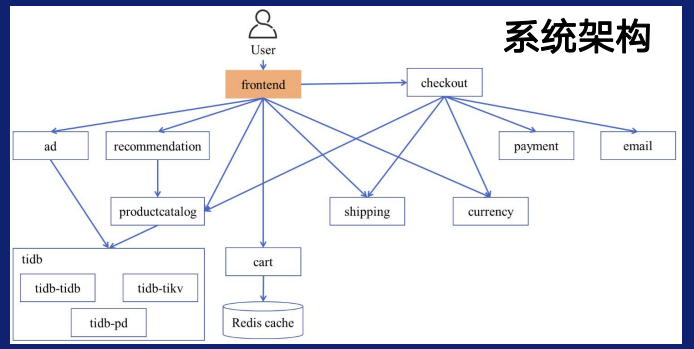
性能开销与实时性

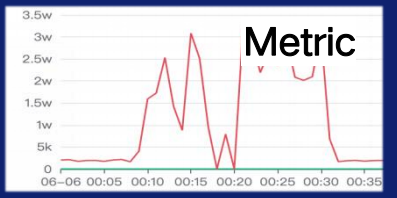
- 故障场景快速响应的SLA要求

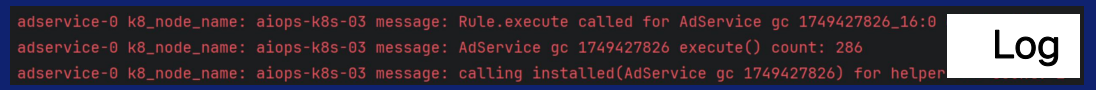
长推理过程的可靠性

- 确定且可复现的推理过程

数据分析







- 数据类型：Metric、Trace、Log
- 系统规模：Service、Pod、Node
- TiDB组件：Tidb、Tikv、PD

1.2 方案：在线推理与离线学习的闭环架构

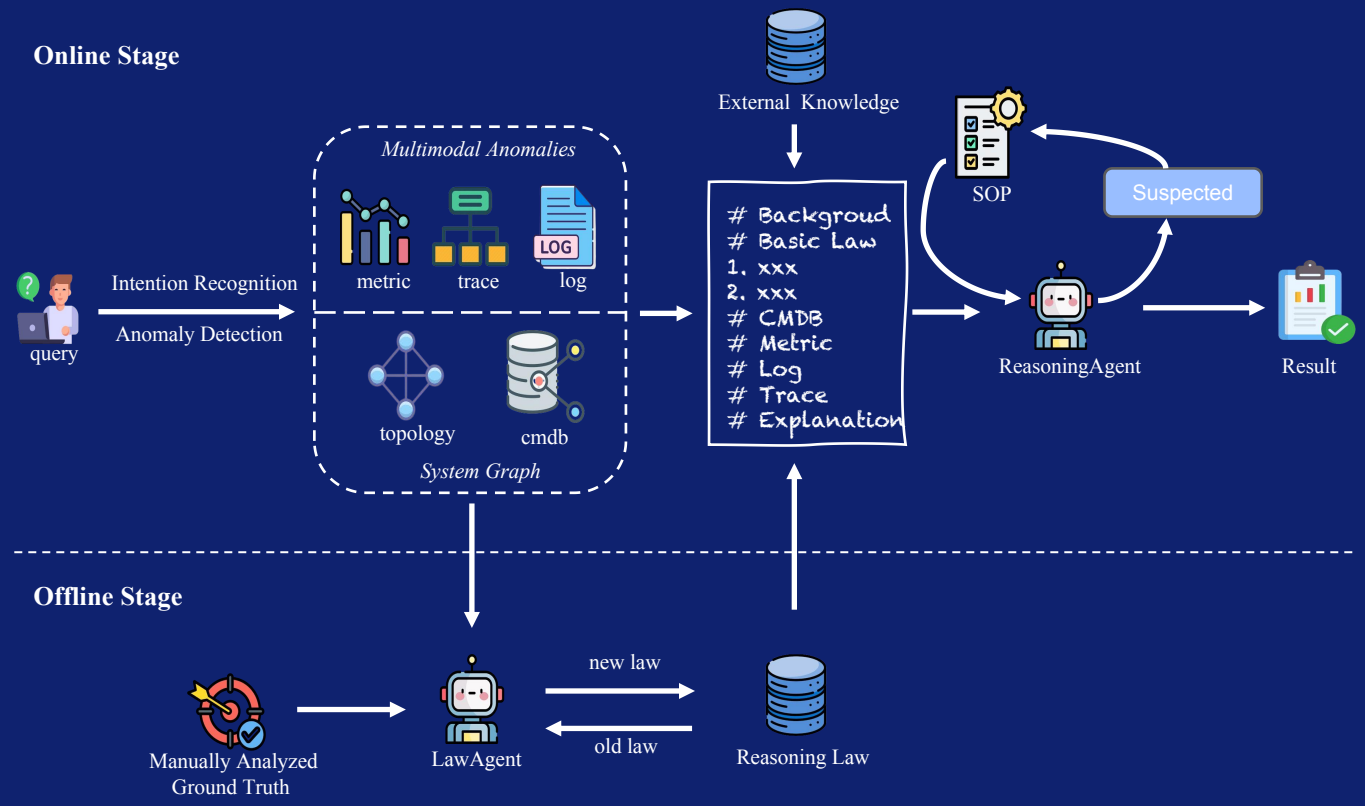
基于**多模态数据结构化处理**、**领域知识自迭代融合**、**SOP分析路径约束**的LLM智能体根因定位框架

