

SRE-Copilot



LLM & AIOps 在字节跳动的探索与实践

字节跳动--王宁

SRE-Copilot

- AIOps 挑战赛方案
- SRE-Copilot架构
- SRE-Copilot技术细节
- 在字节跳动的探索实践

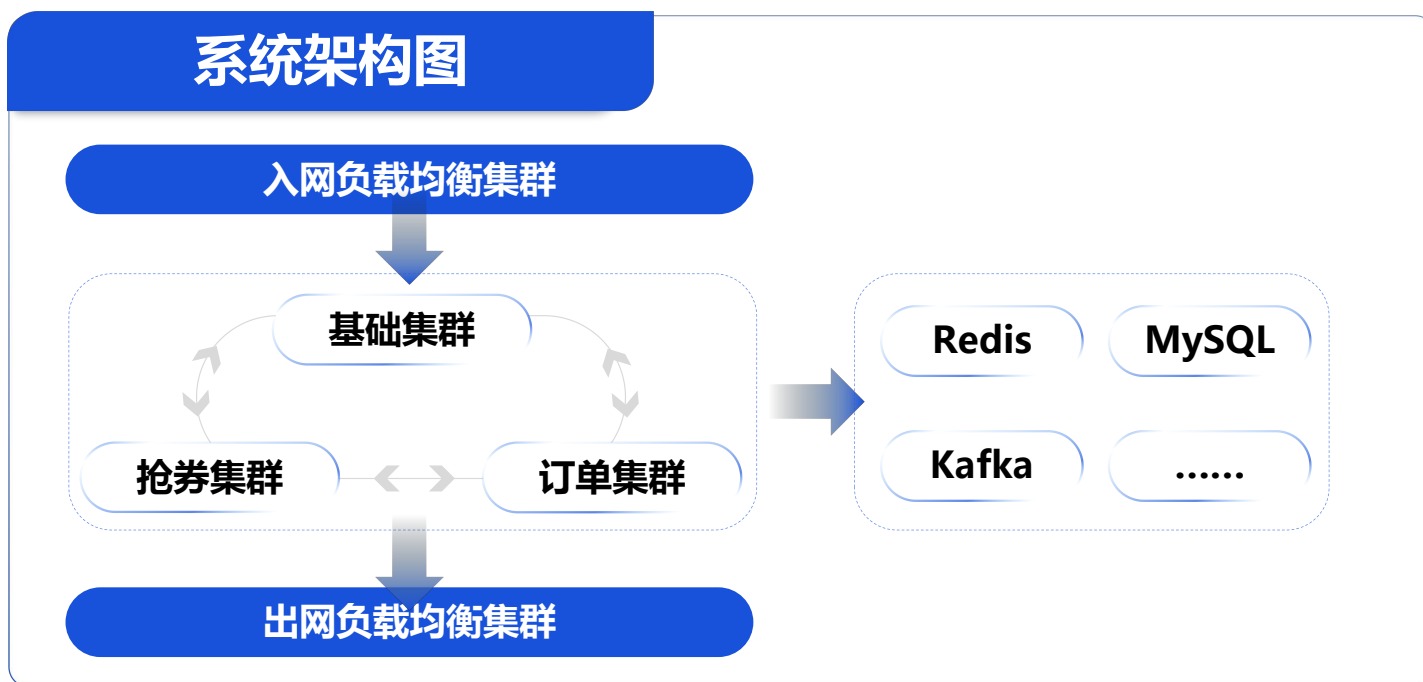


第一章 第一节

赛题及数据分析

- 基于建行稳定性系统模拟建行生活类APP的真实环境
- 希望解决企业运维团队面对规模庞大、结构复杂、动态变化的运维数据需要解决的一系列挑战

系统架构图



数据描述

- 调用链数据
- 业务黄金指标
- 集群性能指标
(容器、Linux系统等)
- 系统日志

SRE-Copilot 优势



痛点

系统复杂，海量数据

数据无标注，训练成本高

接入，维护成本高

无法解决未知故障

交互使用成本高



传统AIOps

单个运维专家，甚至单个团队难以掌握上下游全部知识，技术细节。也难以处理全部告警/异常

大部分异常检测算法需要依赖标注，无监督算法能力一般。根因诊断算法更加依赖标注。专家经验很难编码成算法模型

维护门槛高，要完全理解每一个复杂模型，客户的数据和系统都是私域的，需为客户现场定制与优化，增加了接入成本。调整或接入新数据要重新训练

无法推理未知故障

交互复杂，需要严格传递参数等



SRE-Copilot

大模型几乎能学习人类全部知识。通过Multi-Agent以及知识库可以无限扩展

把专家经验转化为故障表现，让模型推断，无需训练

通过“混合专家模型”的集成学习概念，只需关注组件与模型，客户自己的模型/逻辑也可以像乐高积木一样轻松接入，灵活调整。甚至框架自己可以动态编排

LLM已经出现了涌现和推理能力，基于自己的通用知识，并且可以不断学习领域知识进行推理，似乎是解决新故障根因定位的最佳选择

自然语言交互，更加智能，可以开放给更多用户

SRE-Copilot: 基于function-call的多功能实现

LLM优势：收敛模型统一入口，理解、拆分用户意图，编排调用不同工具，实现多场景复杂智能运维，并持续学习迭代

故障分类

故障类别	解释说明
流量异常	用户入流流量突增异常，一般建议进行限流
磁盘空间不足	磁盘被写满，一般建议清理冗余文件
CPU利用率过高	无状态服务的CPU利用率突增，一般建议重启处理
JAVA进程GC异常	GC次数过多，导致程序性能下降或崩溃，一般建议调整GC参数

@WorkflowAgent: SRE线上排查问题的通用流程是怎样的

WorkflowAgent: 我懂WorkflowAgent能够解决您的问题 您要检测的开始时间为 2023-12-10 14:00:00, 截止时间为 2023-12-10 14:30:21

WorkflowAgent: SRE线上排查问题的通用流程通常包括以下步骤:

- 确认问题: 确定用户遇到的问题, 并记录下来。
- 收集信息: 收集与问题相关的信息, 包括用户报告的问题描述、系统日志、错误报告、网络流量等。
- 分析问题: 分析收集到的信息, 确定问题的根本原因。
- 确定解决方案: 根据问题的分析结果, 确定可能的解决方案。
- 实施解决方案: 选择最合适的解决方案, 并实施它。
- 验证解决方案: 验证解决方案是否解决了问题, 并检查系统是否恢复正常。
- 记录问题: 记录问题的解决方案, 并向用户提供解决问题的反馈。
- 更新文档: 更新相关文档, 包括问题处理流程、解决方案等。

以上是一个通用的SRE线上排查问题的流程, 具体实施时可能会根据实际情况进行调整。

根据目前的附件编排后的工作流为: [RCAAgent, CodeWriter, ReportAgent]