在线阶段

实时分析可观测数据生成异常报告，结合知识库与定位规则进行根因推理定位，并对可疑结果按SOP校验以确保准确性

离线阶段

从历史人工定位结果中提炼可泛化的经验规则，持续优化规则库以推动历史经验在新场景中的迁移与应用



1.3 效果：实现1分钟内高效且稳定的LLM智能体系统

基于**多模态数据结构化处理、领域知识自迭代融合、SOP分析路径约束**的LLM智能体根因定位框架

- **实用性**：实现了运维领域中常见的三种模态数据异常检测和数据处理方案，广泛适配运维场景
- **创新性**：在多模态可观测数据处理、领域知识融合、大模型可靠性推理都提出了创新方案
- **扩展性**：具备定位规则自学习与自迭代的能力，实现了历史案例的沉淀和复用，展现出较强的泛化能力

数据集上取得了73.95得分，最高分达到74.2，客观分 **第一名**

平均每个问题的耗时约为**1分钟内**，**平均消耗Token数9200.5**

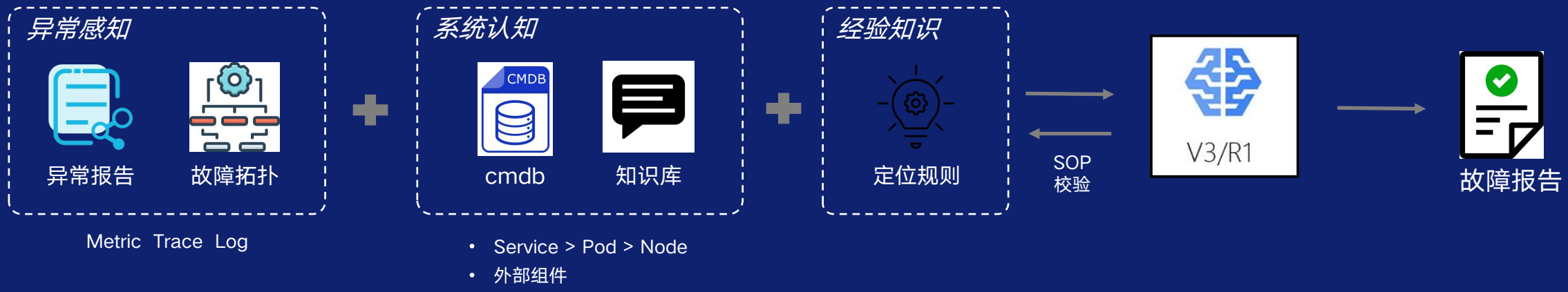
初赛排名	团队名称	分数1	分数2	分数3	平均分	复现后排名	排名变化
1	FastReject	74.2	73.71	73.94	73.95	1	0
4	xwy6475	55.36	53.72	54.77	54.62	2	2
3	hwlyyzc	55.06	54.21	54.54	54.60	3	0
2	我们是有技术的	49.81	49.57	49.53	49.64	4	-2

第二章节

方案实现

2. 方案实现：创新点介绍

根因定位：融合实时异常报告、系统基础背景、经验知识与预定义规则进行综合推理与分析



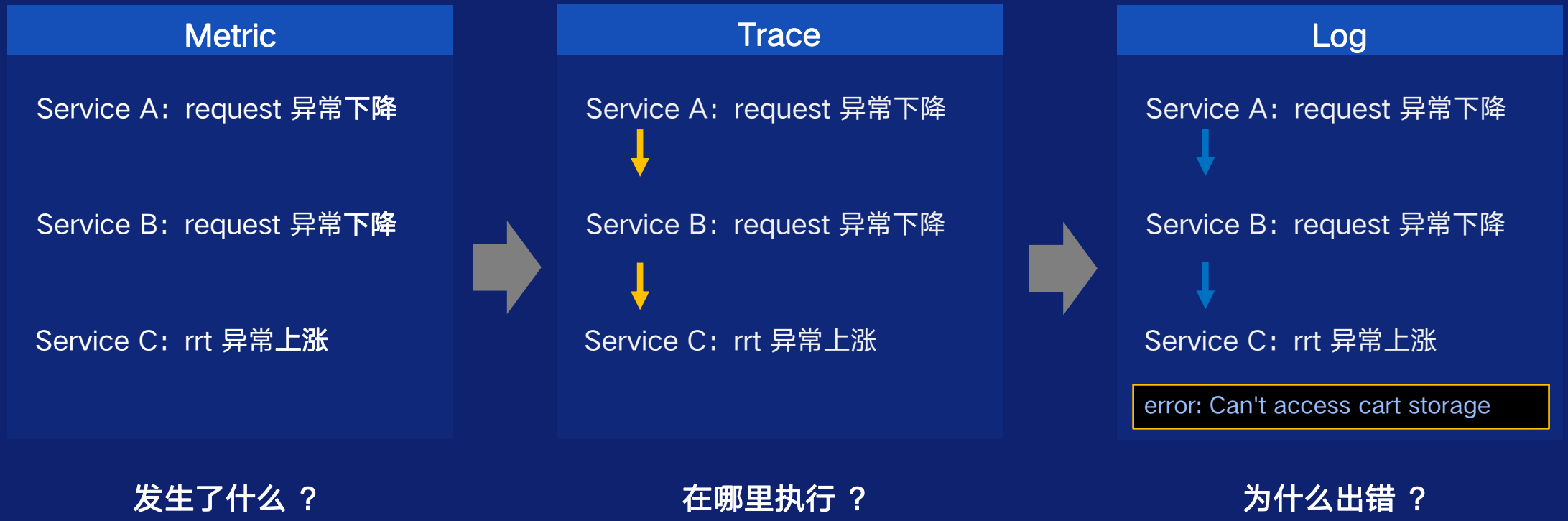
创新点

1. 异常感知：将多模态数据转化为结构化异常报告，为模型推理提供有效证据链
2. 系统认知：融合静态拓扑、动态关联关系、知识库构建的推理增强，为模型提供全局视角分析问题
3. 知识迭代：从经验知识库中实时匹配相关的定位经验规则（由离线流程更新）
4. 结果校验：对可疑的定位结果，遵循SOP进行自动化校验，输出最终根因

2.1 异常感知：多模态可观测数据结构化文本表示

问题：如何让大模型理解多模态数据是实现高效根因定位的关键

方案：统一语义表示，将多模态数据转化为大模型**可理解**、**可关联**、**可推理**的结构化文本



2.1 异常感知：多模态可观测数据结构化文本表示

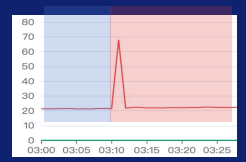
✓ Metric：提取异常曲线变化的关键特征

异常检测：按照 Service > Pod > Node 逐层下钻 进行异常检测，根据曲线特征适配不同算法

上下文注入：从时间序列 **数值特征** (均值变化率) 和 **曲线特征** (阶梯曲线和尖峰曲线) 提取异常指标

- [指标名]
「对象名」「变化趋势」「均值变化率」（「正常均值」-> 「异常均值」）（「曲线特征」）

效果



memory_usage_rate 67.94%

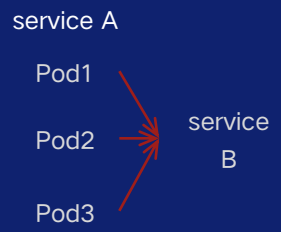
```
# The node infra metric:
- node_memory_usage_rate:
Node aiops-k8s-06 increase
12.88% (21.27 -> 24.01)
(spike to 67.94)
```

✓ Trace：构建异常传播拓扑链路

主调服务→被调服务　　平均耗时、调用数、失败数　　提取错误状态码信息模版　　构建异常拓扑链路

数据汇聚 → 指标统计 → 异常检测 → 拓扑构建

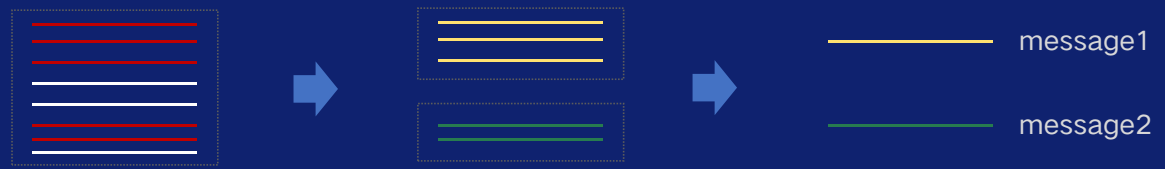
[span_pattern]: 「Service A」（「Pod」）-> 「Service B」 [指标名]: 「指标值」
[error_message]: HTTP status code: 500




```
[span_pattern]:
frontend(frontend-0,frontend-
1,frontend-2) -> cartservice,
port: 7070, [count]: 140
[error_message]:
 - Can't access cart storage.
```