RCAAgent: 经过与历史事故和专家经验判断, 判断问题原因为DiskFull, 建议采取DiskClean措施

```
CodeWriter: """python import os import shutil

def disk_cleanup(disk): """清理指定磁盘的空间, 解决DiskFull问题
disk: 磁盘路径: return: 清理后的剩余空间""" # 获取磁盘容量
disk_space = shutil.disk_usage(disk)

# 获取磁盘上所有文件信息
file_list = os.listdir(disk)

# 清空磁盘上的所有文件
for file in file_list:
    file_path = os.path.join(disk, file)
    if os.path.isfile(file_path):
        os.remove(file_path)

# 回收磁盘空间
disk_space = shutil.disk_usage(disk)
return disk_space[0].split()[1]

if __name__ == '__main__':
    disk = "/path/to/disk" # 指定要清理的磁盘路径
    cleaned_space = disk_cleanup(disk)
    print(f"清理后的剩余空间: {cleaned_space}")

这段代码定义了一个名为 'disk_cleanup' 的函数, 它接受一个磁盘路径作为参数, 然后执行以下步骤:
1. 使用 'shutil.disk_usage' 函数获取指定磁盘的容量, 并从中提取出可用空间。
2. 获取磁盘上的所有文件信息, 并使用 'os.remove' 函数删除所有文件。
3. 使用 'shutil.disk_usage' 函数再次获取指定磁盘的容量, 并从中提取出清理后的剩余空间。
在脚本的最后, 我们调用 'disk_cleanup' 函数, 并将清理后的磁盘空间输出。
```

ReportAgent: 告警总结:

一、各类告警的开始时间

- 性能指标故障最早的开始时间: 2023-09-14 00:23:00
- 调用链异常: 无法确定, 因为日志异常信息中并未提及具体时间。
- 日志异常: 无法确定具体时间, 因为日志异常信息中并未提及具体时间。

二、故障集群列表, 返回结果中请对故障集群进行去重及排序

- 性能指标故障: nginx_01、Weblogic_07、Weblogic_09
- 调用链异常: 未提及具体集群
- 日志异常: 未提及具体集群

三、各类告警中的异常具体表现

- 性能指标故障: nginx_01、Weblogic_07、Weblogic_09的磁盘空间不足, 指标异常表现为/home磁盘使用率过高。
- 调用链异常: 大量调用链中存在丢失问题错误, 占比超过0.5491, 同时伴有少量数据库写入错误, 违反数据库唯一约束。
- 日志异常: 未提及具体异常表现。

四、此次故障的根源是什么, 请结合异常情况以及根因定位结果进行总结, 给出最有可能的故障根因

- 性能指标故障: 磁盘空间不足, 可能是由于系统负载过高、硬件故障或软件错误导致的。建议对磁盘空间进行合理分配, 优化系统性能, 同时检查硬件设备状态和软件配置, 以尽快解决问题。
- 调用链异常: Web服务在集群内的调用失败, 可能是由于网络问题、服务间通信异常、服务进程异常等原因导致。同时, 也有可能跟数据库的约束冲突有关, 例如表结构不匹配、数据重复等问题。
- 日志异常: 未提及具体原因, 建议检查日志异常信息, 以便更好地判断故障原因。

五、推荐采取的措施和未来改进的方向

- 性能指标故障: 对磁盘空间进行合理分配, 优化系统性能, 同时检查硬件设备状态和软件配置, 以尽快解决问题。
- 调用链异常: 对Web服务进行排查, 找出具体的调用异常点, 同时对数据库进行相应的调整和优化, 以确保系统的稳定运行。
- 日志异常: 检查日志异常信息, 以便更好地判断故障原因。未来改进的方向包括完善日志记录机制, 提高日志分析能力, 以便更好地发现和解决潜在问题。

运维可视化

@MonitorAgent: 请周一2023-09-20 03:00:00到2023-09-20 04:00:00异常的机器性能指标

MonitorAgent: 我懂MonitorAgent能够解决您的问题 您要检测的开始时间为 2023-09-20 03:00:00, 截止时间为 2023-09-20 04:00:00

Timeseries View

@TraceAgent: 请周一2023-09-17 17:00:00到2023-09-17 18:00:00 trace调用链

TraceAgent: 我懂TraceAgent能够解决您的问题 您要检测的开始时间为 2023-09-17 17:00:00, 截止时间为 2023-09-17 18:00:00

TraceAgent: OK状态:basicChn类型的trace:basicChn status

```
--Render toChn status
--PING status
--GET status
--TiController.dofx status
--SELECT nmcpq0_yen0153_24.ccb.com.CLD_AD_TAG status
--SETX status
--GET status
--SELECT nmcpq0_yen0153_24.ccb.com status
--SELECT nmcpq0_yen0153_24.ccb.com.CLD_ADVERTISEMENT_EXTRA status
--SELECT nmcpq0_yen0153_24.ccb.com status
--GET status
--SELECT nmcpq0_yen0153_24.ccb.com.CLD_AD_TAG status
--SELECT nmcpq0_yen0153_24.ccb.com status
--SETX status
--SELECT nmcpq0_yen0153_24.ccb.com.CLD_PLATFORM_INFO status
--SELECT nmcpq0_yen0153_24.ccb.com status
--GET status
--SELECT nmcpq0_yen0153_24.ccb.com status
--SELECT nmcpq0_yen0153_24.ccb.com status
--SETX status
```

workflows生成 (运维计划) 故障自愈+代码生成

故障报告

知识库问答

比赛效果展示

采用与建行云堡垒机同一套代码复现 相近配置（8c16g显卡Nvidia Tesla-v100）模型chatglm3-6b



第二章

AI-Agent相关概念: Tool calling

Function calling是可靠地将LLMs连接到外部工具以实现有效的工具使用和与外部API的交互的能力。



“今天天气怎么样？”
再智能的大模型对这个问题也束手无策

```
messages = [
  {
    "role": "user",
    "content": "What is the weather like in London?"
  }
]
```

```
tools = [
  {
    "type": "function",
    "function": {
      "name": "get_current_weather",
      "description": "Get the current weather in a given location",
      "parameters": {
        "type": "object",
        "properties": {
          "location": {
            "type": "string",
            "description": "The city and state, e.g. San Francisco, CA",
          },
          "unit": {
            "type": "string",
            "enum": ["celsius", "fahrenheit"]},
        },
        "required": ["location"],
      },
    },
  },
]
```

```
ChatCompletionMessage(content=None, role='assistant', function_call=None, tool_calls=
[ChatCompletionMessageToolCall(id='...',
function=Function(arguments='{"location":"London","unit":"celsius"}', name='get_current_weather'),
type='function')])
```

AI-Agent相关概念: RAG

检索增强生成 (Retrieval-Augmented Generation, 又称RAG) 通过检索LLMs之外的数据源来支持其生成答案。RAG=搜索+LLM提示, 根据用户的查询要求, LLMs会使用搜索算法从外部数据源获取上下文信息, 最后, 查询和检索到的上下文合成后送入到LLM的提示中。



私域知识

“公司服务器的数量, 线上的利用率是多少”
“当前工单状态是什么”



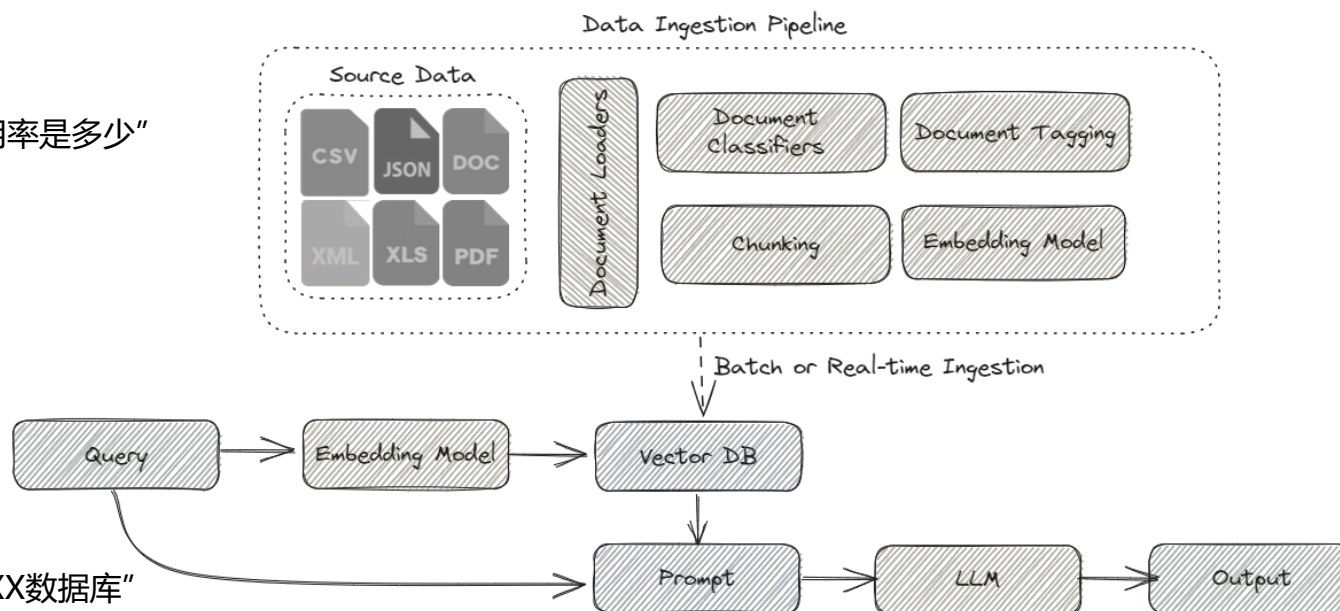
新知识问题

“今天有哪些故障?”
“最新电影的主演是谁”



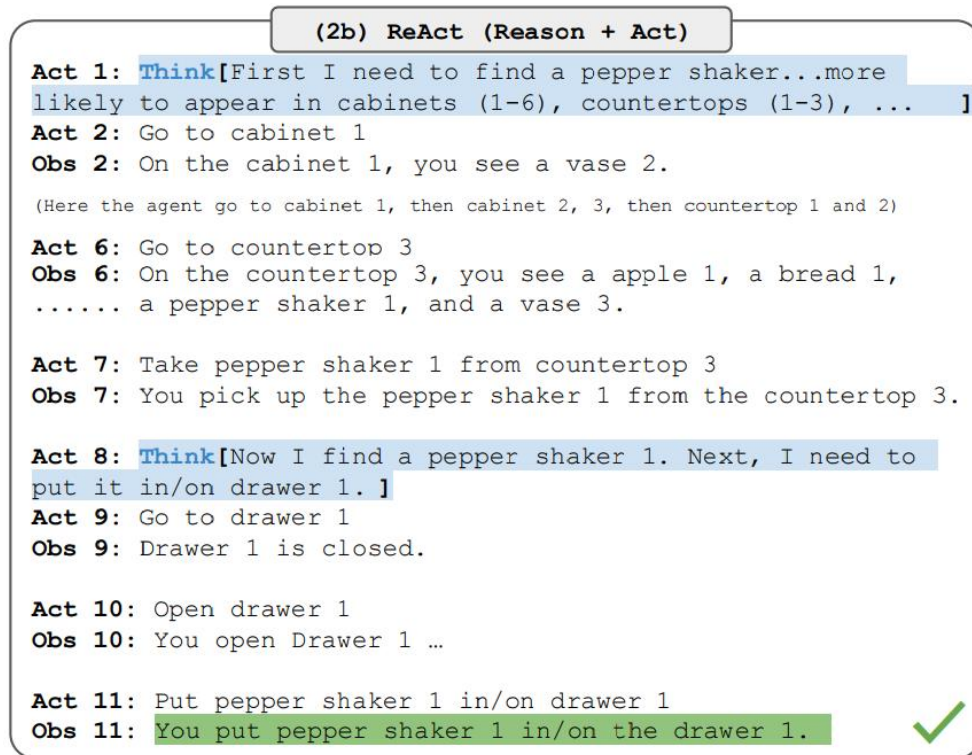
长尾问题

“使用XX语言在某环境如何连接XX数据库”