✓ Log：模板提取结构化日志

异常关键词日志过滤　　drain算法模板提取　　生成结构化异常日志文本



「Template ID」「Template Count」「Template Message」

```
"@timestamp": 1749867501407,
"k8_pod": "adservice-1",
"k8_node_name": "aiops-k8s-07",
"message": "AdService gc 1749867023 execute()",
"template_id": 2,
"template_count": 201
```

2.2 系统认知：融合静态拓扑与动态关联关系

问题：缺乏系统实体间的关联关系，会导致大模型推理链条长，容易陷入无法自证的逻辑循环中

方案：系统全链路**静态拓扑**和系统**动态CMDB**部署信息可以帮助模型更好地理解故障传播关系，将割裂的数据串联起来

✓ 静态拓扑：服务之间的调用依赖关系，将HipsterShop全拓扑架构拆成从头到尾的调用链，作为理解故障传播的基础骨架

```
frontend -> adservice -> tidb
frontend -> productcatalogservice -> tidb
.....
```

优化前

3260fa48-316

Other services like cartservice and shippingservice have client_errors, but their issues might be downstream effects. For example, if frontend is failing, it can't send requests properly to downstream services, causing them to show decreased requests and possible errors. Law 5.1 suggests that if a service's downstream components have issues but others don't, the root cause is the downstream. However, here frontend is the entry point, and multiple downstreams are affected, pointing back to frontend as the source.

The node infra metrics show a spike in disk read on nodes 03 and 06, but no node's CPU or memory is mentioned as high. Law 3 says node issues are when CPU/memory approach 80-90%, which isn't the case here. So node-level issues are less likely.

Tidb-tidb has decreased uptime and connection counts, but since the frontend's errors are more critical and directly impacting user requests, the primary issue is frontend. The tidb issues might be a secondary effect or unrelated.

Therefore, the root cause component is the frontend service.

优化后

So the chain is frontend -> productcatalogservice -> tidb-tidb. If tidb-tidb is down, productcatalogservice can't process requests, leading to high rrt and frontend errors.

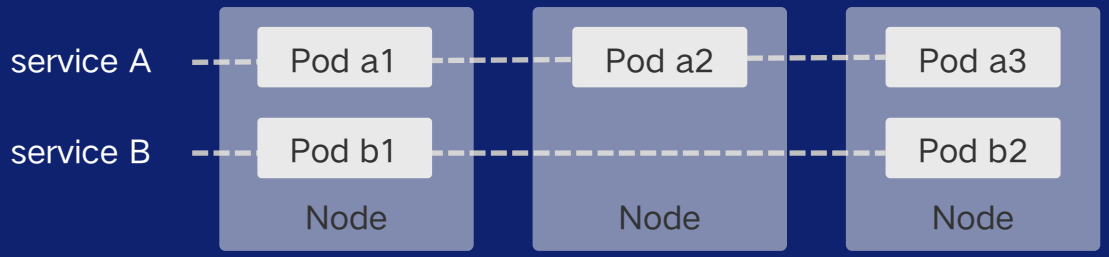
Therefore, the root cause component is tidb-tidb.

But wait, the productcatalogservice's rrt is extremely high. If tidb-tidb is down, productcatalogservice would have errors or timeouts when trying to connect, which would increase rrt. So the high rrt in productcatalogservice is a symptom of tidb-tidb being down.

Therefore, according to Basic Law 5.1, since frontend's issue is due to productcatalogservice, which in turn is due to tidb-tidb, the root cause is tidb-tidb.

But the logs mention "Region issue for tikv" in the background, but there's no log data here. However, the tidb-tidb's own metrics indicate it's not functioning. So the component should be tidb-tidb.

✓ 动态关联：k8s集群部署关系动态变化，实时提取三元组关系作为大模型推理分析的关键上下文信息



service_name -> pod_name -> node_name

优化前

8cf433b4-122

Law 8.1 says if only one pod is abnormal, it's the root cause. Since the service-level metrics are up (due to other pods), the problem is with currencyservice-0. The node's memory spike might be a red herring, or maybe the pod is on that node. But the user didn't specify which node the pod is on. The CMDB info isn't provided, so we can't confirm.