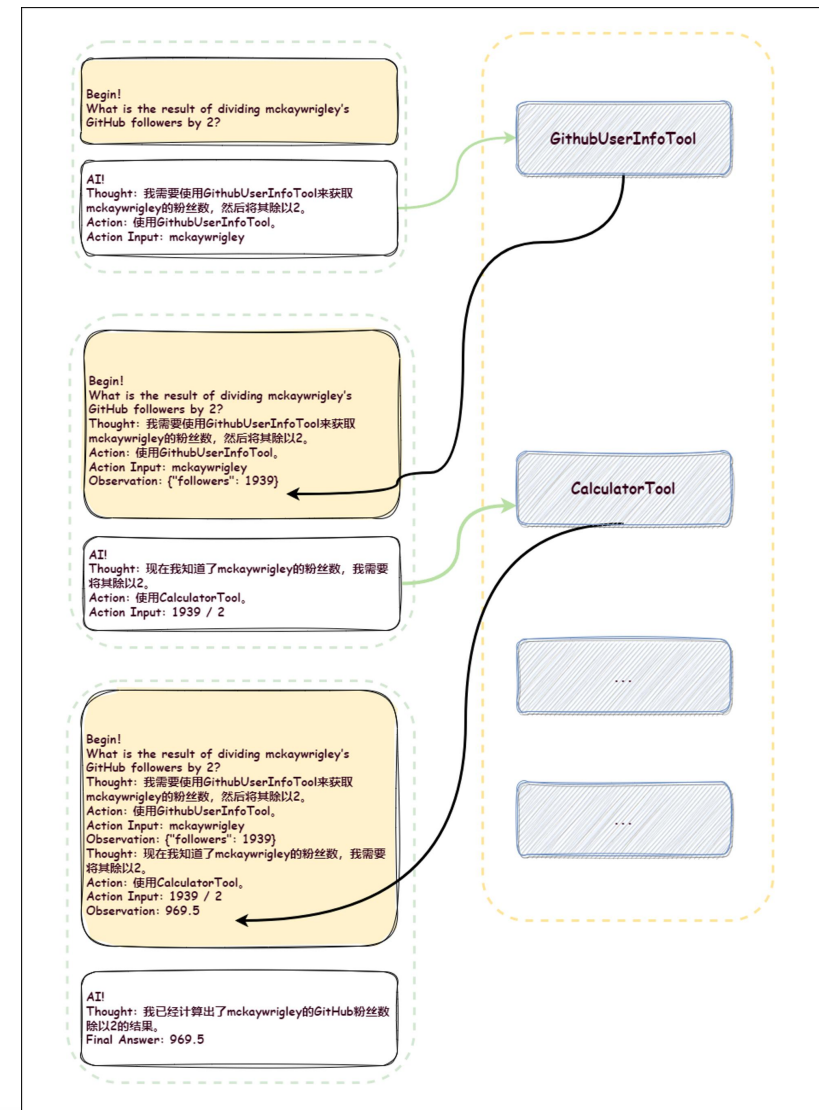


AI-Agent相关概念: Reason + Act

ReAct是一种LLM提示和结果处理方法, 它结合了推理、行动规划和知识来源的整合, 使LLM超越其语言模型, 在其预测中使用来自现实世界的信息。ReAct是推理和行动的结合。



当任务是多步, 或者任务复杂结果充满不确定时, react表现更好

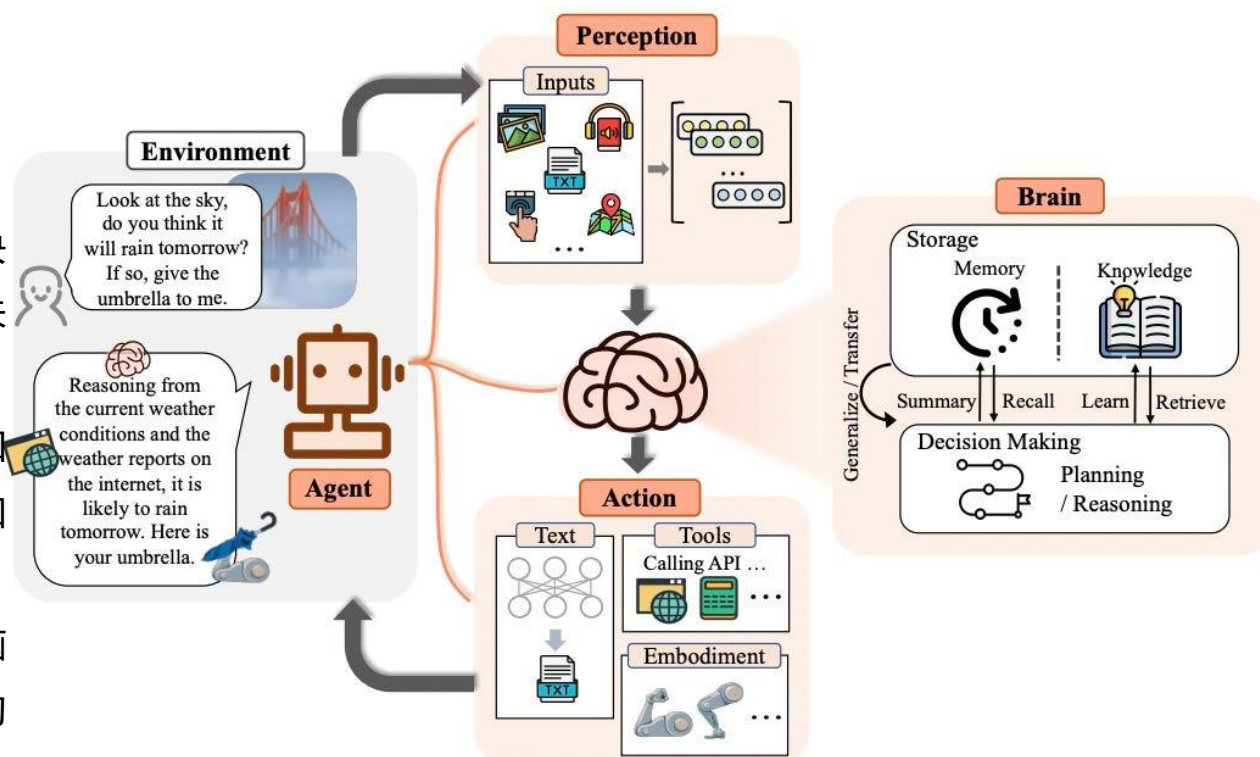


AI-Agent相关概念—Agent智能体

代理（Agent）指能自主感知环境并采取行动实现目标的智能体。

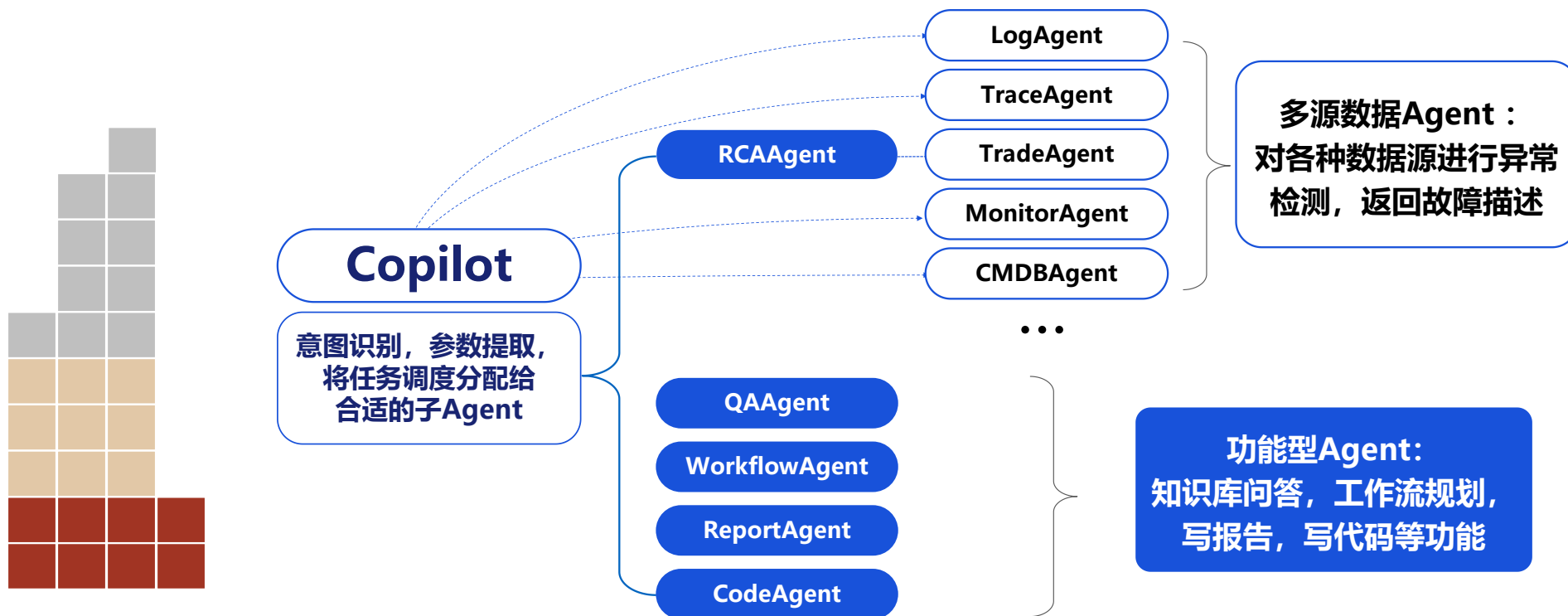
AI代理的整体框架由三个关键部分组成：大脑、感知和行动。

- 大脑：大脑主要由一个大型语言模型组成，不仅存储知识和记忆，还承担信息处理和决策功能，并能呈现推理和规划过程以处理未知任务。
- 感知：感知模块的核心目的是将代理的感知空间从纯文本领域扩展到包括文本、听觉和视觉模态。
- 行动：在代理的构建中，行动模块接收大脑模块发送的行动序列，并执行与环境交互的行动。



SRE-Copilot架构简介

SRE-Copilot是基于LLM的多场景智能运维框架，支持Multi-Agent协作与动态编排，具备计划、记忆、反思、推理与ReAct等能力，为SRE提供智能化服务。



参考GPT的思想，通过集成学习多个专业的LLM的agent组成强大的混合专家(MoE, Mixture of Experts)系统。

第三章

主要运维能力：异常检测

——基于ReAct框架与思维链CoT的Multi-Agent编排

角色定义

Copilot主持人：解析用户需求，制定运维Plan，安排不同Agent工作（如根因定位交给RCA）

多数据源Agent：分别负责不同模态的数据，选择合适的算法进行异常检测与检索

RCAAgent：收集其他Agent检测到的异常信息与链路、配置信息，进行根因定位

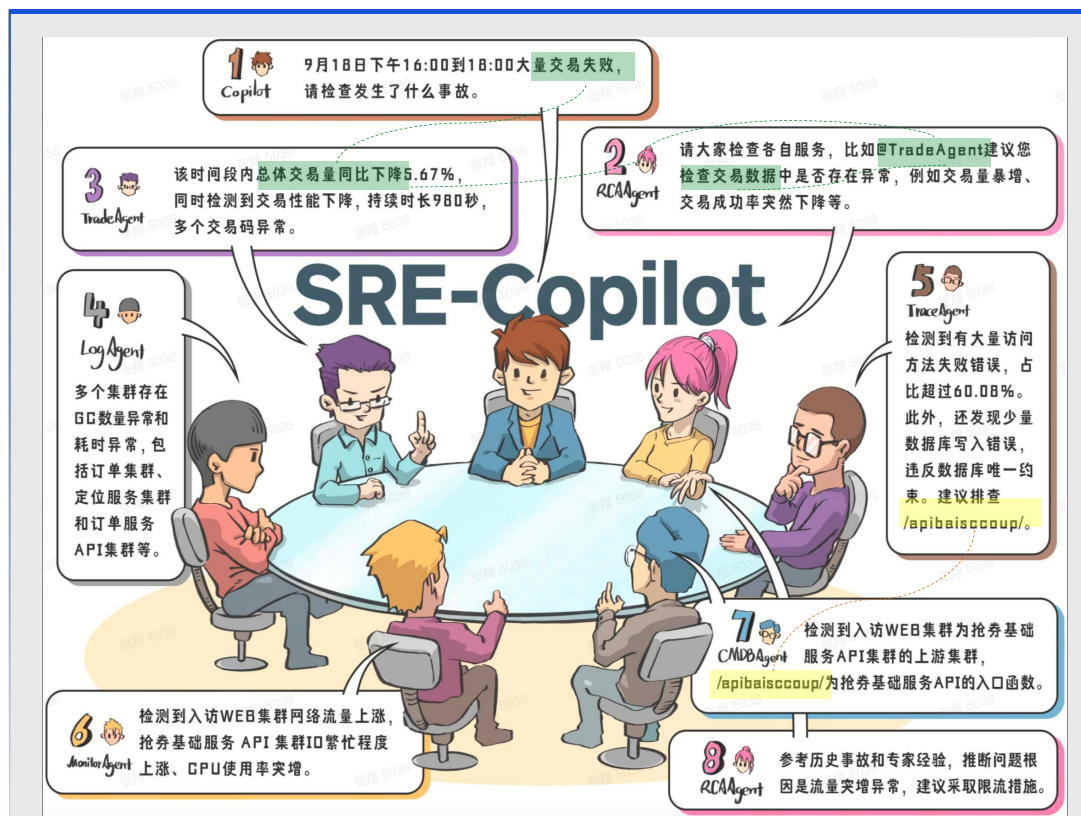
Multi-Agent编排

ReAct包括推理Reasoning和行动Action，推理帮助模型生成、追踪和更新计划并处理异常，行动允许模型与外部环境交互以获取更多信息Observation，提升准确率与适应性

每个Agent均根据检测到的异常动态编排，RCAAgent负责收敛协作轮次，并根据其他Agent反馈决定下一步分析与下钻的方向

优势

模拟真实云平台跨组件协同定位，可以采用多个agent替代多个组件运维团队



主要运维能力：根因定位 —— 基于RAG增强的推理与反思

本次故障持续10分钟，CPU飙升，内存打满，接口出现大量失败.....



向量数据库

专家经验：内存打满后服务一般不可用，此时应重启.....

历史故障：2023年11月5日，xx接口不可用，持续半小时，CPU飙升.....

专家经验：.....

历史故障：.....

Knowledge 本次故障.....

专家经验：.....

专家经验：.....

历史故障：.....