But according to Law 8, if service-level has persistent anomalies, it's a service issue. But here, the service's metrics are up, not down. The anomaly is the pod's decrease. So per Law 8, since the service-level isn't showing a decrease (it's actually increased), the pod's issue is the root cause. The node's memory spike might be a contributing factor if the pod is on that node, but without CMDB data, we can't be sure.

The logs don't mention any errors, so maybe the pod is just not receiving requests. The reason could be a network issue or the pod being unhealthy. But the key here is the pod's metrics. So the root cause component should be currencyservice-0.

优化后

Given the Basic Law 3, the node's memory spike is significant. Even if the currencyservice is showing anomalies, the node's issue might be the root cause affecting services on it, which in turn affect others. Since frontend is on aiops-k8s-03, and frontend calls currencyservice, if frontend is struggling due to node issues, it might be generating errors or retries, leading to increased load on currencyservice.

Therefore, the root cause component is node aiops-k8s-03 due to its memory spike, which is affecting services running on it, leading to cascading effects on currencyservice.

2.2 系统认知：基于运维知识库的推理增强

问题：监控指标数据缺乏必要元信息说明（指标含义、合理范围以及量纲单位），进一步增加LLM的理解难度

方案：结合比赛官网的指标解释构建指标元信息知识库，统一量化指标认知

由于数据采集组件不一致，相同指标的量纲存在差异，在数值上会误导模型分析

- 异常指标1：Pod　CPU使用率 0.4
- 异常指标2：Node CPU使用率 40
- 异常指标3：Node 读取字节数 1680.4

异常指标严重程度认知

优化前：异常指标3 > 异常指标1 > 异常指标2

优化后：异常指标1 = 异常指标2 > 异常指标3

问题：大模型难以准确理解TiDB相关错误日志的深层含义，导致无法正确识别故障组件

方案：自动拉取TiDB官网文档构建TiDB常见错误信息的知识库，实现精准信息检索与增强

Log异常信息：Error 9005 (HY000): Region is unavailable

Error Number: 9005

The complete error message: `ERROR 9005 (HY000): Region is unavailable`

The accessed Region or a certain Raft Group is not available, with possible reasons such as insufficient replicas. This error usually occurs when the TiKV server is busy or the TiKV node is down.

实时匹配：当检测到TiDB相关日志时，自动触发知识库检索，将检索结果注入提示词

优化前：the error is "Region is unavailable" which is more related to TiKV. However, the store_up_count in PD is a critical metric here. If PD reports fewer stores, then regions might not have enough replicas.

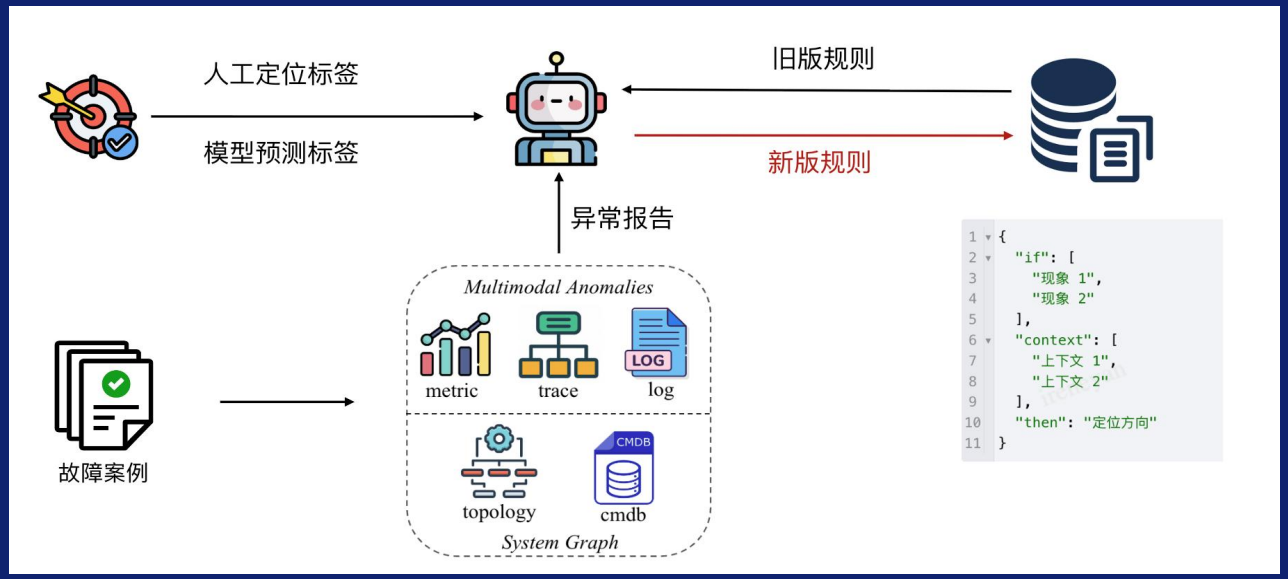
优化后： The store_up_count in PD dropping suggests that PD is aware of TiKV nodes being down. However, the actual problem causing the region unavailability is likely with TiKV (since regions are part of TiKV).