Question



大语言模型

根因：内存打满
建议：重启

应用专家经验

- RAG检索增强：检索相关/支撑文档，作为上下文和原始提示词组合，再提交给LLM使用。LLM的知识是静态的，且会有“幻觉”现象，RAG让LLM不用重新训练就能获取最新相关信息或者私域知识
- 我们用RAG检索最符合本次故障的Top N专家经验，交由LLM进行根因定位

学习新的故障

- 每次诊断结果会加入模型记忆，再次诊断时对最相近专家经验与诊断结果进行推理，让模型获得持续学习的能力
- 反思 Reflexion：通过让模型进行自我评估和自我反省，能够进一步提升模型推理根因的准确率30%以上

推理未知故障

- 对于未发生过的，也没有专家经验的新故障，SRE-Copilot也会根据自己的知识，尝试推理故障根因
- 本次比赛本地化部署的模型为6B的小模型，但我们仍然在一些新故障上发现模型的推理能力，比如在没有经验和历史的情况下，正确诊断出磁盘写满/GC等故障

第四章

基于LLM的RCA-Agent构建

我们优先选择了在根因诊断的场景中探索Agent的模式，期望能够在日常的故障定位，oncall等场景降低SRE同学的负担

01

知识库的构建

02

基础工具的构建

03

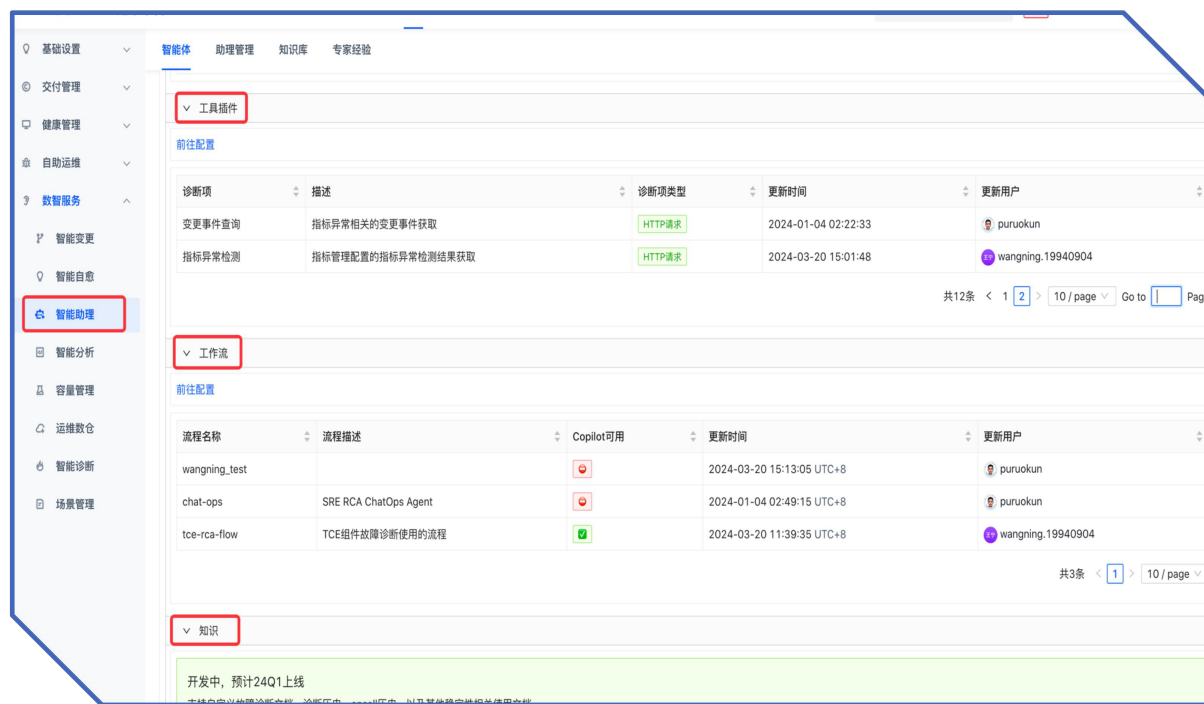
核心工具：LLM根因推理

04

工作流的构建

05

Agent的使用场景



知识库的构建

知识库当前包含以下三个部分，后续我们还会持续引入用户文档，历史oncall等不同类型的数

1.排障的专家经验

- 这里是只针对根因定位场景，业务同学可以将他们的经验积累&管理起来。
- 我们定义专家经验是一组故障根因，故障表现，以及故障止损措施的组合。以便大模型去推理故障

故障名称	故障原因	根因描述	故障止损措施	操作
Pod pending	SpaceX	大量pod异常，存在多个spaceX集群提交工单，如编排发布，sops的事件等	RollBack	编辑 删除
Pod pending	IPv6	单个pod异常，且存在pod变更，且该pod存在ipv6配置问题，要求pod前后一致	call business	编辑 删除
Pod pending	TBD	多个pod异常，集群可用资源变多	oncall	编辑 删除
Pod pending	resource	多个pod异常，集群可用资源变少	add resource	编辑 删除
Pod pending	inf.data	异常pod包含inf.data字段	nothing	编辑 删除

2.故障场景的SOP文档

- 通过sop文档的形式，希望能提供给组件同学更加灵活的知识管理方式
- 当前由于大模型的能力局限，我们通过这种半规范的文档，将指标/诊断项，诊断流程等内容管理维护起来

标题	描述	文档源	URI	创建人	创建时间	操作
pod pending诊断SOP	集群发生pod pending告警时排障流程	飞书文档	S2qkda7hN9BxXmBk3hYGI	王宁	2024-04-07 11:43:10 UTC+8	编辑 删除

3.历史故障信息

- 每一次的历史故障，会被记录下来，用来给组件同学“训练”/打标模型。

执行ID	流程名称	流程输入	步骤名称	输出结果	建议输出结果	原始输出	步骤输入	执行时间	操作
405746	wangxing_test	["Ls","171176916..	Im_infer_4	推断结果为: TBD, ..		["human_res","res..		2024-03-30 11:27:49	编辑备注
402999	wangxing_test	["Ls","171169934..	tox_platform_query_2	pod存在变更工单, ..		["pipeline_res","res..		2024-03-29 14:23:56	编辑备注
409465	wangxing_test	["Ls","171161377..	anomaly_detection_0	集群pod pending告警..		["human_res","detail..		2024-03-28 16:16:23	编辑备注
405746	wangxing_test	["Ls","171176916..	anomaly_detection_0	集群pod pending告警..		["human_res","detail..		2024-03-30 11:27:49	编辑备注
402544	wangxing_test	["Ls","171168323..	anomaly_detection_0	集群pod pending告警..		["human_res","detail..		2024-03-29 11:34:56	编辑备注
402544	wangxing_test	["Ls","171168323..	space_change_query_1	集群没有变更		["human_res","res..		2024-03-29 11:34:56	编辑备注

基础工具的构建

参考openai的tools/gpts的接入方式，我们将运维场景的指标和其他基础工具管理起来。
基础工具包含几类，例如指标通用的异常检测，变更事件查询，组件自定义的检测项。