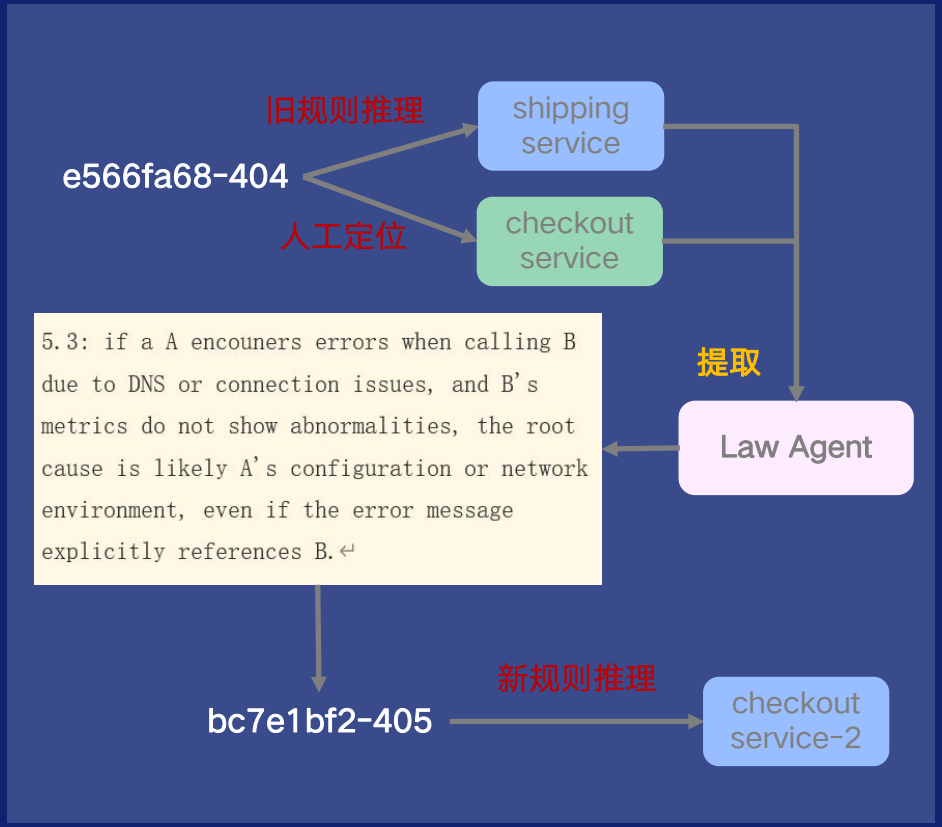
2.3 Law Agent：专家经验自动化规则提取更新

专家定位经验起着关键作用，本质上是经过故障案例验证的决策模式，既包含**成功**的定位路径，也涵盖需要**规避**的误判方向

问题：依赖人工分析效率低，如何将隐性、模糊的经验转化为可理解、可执行的**显性规则**

方案：我们实现了具有**自我迭代**与**持续进化**能力的定位经验规则智能体





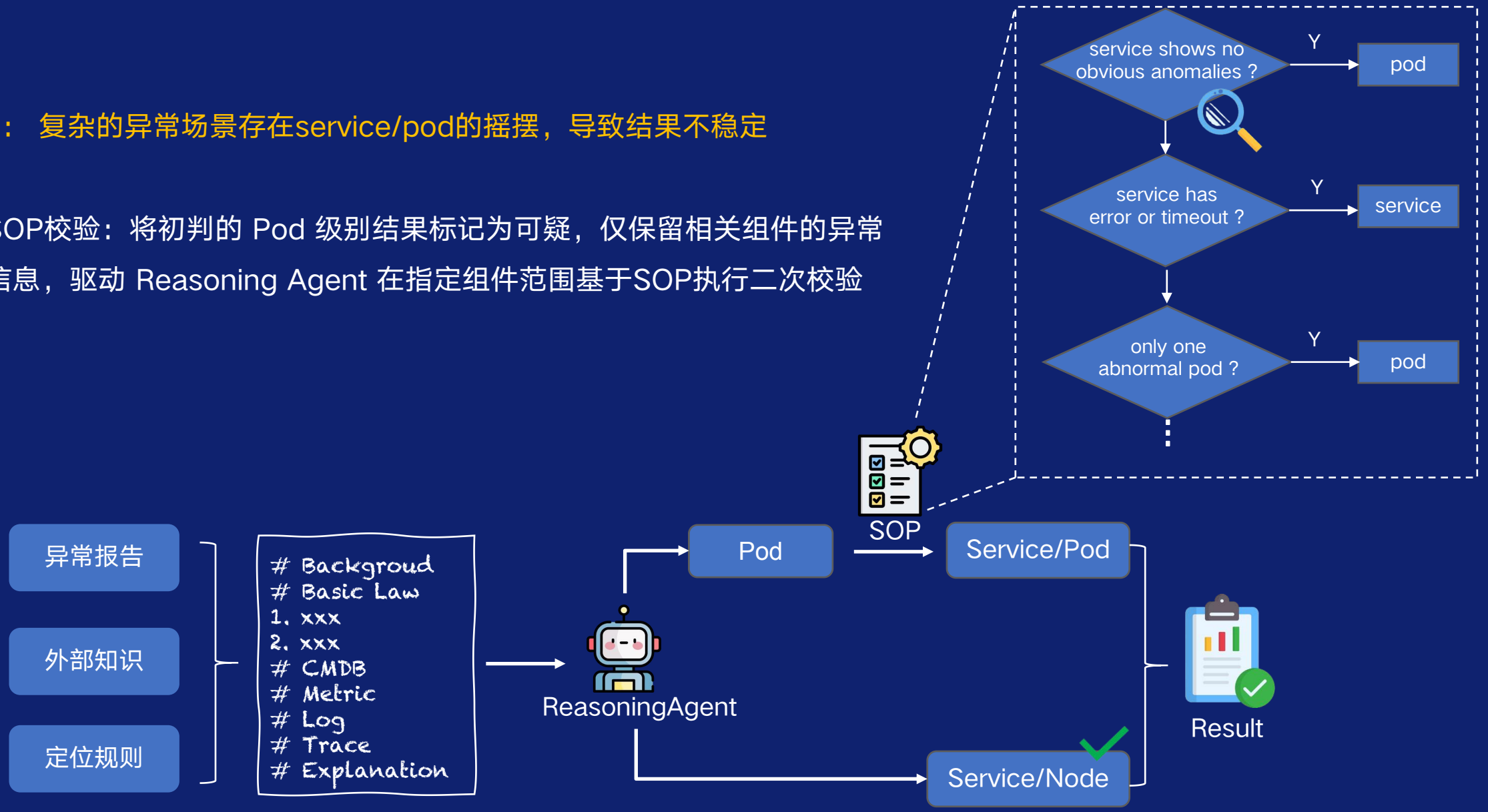
优势：

- 人工经验作为知识沉淀，自动化流程减少了信息传递过程中的主观性
- 发现隐形关联，能够挖掘出深层次、多维度的故障关联模式

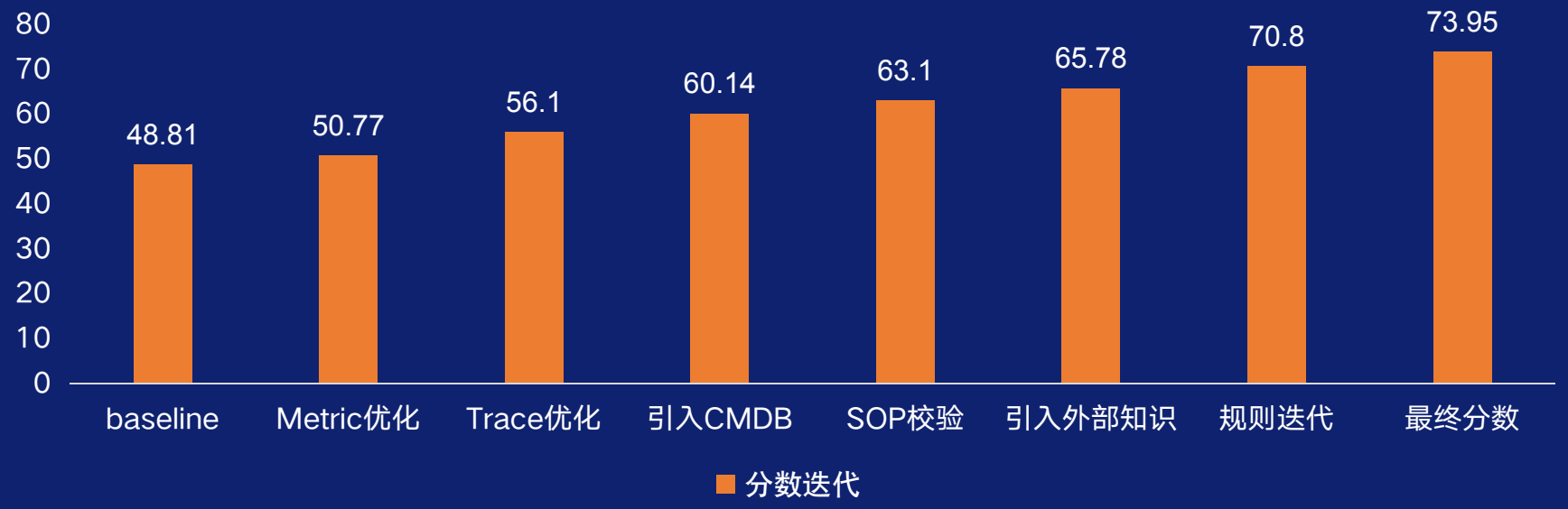