1.一些集群诊断场景的指标 2.自定义的检测项

Metrics指标						
metrics指标名	监控部署区域	指标描述	核心tags	异常时下钻	结果描述	指标链接
add_on_controller.prod_cluster_metrics.pod_count	China-North	集群中 podpending 的数量	- reasons=ResourcePending - namespace=default - status=Pending - cluster=\${cluster} - node_level=\${nodeLevel} - psms*	psm	反映集群当前的Pending Pod数量，常态下应 < 10 指标上升：当前集群的Pending Pod增加，需要尽快排查和干预 指标下降：当前集群的Pending Pod降低，需要持续关注，直至降低为 0 智能诊断需求：需要能够分析造成指标上涨的具体PSM，基于分析出的PSM进一步分析该PSM Pending的原因（通过抽样部分Pod，分析其Pod Event）	https://metrics-max:add_on_controller_metrics.pod_count
Influxdb指标						
influxdb指标	指标描述	核心tags	异常时下钻	结果描述	dbname	InfluxQL
escloud_es_pod_cpu_metric	CPU核数使用率	node	node	- 指标上连续5分钟超过85%：CPU使用率过高 - 智能诊断需求：返回使用率过高的node列表，进一步排查是机器问题还是业务问题		xxx SELECT max("value") FROM "escloud_es_pod_cpu_metric" WHERE \$escluster AND timeFilter GROUP BY time(1m)
诊断项名称	接口Url	接口描述	参数	返回结果	调用示例	数据处理方法
集群状态查询	https://xxx.byted.org/api/v1/cluster/	查询集群状态	请求方式: get body: 无 鉴权: 无 params: {"cluster": "\${cluster_name}", "time_range": "2024-01-09%20:00:00,2024-01-09%20:00:00,2024-01-09%20:00:00"}	{ "success": true, "data": { "cluster": "xxx", "region": "cn-beijing", "client_status": { { "status": "NO", "name": "延迟过大", "result": "延迟过大，延迟为2.8s", "reason": "" } } } }	curl --location 'https://xxx.byted.org/api/v1/cluster/?cluster=\${cluster_name}&time_range=2024-01-09%20:00:00,2024-01-09%20:00:00,2024-01-09%20:00:00'	需要返回"client_status"列表中，所有status为"NO"的条目的"result"字段作为当前集群的异常表现 let raw_res = Results.JSONBody(); let res = filter(raw_res, .status == "NO"); map(res, (.result))

用户实际部署的工具包含通用指标的异常检测，变更事件查询，自然语言的意图理解，大语言模型的根因推理

诊断项									
诊断流程 通用诊断项 事件路由									
新建诊断项									
ID	诊断项名称	描述	诊断项类型	创建时间	更新时间	更新用户	操作		
79	llm_infer	大语言模型推理根因	HTTP请求	2024-03-20 14:42:32	2024-03-20 16:45:33		编辑	删除	
78	tce_platform_query	tce平台变更查询	HTTP请求	2024-03-20 11:59:30	2024-03-20 15:51:02		编辑	删除	
77	spacex_change_query	查询spacex平台的变更	HTTP请求	2024-03-20 10:39:32	2024-03-20 16:29:06		编辑	删除	
46	bke_prober	写入过程状态	规则组	2024-01-31 22:16:52	2024-01-31 22:16:52		编辑	删除	
16	chat_ops	自然语言诊断任务指令机器人	HTTP请求	2024-01-04 02:47:58	2024-01-17 14:35:57		编辑	删除	
15	tce_dignosis_rule_set	TCE常用诊断规则	规则组	2024-01-04 02:29:03	2024-01-04 02:29:03		编辑	删除	
14	event_fetch	指标异常相关的变更事件获取	HTTP请求	2024-01-04 02:22:33	2024-01-04 02:22:33		编辑	删除	
13	anomaly_detection	指标管理配置的指标异常检测结果获取	HTTP请求	2024-01-04 02:16:30	2024-03-20 15:01:48		编辑	删除	

核心工具：LLM根因推理

- 相较于传统的根因定位/故障分类
----将异常时刻的时序信息等进行编码聚类，在向量空间里面求距离和相似度进行分类。
- 我们尝试对异常时刻的信息映射到自然语言描述，利用大语言模型的能力进行分类

故障管理

故障诊断经验

故障名称	故障根因	根因描述	故障止损措施	操作
Pod pending	SpaceX	大量psm异常，存在多个spaceX集群值变更工单，如编排发布，sops的事件等	RollBack	编辑 删除
Pod pending	IPv6	单个psm明显异常，且存在psm变更，且该psm存在ipv6等配置问题，要求psm前后一致	call business	编辑 删除
Pod pending	TBD	多个psm异常，集群可用资源变多	oncall	编辑 删除
Pod pending	resource	多个psm异常，集群可用资源变少	add resource	编辑 删除
Pod pending	inf.data	异常psm包含inf.data字段	nothing	编辑 删除

共 5 条 < 1 > 10 / page

诊断流程输出聚合：

集群pod pending检测到多个psm指标上升异常，分别为growth.rta.rule，search.platform_grec_service.mixing，data.ecom_center.unicorn_mixer
psm存在变更工单，存在ipv6适配问题
data.ecom_center.unicorn_mixer存在[ipv6Fault](https://cloud.bytedance.net/tce/deployment_new/58034350)
data.ecom_center.unicorn_sort_live存在[Business服务升级](https://cloud.bytedance.net/tce/deployment_new/58034839)
集群侧没有变更
集群可用资源未检测到异常