2.4 Reasoning Agent：SOP驱动的推理过程

问题： 复杂的异常场景存在service/pod的摇摆，导致结果不稳定

✓ SOP校验：将初判的 Pod 级别结果标记为可疑，仅保留相关组件的异常信息，驱动 Reasoning Agent 在指定组件范围基于SOP执行二次校验



整体效果



平均每个问题的耗时约为**1分钟**，**平均消耗Token数9200.5**

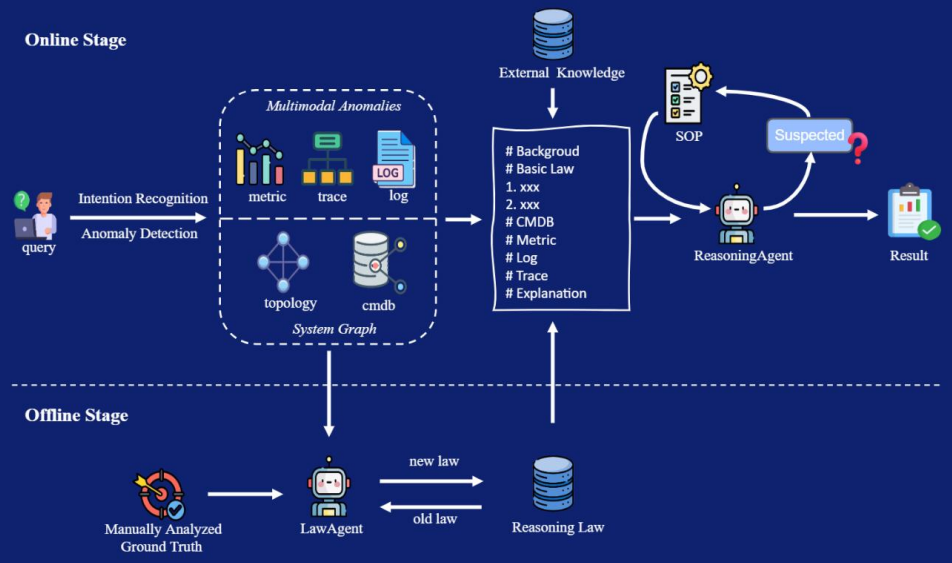
第三章节

总结与展望

总结与展望

总结

本方案通过实现多模态可观测数据的结构化异常描述、具有自迭代能力的定位经验规则库以及SOP辅助的定位校验，充分利用大模型的分析推理能力，在保证高准确率的同时，兼具了定位结果的稳定性和可扩展能力



- ✓ **结果准确且稳定**
  - 数据集上取得了73.95得分，多次运行分差不超过 ±1
- ✓ **实时定位效率高**
  - 1分钟内输出结果，满足实际运维中对快速响应的需求
- ✓ **系统可扩展性高**
  - 定位规则自迭代，适配不同业务场景的差异化定位需求

未来持续发力

致力于大模型在AIOPS领域更全面、更自主、更协同、更可靠的应用探索与落地

OpenAIOps AIOPS | 2025 CCF国际AIOps挑战赛
2025 CCF International AIOps Challenge

THANKS

主办单位：中国计算机学会（CCF）

承办单位：中国计算机学会互联网专委会、中国科学院计算机网络信息中心、中国移动研究院、清华大学

协办单位：华为2012实验室、阿里云、中兴通讯、中国移动九天团队、南开大学、西安电子科技大学、清华大学计算机科学与技术系、神州灵云