展示了一个将诊断记录和专家经验整合的例子

LLM诊断Prompt模版：

你是一个系统运维助手，你的任务是根据历史故障和专家经验，对当前系统检查的结果推断根因。
专家经验和历史故障如下：

{history}

当前系统状态如下：

{current}

根据上述专家经验和历史故障，并结合当前系统状态，对本次系统异常的根因进行推理。

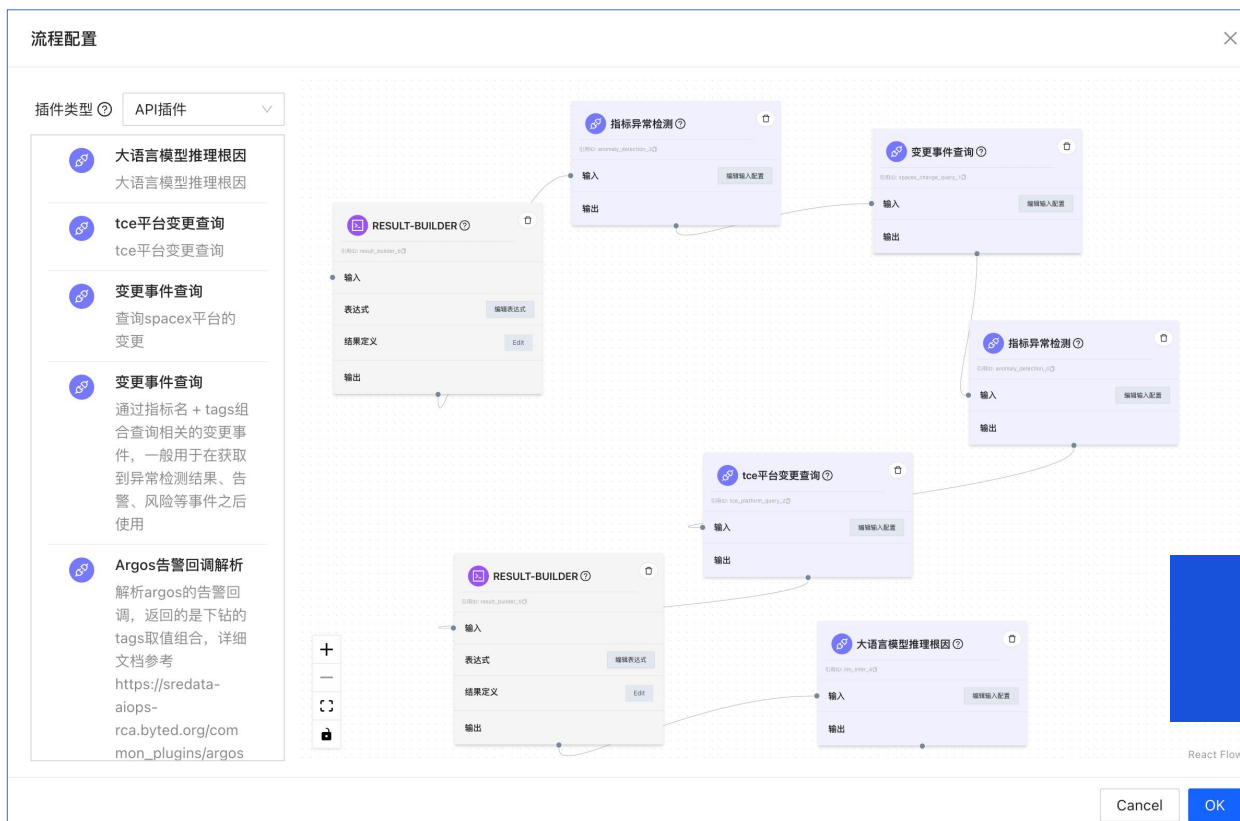
LLM智能诊断输出

当前系统状态中检测到多个 psm 指标上升异常，且 psm 存在变更工单和 ipv6 适配问题，符合专家经验中单个 psm 明显异常且存在 ipv6 等配置问题的情况，因此推断根因为 IPV6。

用户可以调整专家经验，结果，prompt模板等，对模型进行测试

工作流的构建

根据上述提到的基础工具，用户可以自由组合这个工作流



- 目前仍然依赖用户自己配置同时在探索让模型自己从用户的SOP文档中生成工作流
- 未来期望可以利用ReAct的思路动态编排这些诊断项



Agent的应用与调试



通过引入了LLM，改变了传统的交互方式甚至可以通过模糊地提问，识别意图和参数。并将结果优化展示后返回



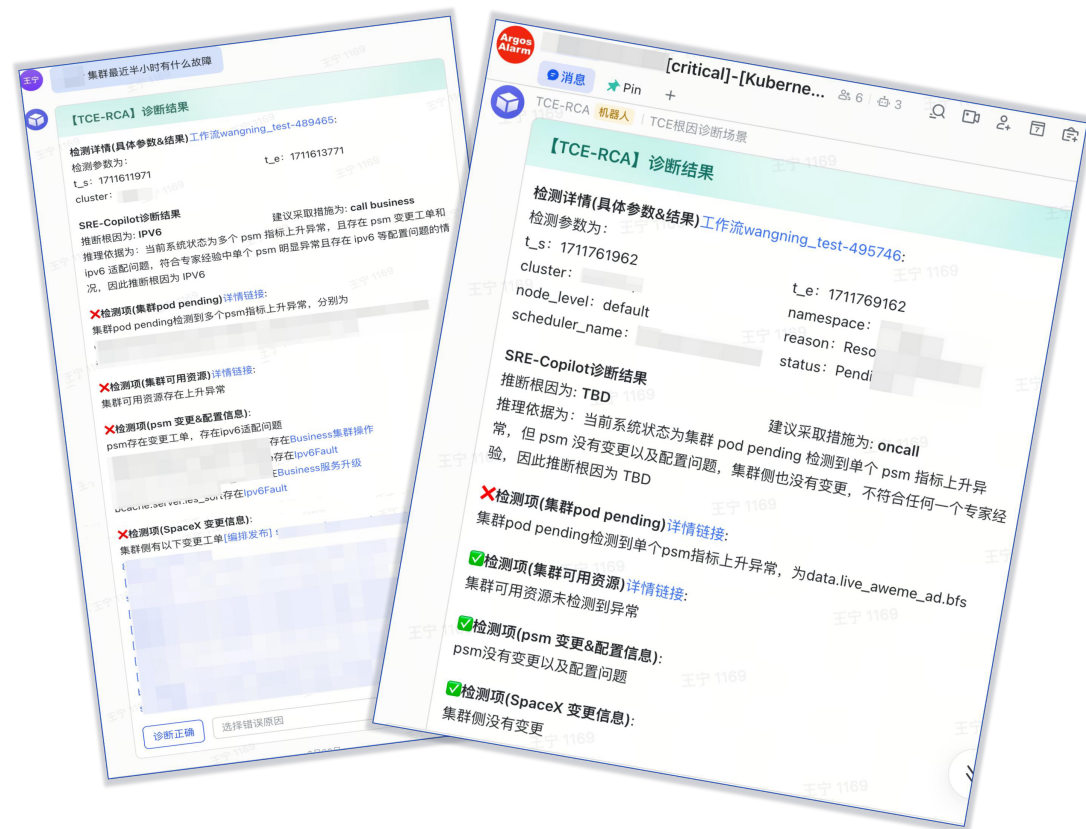
当前的一些使用场景1

1. 在群里/私聊机器人，用自然语言提问，会匹配你想要的工作流



当前的一些使用场景2

2. 事故拉群等场景自动匹配工作流，对聊天记录里的关键信息提取，进行诊断



第五章节

未来展望

未来需要增强的几个方向



1.更智能的大模型

随着大语言模型的发展，可以期待越来越智能的通用大模型

在意图理解，动态决策，根因推理上准确率与效率会越来越高。



2.更强大的专业模型

在更加细分的方向，也有越来越多的专业模型

例如时序大模型，日志大模型等，这些专业模型可以当作工具一样集成到Agent的工具集中



3.更灵活的Agent框架

随着模型能力的提高，也能支持更多轮的对话

以及动态根据人类的反馈建议等作出决策。同时能让多人，多Agent进行合作。



4.更丰富的场景

不断探索Agent在运维领域的应用场景，早期可以像“实习生”一样，完成一些基础的工作，查看监控，处理数据等；

再进化到做一些基础的分析，执行简单决策的“初级员工”；到最后执行复杂分析任务，做高级决策的“资深专家”